Medical History Database and Correlation System (MHDCS)

[A framework for demonstrating BON/Eiffel Object Oriented software design and development methodologies.]

Final Report

Sam Sampson Lightstone
Light@ca.ibm.com

July 23, 1999

TABLE OF CONTENTS

STATEMENT OF PURPOSE AND REQUIREMENTS

For a complete description of the project requirements, please see: *“Prototype For Inter-Hospital Patient Database ‘Medical History Database and Correlation System’ (MHDCS), Functional Requirements”* dated May 6, 1999. A brief summary of the high level goals for this system is repeated here.

Due to the importance of personal and family medical histories in providing patient care, the ability to maintain patient hospital events, and correlate medical histories of blood relatives is extremely important.

Few hospitals today maintain structured computerised patient histories, and there is little data collected from patients regarding family histories (related illnesses between relatives). Most records are maintained in paper files, with hand-written notes. Entries are frequently illegible and out of sequence, making review of patient history difficult and error-prone. Moreover, in many cases, due to the private nature of personal illness, it is common for individuals to be uninformed regarding medical histories of relatives. Therefore, while close relatives may have highly relevant medical histories that can reveal important considerations in patient treatment, it is frequently the case that such relationships go undiscovered.

The MHDCS software shows a simple programmatic solution for solving both of these issues:

1. Comprehensive maintenance and query capability of patient medical histories.
2. Correlation of patient medical issues with those of close family relatives.

The development of database software for maintaining and correlating patient/relative medical histories across regional hospitals would be an invaluable asset in providing superior medical care. As well, over time, a broad collection of data would enable “data mining” which may surface unexpected medical relationships previously unrecognised. Use of this data for data mining could lead to advances in medical treatments and patient care.

The purpose of this project is to design and develop a prototype of this system, sufficient for demonstration purposes, as a proof of concept entity. This prototype may be used as a demonstration vehicle for marketing and sales of MHDCS.

DESIGN OVERVIEW

AUDIENCE AND ASSUMPTIONS

This document describes the high and low level design details of the prototype system for the “Medical History Database And Correlation System” (MHDCS). The prototype system has been designed in the Eiffel language. Readers of this document are presumed to be fluent in Eiffel and BON, and have a background in Object Oriented software construction.

METHODOLOGY

MHDCS has been designed using standard Object Oriented software construction methodologies, through the Eiffel suite of development tools, produced by Interactive Software Engineering Inc, Santa Barbara, California. An emphasis is placed on software reuse (particularly through use of **EiffelTime** and **EiffelStore** clusters), self documentation, and design by contract. The use of OOD in this design is expected to also demonstrate the open closed principle and the uniform access principle.

HIGH LEVEL SYSTEM DESIGN, AND CLUSTER OVERVIEW

The notion of a data repository in EiffelStore

EiffelStore, the Eiffel cluster for interfacing to database systems (including relational and object oriented databses) includes the notion of a "data repository". A data repository maps directly to a single relational table. This abstractions provides a powerful, and easy machanism for performing table operations (select, insert, delete) through Eiffel objects. The repository capability is encapsulated in the **EiffelStore** class named **DB_REPOSITORY**.

Aside from providing a mechanism for Eiffel application developers to access relational database services at a table level, repositories have the outstanding feature of mapping Eiffel objects directly to relational tables! This makes it possible, with very few programming steps, to insert the attributes of an Eiffel object directly into corresponding table columns, or alternatively to select directly from a relational table into an Eiffel object. It is important to note that this capability of object/table mapping is a special and notable feature of EiffelStore, and not at all an industry-standard ODBC, or RDBMS capability.

Mapping between Eiffel objects and relational tables is performed through the DB_REPOSITORY services, and is dependant on the conformance to the repository "double matching rule". The double matching rule reads as follows:

1. A table column and an object attribute must match in name
2. A table column and an object attribute must be compatible in type.

Overview

MHDCS is designed using four essential native clusters. These include:

- An application cluster **"APPLICATION"**, which includes instantiation of the three additional clusters, and which controls state transitions for the application

- An interface cluster **"PANEL",** containing classes for UIs, and handshaking with lower level clusters for performing database services.

- A data repository cluster **"DATABASE REPOSITORIES",** which performs database operations, including connection, session monitoring, repository operations, such as table query, and table insert.

- A cluster containing mapping classes **"ENUMERATED MAPS",** for enumerated types such as gender, medical specialty, practitioner, consultation type. These mapping classes help encapsulate typing and mapping between enumerators (numbers) and external representations (strings, or enumerated values as they may appear in a UI, which are completely unrelated to internal enumerators). These classes help de-couple the application cluster from the repository cluster.

Aside from the four native clusters described above, MHDCS re-uses the **EiffelStore** cluster for database interfacing and the **EiffelTime** cluster for date and time manipulation. Use of these clusters is described in greater detail below.

DESIGN RATIONALE

This section briefly describes the design principles for each cluster in the MHDCS system.

The Application and Panel clusters

This cluster is modeled after the multi-panel interactive system, described in B. Meyer "Object Oriented Software Construction 2nd Ed", Prentice Hall, 1997, chapter 20 pp. 675-694. The Application maintains both an array of instantiated user interface panels, as well as a two-dimensional array of state transitions which can be navigated based on user selection.

Similarly, the Panel cluster contains class definitions of user interface panels. In this cluster there is a single deferred class named **INTERFACE** from which all user panel classes derive. Each panel is considered a "state" in the state transition model. The application class requires knowledge of the state transitions that should be made based on user selections from any given panel, which is encoded into the two-dimensional state transition array, indexed by user selection. After each transition, the application invokes the **process_panel** feature of the panel object referenced by the new target state. This process continues until the target state is state 0, which by convention requests application exit.

The **PANEL** cluster contains a deferred class **INTERFACE**, as well as five implemented panel classes. Each panel class represents a UI for a specific encapsulation of data. These classes include:

- **OVERVIEW_PANEL**, A help panel providing textual information about the nature of the MHDCS system.
- **HOME_PANEL,** The initial UI panel, providing basic navigation through the system
- **QUERY_PANEL**, The UI panel for requesting database queries of medical consultations, diagnosis, patient and patient relatives medical histories.
- **LIST_USERS_PANEL**, The UI panel for listing known OHIP users.
- **NEW_EVENT_PANEL**. The UI panel for entering new data, and possibly new OHIP users, into the system.

The Database Repositories cluster

This cluster is structured around a high level class named **PATIENT_DATA_REPOSITORY**, and several utility classes which it uses. This cluster, and in particular the **PATIENT_DATA_REPOSITORY** class, are the primary interface for the rest of the system into database operations. This cluster is built on top of **EiffelStore**. The cluster's class set includes:

- **PATIENT_DATA_REPOSITORY**, The primary interface for database operations in the system. This class is a client to the EiffelStore, EiffelTime and Enumated_maps clusters.

- **OHIP_USER**, a mapping class, used in conjunction with the EiffelStore DB_REPOSITORY class. This class follows the EiffelStore "double-matching rule" for the relational table OHIP_USERS.

- **MEDICAL_EVENT**, a mapping class, used in conjunction with the EiffelStore DB_REPOSITORY class. This class follows the EiffelStore "double-matching rule" for the relational table MEDICAL_EVENTS. This class can be used to hold data representing a single record in the repository.

- **MEDICAL_DATABASE**, a class encapsulating database level data and operations. This class operates on the database as a whole, and not on individual tables. It is used by the **PATIENT_DATA_REPOSITORY** class for performing login, database connection, disconnect, logoff, session monitoring for success or failure, and other session based operations.

The Enumerated Maps cluster

The need for this cluster grows from the single choice principle: "Whenever a software system must support a set of alternatives, one and only one module in the system should know their exhaustive list." In the MHDCS project there are several enumerated type which must maintains a persistent enumeration throughout the system since they can be stored inside the relational database. Examples include: types of medical specialties, types of patient consultations, etc. The single choice principle leads to a design requirement to decouple the interface representations and persistent storage representations of data types. Had these representations not been de-coupled, then an exhaustive list of choices would need to be maintained across clusters. The single choice principle requires a central module for these logical mappings. As a result, this cluster was designed to help reduce the inter-dependancy of the Panel cluster and the Database Repository cluster, and provide modular ADTs for mapping enumerated types. The cluster includes classes for mapping types of practitioners, gender, medical consultations, and fields of medical specialty.

Auxilliary clusters, EiffelStore and EiffelTime, reused from Eiffel libraries

MHDCS expliots the **EiffelStore** and **EiffelTime** clusters provided by interactive Software Engineering Inc, which are available as part of the Eiffel installable software that ships with Eiffel v4.3023, used in this system.

EIFFELSTORE

> "EiffelStore is the principal interface between Eiffel and Database Management Systems (DBMS), relational or object-oriented. It enables organizations to combine the power of Eiffel object-oriented development with the need to access existing databases and use them to store objects. The EiffelStore Application Programming Interface (API) is DBMS-independent, so that you can use the same source code with a wide range of actual database products."
>
> **(From the ISE Web page at** http://www.tools.com/products/store.html**)**

In the MHDCS system, interfaces to the **EiffelStore** classes in encapsulated in the **DATABASE_REPOSITORIES** cluster.

EIFFELTIME

The **EiffelTime** cluster is used as a mechanism for performing date and time operation. Standard operations used in the MHDCS project include time-stamping medical consultations, calculation of current date/time, determining the current age of a patient, and mapping between ODBC time/date/timestamp values and **EiffelTime DATE_TIME** classes.

DATABASE DESIGN

ODBC as interface to relational database management systems

Open Database Connectivity (ODBC) is a widely accepted application programming interface (API) for database access. It is based on the Call-Level Interface (CLI) specifications from X/Open and ISO/IEC for database APIs and uses Structured Query Language (SQL) as its database access language. This interface standard is supported by a large set of database vendors, particularly those supporting Windows/NT platforms. Unlike SQL, ODBC is API driven, and generally more conducive to database application development.

Developing MHDCS using an ODBC interface ensures smooth support for a variety of database management systems, including most of the world's leading vendors, such as Oracle, DB2, Sybase, and Microsoft SQLServer.

The design choice to use ODBC was also motivated by a development limitation. In particular, **EiffelStore** provides interface support for only three database server types. These include Oracle, Sybase, and ODBC. However, the development environment available requires the use of DB2 Universal Database. Therefore, in order to use DB2 as a DBMS for the MHDCS system, it was necessary to use the ODBC interface provided by **EiffelStore**.

Database schema

The database is designed using two relational tables, and associated table indexes for fast searching. One table hold sthe complete of all known OHIP users, and their associated meta-data, while the other table contains records for medical events (or medical consultations), which are entered in the database by medical practitioners.

The table and index definitions, are defined as follows (described below in SQL data definition language syntax):

```
table "medical_events" ("patient_ohip_num"    INTEGER NOT NULL,
                        "complaint"           varchar(200),
                        "medical_specialty"   INTEGER,
                        "practitioner_name"   varchar(100),
                        "practitioner_type"   INTEGER,
                        "absolute_date"       TIMESTAMP NOT NULL,
                        "hospital_name"       varchar(70),
                        "final_diagnosis"     varchar(200),
                        "comment"             varchar(200),
                        "event_type"          INTEGER NOT NULL);

index "medevent_inx1" on "medical_events" ( "patient_ohip_num" ASC, "absolute_date" ASC );

table "ohip_users" ("given_names"            varchar(51) NOT NULL,
                    "surname"                varchar(35) NOT NULL,
                    "patient_ohip_num"       INTEGER NOT NULL,
                    "mom_ohip_num"           INTEGER,
                    "dad_ohip_num"           INTEGER,
                    "date_of_birth"          DATE NOT NULL,
                    "gender"                 INTEGER NOT NULL);
```

```
create unique index "ohipinx1" on "ohip_users" ( "patient_ohip_num" ASC );
create index "ohipinx2" on "ohip_users" ( "surname", "given_names" ASC );
```

It is a fundamental requirement of the MHDCS system that these tables exist with precisely these definitions and field names. Upon starting the MHDCS software will examine the database for the existence and conformance of these tables. The system will not proceed without asserting the existence and conformity of these tables.

The storage layout and recovery scheme of the database is independent of the software system, and is therefore assumed to be a design choice of the Database System Administrator (for examples, choice of logging technique, number of disks, assignment of tables and indexes to various storage containers, etc). Similarly, database tuning parameters, such as sort heap size, bufferpool, etc, are all independent of this design, and fall under the purview of the Database System Administrator.

Database Name

The MHDCS software is intentionally designed to be connectable to any ODBC compliant database. As a result, the database name is an input parameter provided by the user upon startup. This enables the user to run the system against multiple data warehouses or data marts, as required.

DATABASE SOFTWARE & INITIAL DATABASE POPULATION

The prototype is designed to operate on top of commercially available database software. For broadest possible DBMS support with **EiffelStore**, database interaction (for event insert, query and display) is being designed using the **EiffelStore** ODBC handle. All database transactions (query, insert, update) have been designed to comply with ODBC level 1 interfaces for maximum compatibility with other database vendors.

For initial data population of the system (bulk insertion of data, prior to the system going online) native DBMS utilities are assumed, as per the project specification (i.e. Import, Load etc).

DATA INTEGRITY CONSIDERATIONS

The prototype design is demonstrates data integrity characteristics by detection partial page I/O, and ensuring crash recovery capabilities through database restart or backup/restore and rollforward recovery. By design, these capabilities are assumed provided by the underlying database software upon which the prototype is constructed.

USER MANUAL

ABOUT THIS MANUAL

The MHDCS user’s guide is a comprehensive guide that contains all of the procedures you need to work with MHDCS. To help you learn and use MHDCS efficiently, this manual is organized by task, beginning with the most common MHDCS features.

The MHDCS software has been designed for ease of use, and if you are an experienced computer user, you may find many of its features self-explanatory. If you’d like to get hands-on experience right a way, install and start MHDCS as described in the “Getting started” chapter of this book, and follow the directions on your screen.

SYSTEM REQUIREMENTS

CPU:	Intel 486 DX 133 MHz CPU system or higher.
RAM:	16 MB minimum
Disk:	600KB, plus database storage.
Operating System:	NT 4.0 or higher (either Workstation or Server).
Database management system:	DB2 UDB v6.1 or higher.

SOFTWARE SUPPORT AND TRAINING SERVICES

For service help with MHDCS, or for information about training and consultation, please contact our support group.

MHDCS Systems Support
1150 Eglinton Avenue East
Toronto, Ontario, Canada
M3C 2G4

e-mail:	light@ca.ibm.com
Telephone:	416 448-3665
FAX:	416 448-4414

GETTING STARTED

Database schema, creating a database and the required tables

You may use MHDCS with any DB2 UDB v6.1 (or greater) database, provided it contains the required relational tables, with the required table definitions. The tables required are as follows (described here using SQL definition syntax):

```
table "medical_events" ("patient_ohip_num"    INTEGER NOT NULL,
                        "complaint"           varchar(200),
                        "medical_specialty"   INTEGER,
                        "practitioner_name"   varchar(100),
                        "practitioner_type"   INTEGER,
                        "absolute_date"       TIMESTAMP NOT NULL,
                        "hospital_name"       varchar(70),
                        "final_diagnosis"     varchar(200),
                        "comment"             varchar(200),
                        "event_type"          INTEGER NOT NULL);

table "ohip_users" ("given_names"            varchar(51) NOT NULL,
                    "surname"                varchar(35) NOT NULL,
                    "patient_ohip_num"       INTEGER NOT NULL,
                    "mom_ohip_num"           INTEGER,
                    "dad_ohip_num"           INTEGER,
                    "date_of_birth"          DATE NOT NULL,
                    "gender"                 INTEGER NOT NULL);
```

Table indexes are recommended on both tables for the OHIP number columns, and date columns.

Installing the product

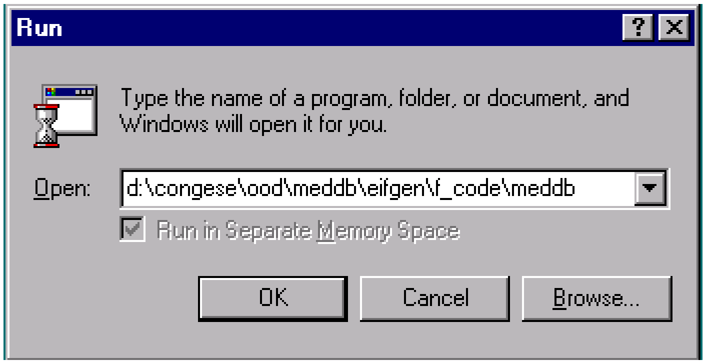
MHDCS is easy to install and use. Copy the mhdcs.exe file to any directory on your computer.

Setting up your environment

MHDCS requires a version of DB2 Universal Database Server to be installed and running prior to beginning the MHDCS program. The DB2 UDB v6.1 is the minimum supported database level.

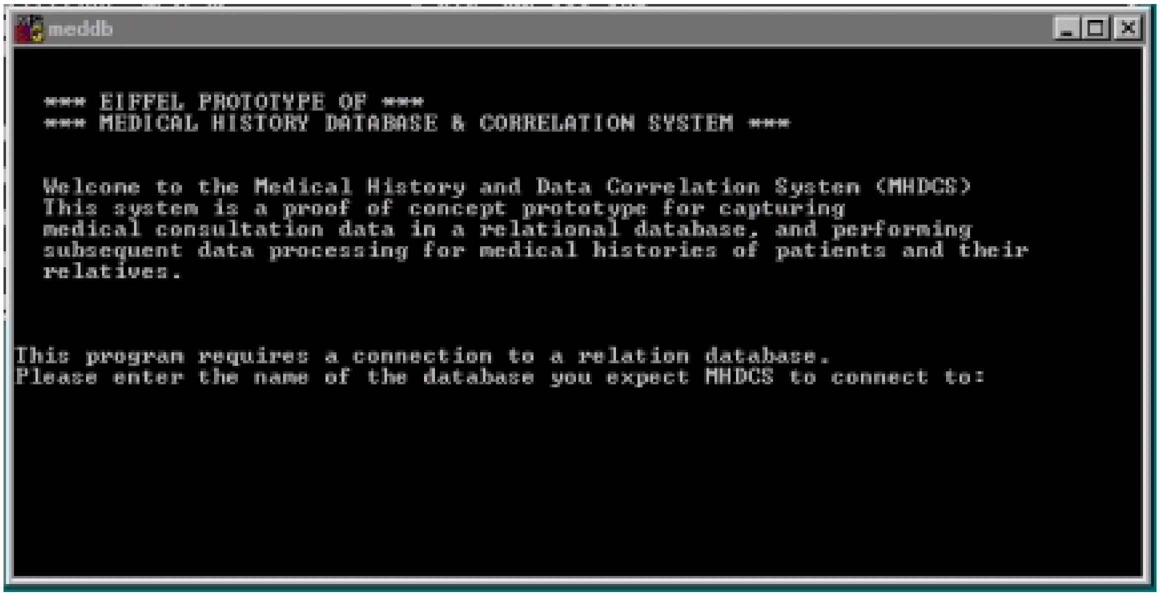
Starting the program

To start MHDCS, proceed to the “start” menu of the Windows action bar. Click on the menu. This will pop up a menu list. From this list click on the menu item. This will pop up a dialog box and prompt you for the name of a program to run. Enter the full path to where you have installed MHDCS, followed by the executable name “mhdcs”, as follows:


Run
Type the name of a program, folder, or document, and Windows will open it for you.
Open:
☑ Run in Separate Memory Space
OK
Cancel
Browse...

```
d:\congese\ood\meddb\eifgen\f_code\meddb
```

This will launch MHDCS, and bring up the initial program screen, with a brief introduction, and a request for a database name.

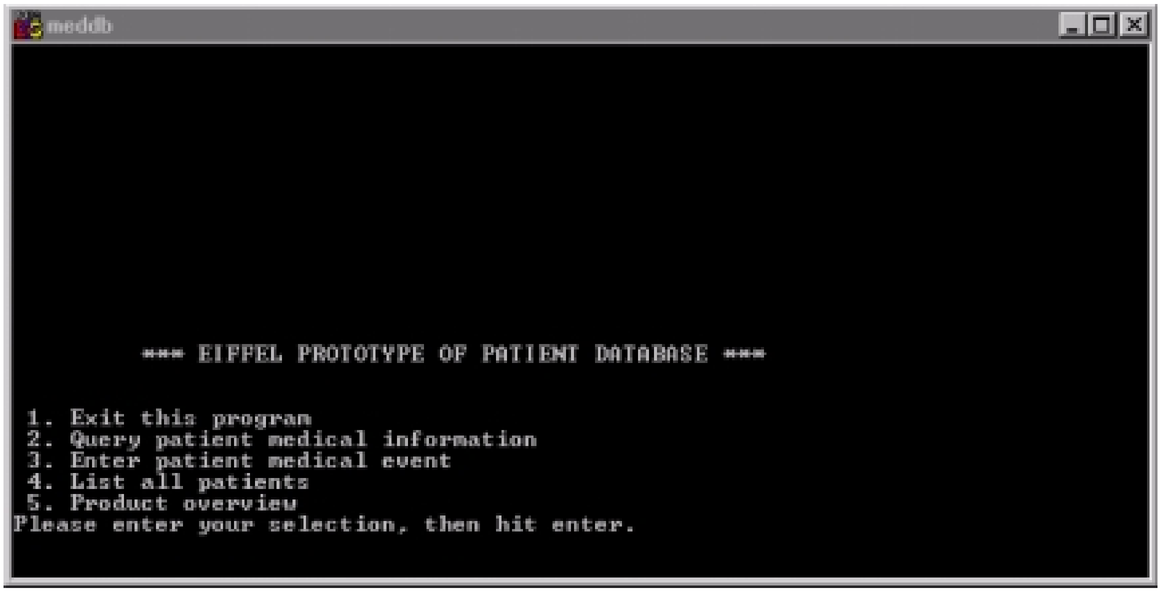


```
  *** EIFFEL PROTOTYPE OF ***
  *** MEDICAL HISTORY DATABASE & CORRELATION SYSTEM ***


  Welcome to the Medical History and Data Correlation System (MHDCS)
  This system is a proof of concept prototype for capturing
  medical consultation data in a relational database, and performing
  subsequent data processing for medical histories of patients and their
  relatives.


This program requires a connection to a relation database.
Please enter the name of the database you expect MHDCS to connect to:
```

Type the name of the database you wish to use, then hit enter. MHDCS will then connect to the database, and bring up the HOME panel. You are then ready to run queries, and insert new data!

NAVIGATING THE MHDCS USER PANELS

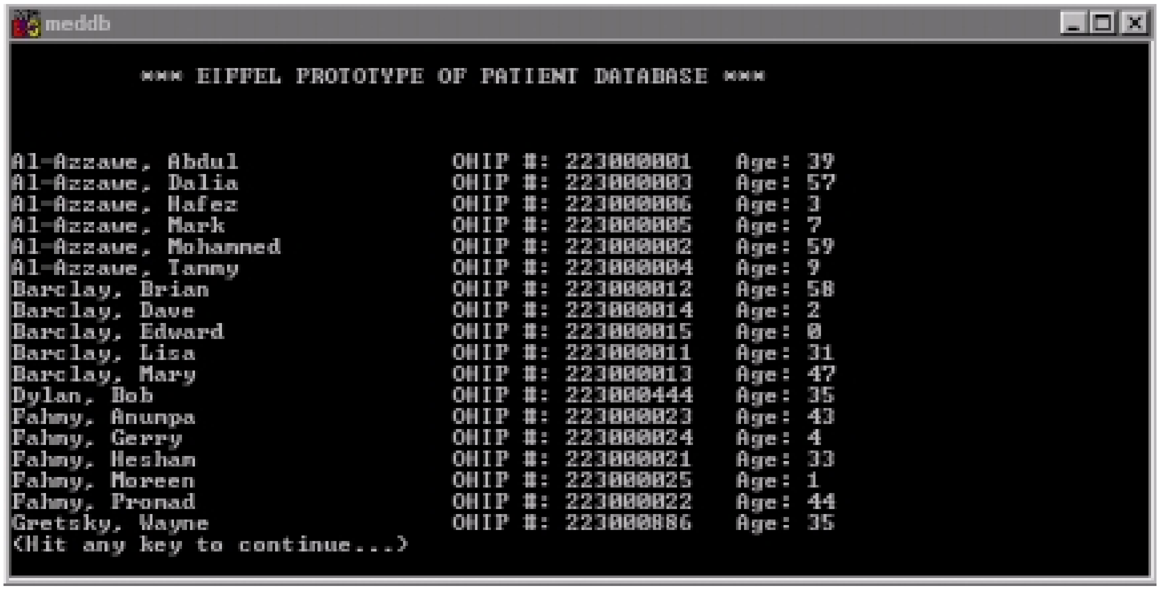
After connecting to the target database, the home panel will be displayed. This panel is the central hub of the MHDCS user interface. From this panel you can make selections that will direct you to the features of the MHDCS software.



```
         *** EIFFEL PROTOTYPE OF PATIENT DATABASE ***


 1. Exit this program
 2. Query patient medical information
 3. Enter patient medical event
 4. List all patients
 5. Product overview
Please enter your selection, then hit enter.
```

LISTING ALL KNOWN OHIP USERS

Selection 4 from the home panel lists all known OHIP uses in the system. A sample of the system output from this panel is shown here. Note that the output is sorted by name, in ascending order, keyed on both last and first names.

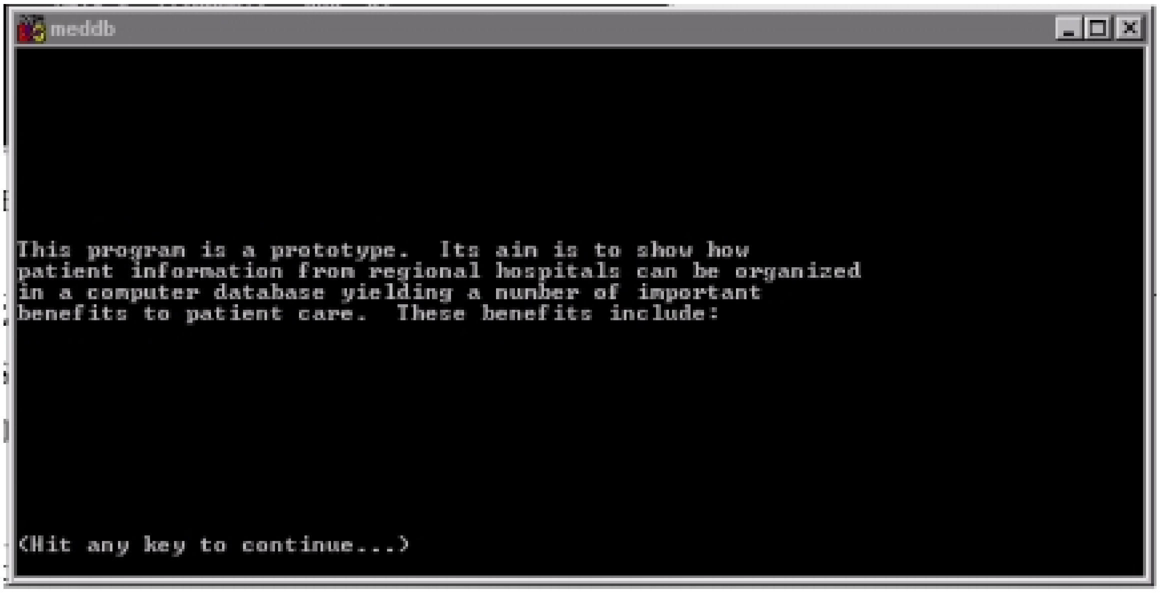


```
         *** EIFFEL PROTOTYPE OF PATIENT DATABASE ***


Al-Azzawe, Abdul               OHIP #: 2230000001   Age: 39
Al-Azzawe, Dalia               OHIP #: 2230000003   Age: 57
Al-Azzawe, Hafez               OHIP #: 2230000006   Age: 3
Al-Azzawe, Mark                OHIP #: 2230000005   Age: 7
Al-Azzawe, Mohammed            OHIP #: 2230000002   Age: 59
Al-Azzawe, Tammy               OHIP #: 2230000004   Age: 9
Barclay, Brian                 OHIP #: 2230000012   Age: 58
Barclay, Dave                  OHIP #: 2230000014   Age: 2
Barclay, Edward                OHIP #: 2230000015   Age: 0
Barclay, Lisa                  OHIP #: 2230000011   Age: 31
Barclay, Mary                  OHIP #: 2230000013   Age: 47
Dylan, Bob                     OHIP #: 2230000444   Age: 35
Fahmy, Anunpa                  OHIP #: 2230000023   Age: 43
Fahmy, Gerry                   OHIP #: 2230000024   Age: 4
Fahmy, Heshan                  OHIP #: 2230000021   Age: 33
Fahmy, Moreen                  OHIP #: 2230000025   Age: 1
Fahmy, Pronad                  OHIP #: 2230000022   Age: 44
Gretsky, Wayne                 OHIP #: 2230000886   Age: 35
<Hit any key to continue...>
```

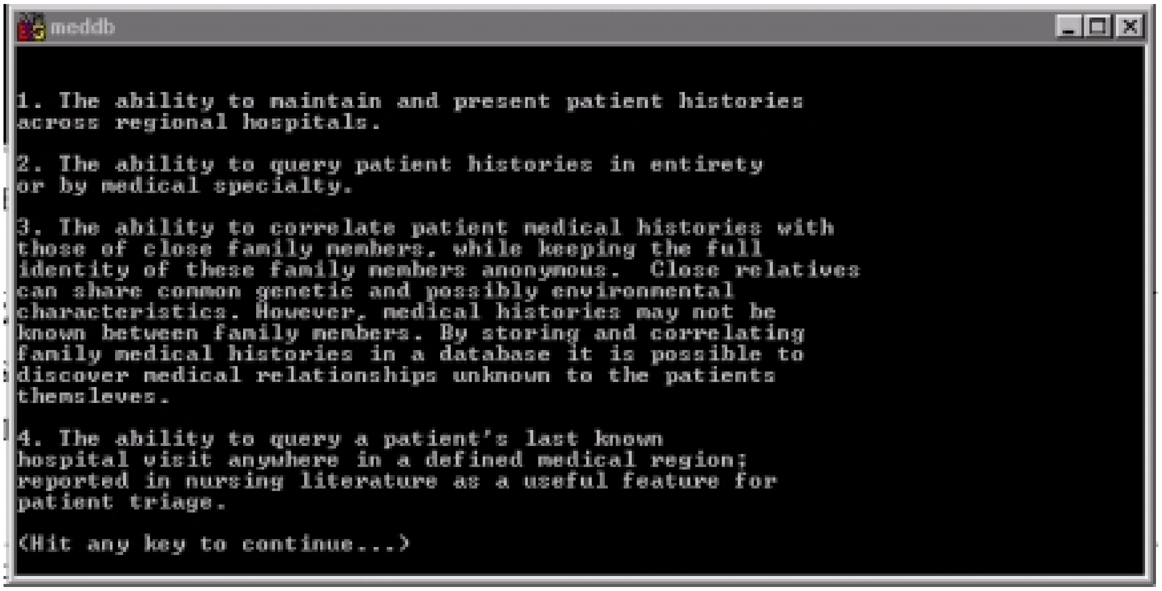
Once the listing of OHIP users is complete, the software will automatically return you to the home panel.

THE PRODUCT OVERVIEW PANEL

Selection 5 from the home panel bring up the product overveiw panbel. This panel provides an easy to read english language overview of the MHDCS software, its goals and features.



```
This program is a prototype.  Its aim is to show how
patient information from regional hospitals can be organized
in a computer database yielding a number of important
benefits to patient care.  These benefits include:






<Hit any key to continue...>
```

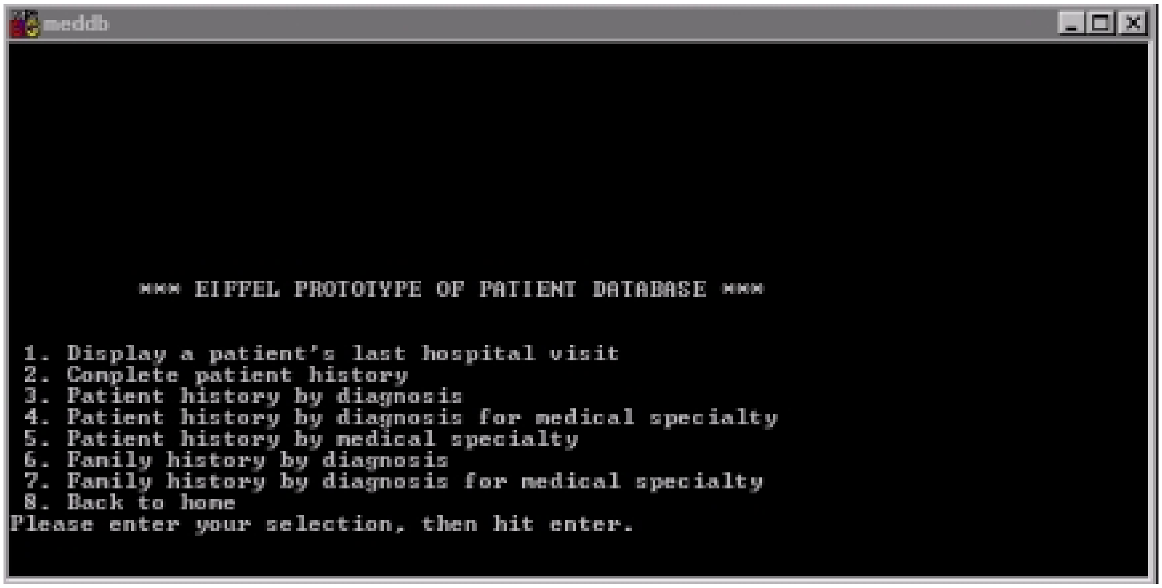


```
1. The ability to maintain and present patient histories
across regional hospitals.

2. The ability to query patient histories in entirety
or by medical specialty.

3. The ability to correlate patient medical histories with
those of close family members, while keeping the full
identity of these family members anonymous.  Close relatives
can share common genetic and possibly environmental
characteristics. However, medical histories may not be
known between family members. By storing and correlating
family medical histories in a database it is possible to
discover medical relationships unknown to the patients
themsleves.

4. The ability to query a patient's last known
hospital visit anywhere in a defined medical region;
reported in nursing literature as a useful feature for
patient triage.

<Hit any key to continue...>
```

Once the text of the product overview is completely displayed, the software returns you directly to the home panel.

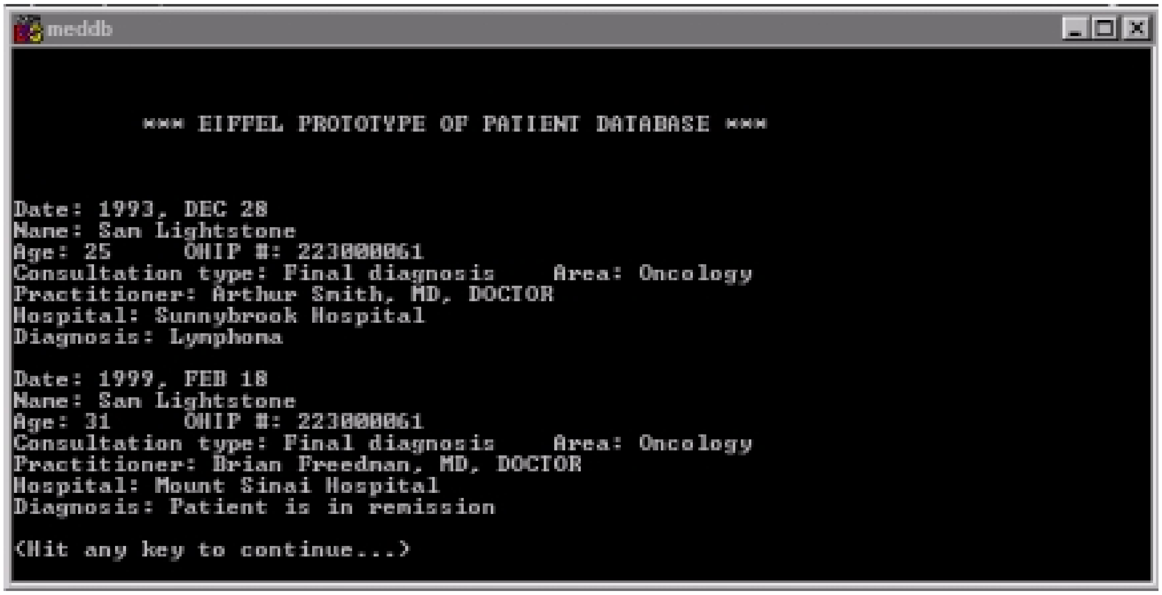
QUERYING PATIENT MEDICAL HISTORY

Option 2 from the home panel will bring up the query panel. This panel provides options for querying patient and patient-relative's medical histories. After each query, you will return to the query panel in case you wish to run subsequent queries. Selecting option 8 will return you to the home panel.



```
        *** EIFFEL PROTOTYPE OF PATIENT DATABASE ***

 1. Display a patient's last hospital visit
 2. Complete patient history
 3. Patient history by diagnosis
 4. Patient history by diagnosis for medical specialty
 5. Patient history by medical specialty
 6. Family history by diagnosis
 7. Family history by diagnosis for medical specialty
 8. Back to home
Please enter your selection, then hit enter.
```

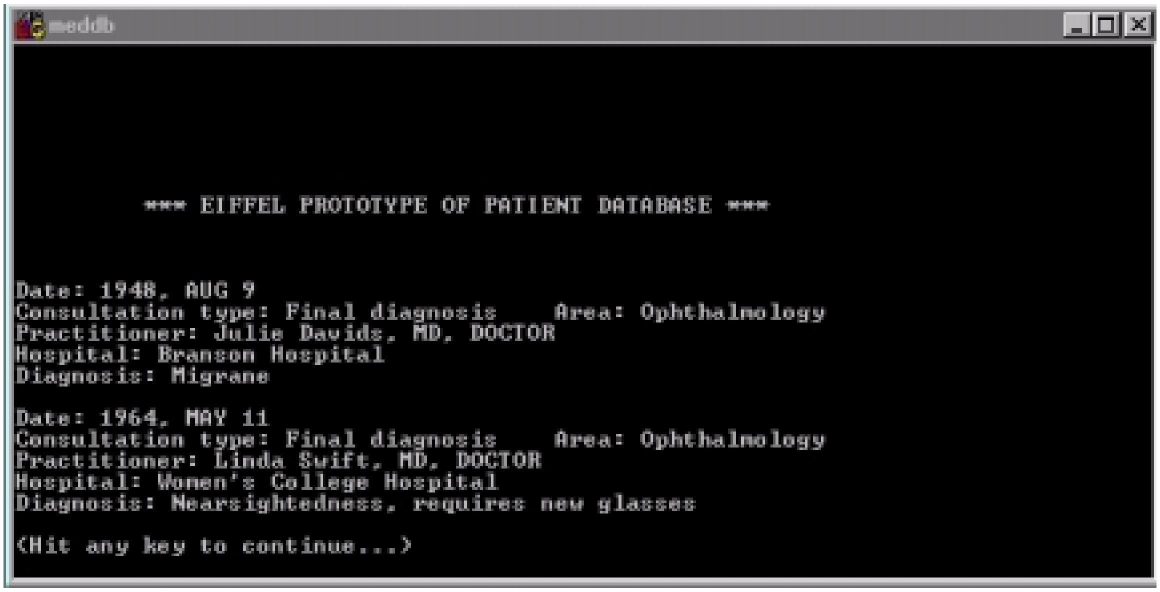
This next slide shows an example of query output for a patient query. Note that the patient's name, age, gender and OHIP number are displayed along with the consultation data, and the attending medical practitioner's identity.



```
         *** EIFFEL PROTOTYPE OF PATIENT DATABASE ***


Date: 1993, DEC 28
Name: Sam Lightstone
Age: 25      OHIP #: 223000061
Consultation type: Final diagnosis    Area: Oncology
Practitioner: Arthur Smith, MD, DOCTOR
Hospital: Sunnybrook Hospital
Diagnosis: Lymphoma

Date: 1999, FEB 18
Name: Sam Lightstone
Age: 31      OHIP #: 223000061
Consultation type: Final diagnosis    Area: Oncology
Practitioner: Brian Freedman, MD, DOCTOR
Hospital: Mount Sinai Hospital
Diagnosis: Patient is in remission

<Hit any key to continue...>
```

QUERYING PATIENT HISTORIES FOR CLOSE RELATIVES

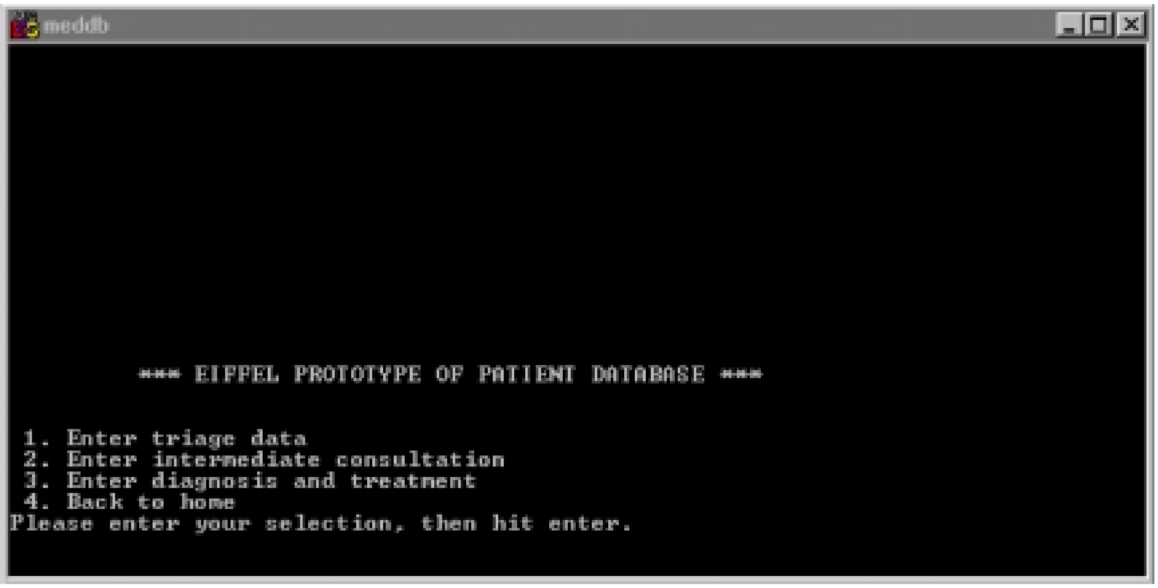
Selection 2 from the home panel brings up the query panel, where you may choose to query patient information, or information about a patients relatives. Note that the system considers the names, ages, and OHIP numbers of patient relatives to be confidential, and will not display these to you when displaying formatted query results. The following diagram is a sample of the query output for diagnosis information of family members. Observe that the patient name, age, gender, and OHIP number are not displayed.



```
         *** EIFFEL PROTOTYPE OF PATIENT DATABASE ***

Date: 1948, AUG 9
Consultation type: Final diagnosis    Area: Ophthalmology
Practitioner: Julie Davids, MD, DOCTOR
Hospital: Branson Hospital
Diagnosis: Migrane

Date: 1964, MAY 11
Consultation type: Final diagnosis    Area: Ophthalmology
Practitioner: Linda Swift, MD, DOCTOR
Hospital: Women's College Hospital
Diagnosis: Nearsightedness, requires new glasses

<Hit any key to continue...>
```

ENTERING NEW CONSULTATION DATA

Option 3 from the home panel will bring up a panel that allows you to enter new consultation data. This includes data for each of the three supported consultation types: triage, intermediate consultation, and diagnosis.

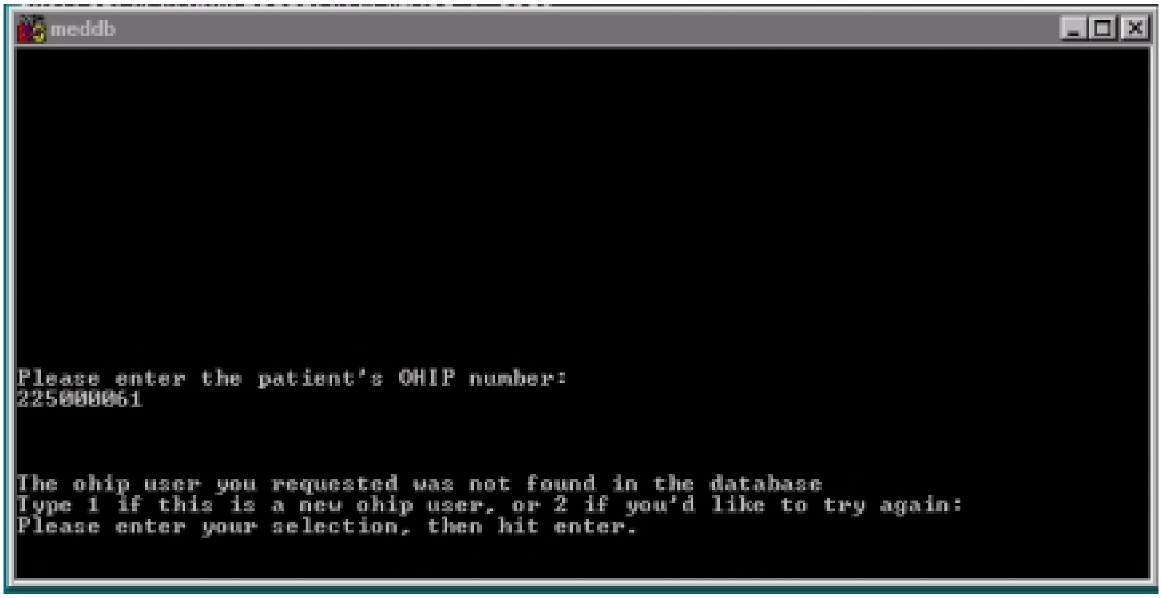


```
         *** EIFFEL PROTOTYPE OF PATIENT DATABASE ***

 1. Enter triage data
 2. Enter intermediate consultation
 3. Enter diagnosis and treatment
 4. Back to home
Please enter your selection, then hit enter.
```

After entering the new consultation data you will return back to this same panel, in case you wish to enter data for another patient, or anther consultation for the same patient. Option 4 takes you back to the home panel.

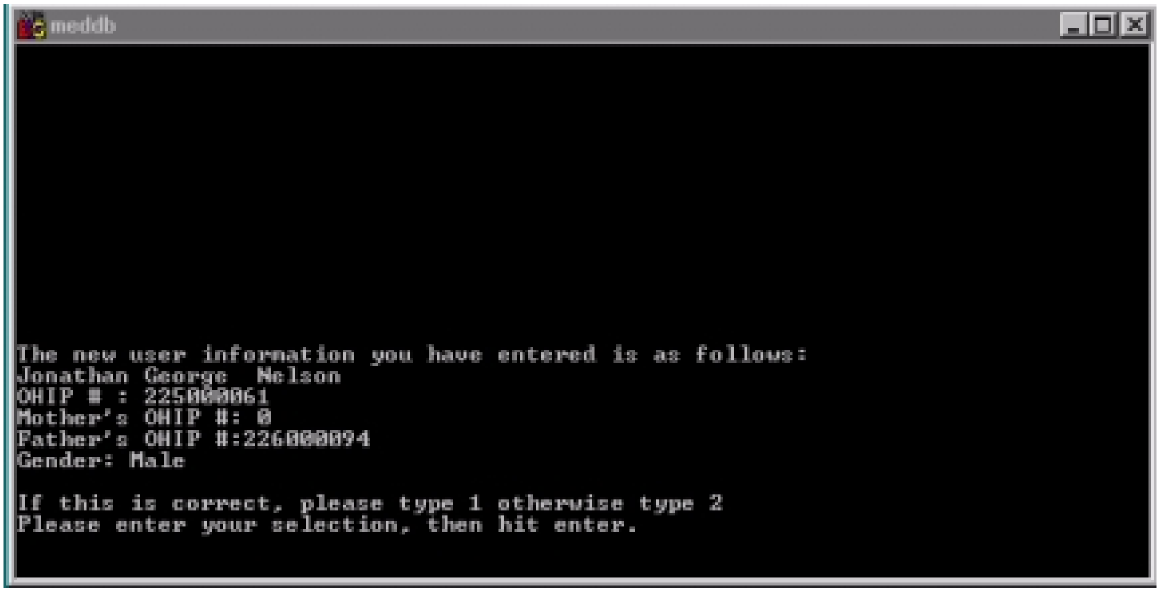
ENTERING A NEW OHIP USER TO THE SYSTEM

A user of MHDCS need not be aware that the consultation data they wish to enter is for a new user. To enter data for a new user in the system proceed as though you were entering consultation for a preexisting user in the database. As described above, select option 3 from the home panel, which takes you to the consultation entry panel. You will be prompted for the patient’s OHIp number, and MHDCS will automatically query the system to determine if this user is currently known in the database. If the OHIP number is not currently found in the system , the application will ask you whether this is a new user, or a typo, as shown below:



```
Please enter the patient's OHIP number:
225000061


The ohip user you requested was not found in the database
Type 1 if this is a new ohip user, or 2 if you'd like to try again:
Please enter your selection, then hit enter.
```

If this is a new user, you’ll be prompted for all the required attributes, such as name, date of birth, gender, and OHIP numbers of the patient’s parents is known. After entering this data, this application will display the data you have entered and ask you for confirmation of its correctness.



```
The new user information you have entered is as follows:
Jonathan George  Nelson
OHIP # : 225000061
Mother's OHIP #: 0
Father's OHIP #:226000094
Gender: Male

If this is correct, please type 1 otherwise type 2
Please enter your selection, then hit enter.
```

PARALLEL PROCESSING POWER

Currently, MHDCS runs on top of IBM’s popular DB2 Universal Database Database Management System, v6.1. This database software provides the MHDCS software you are running with super runtime support for parallel processing through Symmetric Multiprocessors, or Massively Parallel Processors. As well, you may configure your database to perform extensive parallel I/O using multiple disks. Please refer to your database management system’s user manuals for explicit set-up and tuning instructions.

CRASH RECOVERY AND FAULT TOLERANCE

The MHDCS software is designed to perform all of its persistent data operations through the DB2 database management system. Note that all transactions operations to the database, performed by MHDCS for INSERT operations, are committed immediately. As such, all system fault recovery is performed by the underlying DBMS. MHDCS is therefore fully crash recoverable. For additional system failure security, consider taking database backups at regular intervals using the DB2 Backup facility.

OPERATING LIMITS

Storage capacity for MHDCS is currently limited by the table capacity of the underlying database management system.

The current version of this software is a prototype, and runs exclusively on the database server, under the ID of the database administrator. No additional level of user authentication or security is provided.

FUNCTIONAL VERIFICATION TEST PLAN

TEST METHODOLOGY

This test plan assumes the reader and tester has a comprehensive knowledge of the MHDCS User’s Guide. The purpose of these tests are to test the MHDCS system against the system specifications.

The plan coverage is divided into two test categories:

1. Functional Correctness tests. These tests verify the correct behaviour of the system functions against the requirements defined in the system functional specification.

2. Robustness tests. These tests verify the behaviour of the system under abnormal runtime situations, such as system abend, invalid user input etc.

TEST SYSTEM SPECIFICATION

The test system is an Intel P200.

Installed software includes:

1. NT 4.0 Workstation.

2. DB2 Universal Database Version 6.1

3. Medical History Database and Correlation System, version 1.0

4. The test database is a default DB2 database, created using System Manages Space containers. All application tables are created in “USERSPACE1”, according to the database design requirements outlined in the MHDCS User Manual.

5. The test database is preloaded with 58 OHIP patients, and 5.5 thousand consultation records. This data was generated using a simulation program, and is pseudo randomized. This consultationn data includes medical consultations for each of triage, intermediate consultation, and diagnosis. Note that the test data does not generate medical consultation data for children under 5 years of age. The test system will not generate obstetrics data for any men, or for women under 18. The sample data generated by this test system includes:

   - 40 medical practitioners,
   - 7 hospitals,
   - 6 medical specialities (Emergency, Obstetrics, Ophthalmology, Oncology, Psychiatry, Cardiology)
   - 20 practitioner comments
   - 10 medical conditions in each medical speciality
   - 10 patient complaints in each medical speciality
   - 10 diagnosis statements in each medical speciality

FUNCTIONAL CORRECTNESS TESTS

Test case identifier	FCT1
Test objectives	Home panel, test exit
Test operations	Got home panel. Select program exit
Expected outcome	MHDCS application exists normally.

Test case identifier	FCT2
Test objectives	Home panel, test transition to overview panel
Test operations	Proceed to home panel. Select program overview.
Expected outcome	Overview is clearly displayed in three screens, followed by immediate return to home panel.

Test case identifier	FCT3
Test objectives	Home panel, test transition to list all patients
Test operations	Proceed to home panel. Select list all OHIP users.
Expected outcome	All OHIP users are listed in alphabetic order, sorted first by last name, then by first. OHIP # and age and gender should also be displayed. Followed by transition back to home panel

Test case identifier	FCT4
Test objectives	Home panel, test transition to query panel
Test operations	Proceed to home panel. Sleect patient query.
Expected outcome	The patient query panel should pop up, and prompt the user with several new query options.

Test case identifier	FCT5
Test objectives	Home panel test transition to new event panel
Test operations	From the home panel select 3
Expected outcome	Query panel is displayed

Test case identifier	FCT6
Test objectives	Query panel, list patient's last visit
Test operations	Select patient's last visit from query panel, specify a known OHIP user.
Expected outcome	Patient's last hospital triage record should be displayed.

Test case identifier	FCT7
Test objectives	Query panel, query patient complete medical history

Test operations	**Select patient’s complete medical history from the query panel, specify a known OHIP user.**
Expected outcome	**The consultation records for this patient (triage, intermediate consultation, and diagnosis) should be displayed in chronological order.**

Test case identifier	**FCT8**
Test objectives	**Query panel, query patient history by medical specialty**
Test operations	**Select patient’s history for in a medical speciality from query panel, specify a known OHIP user. Repeat this for each of the 6 supported specialities (ER, oncology, ophthalmology, cardiology, obstetrics, psychiatry).**
Expected outcome	**The consultation records should be displayed in chronological order, exclusively for the medical speciality specified.**

Test case identifier	**FCT9**
Test objectives	**Query panel, query medical history in medical speciality of patient relatives**
Test operations	**Query histor of patient relatives from query panel, specify a known OHIP user.**
Expected outcome	**Consultation records should be displayed for the patient’s relatives. The age, gender, and name of the relatives should not be displayed.**

Test case identifier	**FCT10**
Test objectives	**Query panel, test transition back to home panel**
Test operations	**Select the “back to home” option from the query panel.**
Expected outcome	**The home panel should be re-displayed.**

Test case identifier	**FCT11**
Test objectives	**New event panel, test insert of triage data of patient currently known in the MHDCS system**
Test operations	**From the home panel, select the “Enter new consultation data”. From the new event panel, select enter triage data. When prompted, enter the OHIP number of a known OHIP user in the system. Add consulation data.**
Expected outcome	**The steps above should proceed without error. After insertion of the new data, proceed to the query panel and query the patient’s last visit. The new triage data should be displayed.**

Test case identifier	**FCT12**
Test objectives	**New event panel, test insert of triage data for new OHIP patient (currently not recognised in the MHDCS system)**
Test operations	**From the home panel, select the “Enter new consultation data”. From the new event panel, select enter triage**

	data. When prompted, enter the OHIP number of an OHIP user not currently represented in the database (i.e. not in the "ohip_users" table.
Expected outcome	**The system should indicate that the user is not found in the current OHIP users set, and ask if this is truly a new user. If so, you will be prompted for OHIP user data (such as name, gender, date of birth). Following this data entry the system should display your choices and ask you to confirm they are correct. If so, the new data will be update in "ohip_users" and the data entry will proceed with entry of triage attributes.**

Test case identifier	**FCT13**
Test objectives	**New event panel, test insert of intermediate consultation data**
Test operations	**Repeat steps of FCT11, but choose intermediate consultation, rather than triage, from the new event panel.**
Expected outcome	**After the consultation data is entered, select a complete medical history on this user, and the intermediate consultation should appear as the last consultation in this patient's history. When displayed the consultation record should be clearly marked as being intermediate (rather than triage or diagnosis).**

Test case identifier	**FCT14**
Test objectives	**New event panel, test insert of diagnosis data.**
Test operations	**Repeat steps of FCT11, but choose intermediate consultation, rather than triage, from the new event panel.**
Expected outcome	**After the consultation data is entered, select the patient's medical history by diagnosis. The diagnosis data entered in this test case should appear as the final entry.**

Test case identifier	**FCT15**
Test objectives	**New event panel, test transition back to home panel.**
Test operations	**Enter new triage data for a user, by repeating FCT11. Afterwards, select the "Back to home" option.**
Expected outcome	**The "HOME" panel should appear.**

Test case identifier	**FCT16**
Test objectives	**New event panel, test transition back to home panel.**
Test operations	**Enter new triage data for a new user. Use date of birth with boundary condition. Repeat with several interesting birth dates: Jan 1. May 31. Leap years etc. Afterwards run two queries for each patient: 1. List all users and**

	ensure the new patient appears with the correct age, and 2. Query last patient visit, ensure correct consultation data is returned.
Expected outcome	**Correct display of ages and dates. Correct query of patient data and consultation data.**

ROBUSTNESS TESTS

Test case identifier	**RT1**
Test objectives	**Test handling of missing database**
Test operations	**When the MHDCS application (mhdcs.exe) is started, it will prompt the user for a database name. Specify the name of an unknown database**
Expected outcome	**The application should detect the missing database, and report this to the user through a console message.**

Test case identifier	**RT2**
Test objectives	**Test handling of missing relational tables**
Test operations	**When the MHDCS application (mhdcs.exe) is started, it will prompt the user for a database name. Specify the name of an existing database that does not have the required MHDCS tables.**
Expected outcome	**The application should detect the missing relational tables, and report this to the user through a console message.**

Test case identifier	**RT3**
Test objectives	**Home panel, test user selection out of range**
Test operations	**From the home panel specify a selection that is out of range (e.g. 213)**
Expected outcome	**The application should detect the invalid user selection, notify the user of their mistake via a console message, and prompt the user to retry.**

Test case identifier	**RT4**
Test objectives	**Home panel, test user selection non-numeric**
Test operations	**From the home panel specify a selection that is not a vlaid numeric (e.g. eyt7f)**
Expected outcome	**The application should detect the invalid user selection, notify the user of their mistake via a console message, and prompt the user to retry.**

Test case identifier	**RT5**
Test objectives	**Patient query, test invalid OHIP number (valid OHIP number must be 9 digits, not starting with 0)**
Test operations	**Proceed to the query panel. Query patient’s last hospital visit, and specify an OHIP number for the patient that is less than 9 digits. Repeat this for a number with more than 9 digits. Repeat again with a number that is 9 digits and begins with 0.**
Expected outcome	**The application should detect the invalid user input, notify the user of their mistake via a console message,**

	and prompt the user to retry.

Test case identifier	**RT6**
Test objectives	**Patient query, test valid OHIP number of patient not found in the current MHDCS system**
Test operations	**Proceed to the query panel. Query patient’s last hospital visit, and specify an OHIP number that is known not to exit in the “ohip_users” table.**
Expected outcome	**The system should display no consultation records, and continue processing normally.**

Test case identifier	**RT7**
Test objectives	**Test crash recovery during query**
Test operations	**While a database query is running, terminate the application abnormally (through either ctrl-C, or by exiting the DOS session under which the application is running)**
Expected outcome	**Your should be able to restart the application without incident.**

Test case identifier	**RT8**
Test objectives	**Test crash recovery during update**
Test operations	**While performing a database update (insert of new consultation data) terminate the application abnormally (through either ctrl-C, or by exiting the DOS session under which the application is running). Note: in order to abnormally terminate the system in the middle of an update transaction, you may need to test this through a debugger, or enable software hooks, since the timing opportunity is slight (on the order of millisecond)**
Expected outcome	**Your should be able to restart the application without incident. The database should not have any partial consultation records – guaranteed feature of the DBMS.**

Test case identifier	**RT9**
Test objectives	**Test crash recovery after update**
Test operations	**Repeat RT8, but terminate the system just following the consultation insertion.**
Expected outcome	**Your should be able to restart the application without incident. The database should not have any partial consultation records – guaranteed feature of the DBMS.**

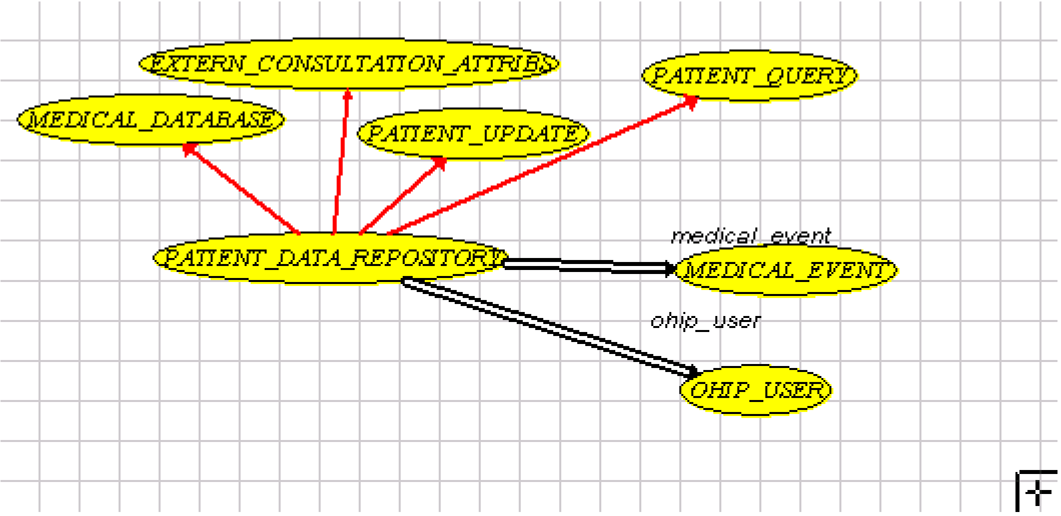
SYSTEM LIMITATIONS AND FUTURE IMPROVEMENTS

DESIGN IMPROVEMENTS

The current design could be improved by further decomposition of the PATIENT_DATA_REPOSITORY class. In particular it is recommended that this class be decomposed into PATIENT_QUERY and PATIENT_UPDATE classes. This would provide better encapsulation of data, and superior modularity within the cluster.

Similarly, the current implementation of PATIENT_DATA_REPOSITORY contains a set of attributes of consultation “bind out” characteristics. These are the consultation characteristics (including both OHIP user details, and consultation data) as they are to be viewed by client classes. The bind-out attributes would be better encapsulated in their own class, such as “EXTERN_CONSULTATION_ATTRIBS”.

The new BON diagram for the cluster would look as follows:



Note that these changes would not incur changes to the client interfaces, since the PATIENT_DATA_REPOSITORY would inherit from classes the features it conmtains in the current implementation/design. This rearchitecture is likely inexpensive, since it predominantly involves moving features from PATIENT_DATA_REPOSITORIES into new classes, rather than the creation of new features.

FUNCTIONAL LIMITATIONS

The current system software is unable to detect a relational table with incorrect definitions. EiffelStore currently has a defect in the “conforms” feature of the DB_REPOSITORY class. This feature is intended to assert the correct specification of a relational table against an Eiffel mapping object. The intent of the designer is to use this feature to test the “ohip_users” and “medical_events” relational table and ensure they contain the expected column types. Due to the defect in Eiffel store, this check is currently not performed.

The EiffelStore cluster appears to have a major problem handling warning conditions from some database operations. Thus, some database operations that complete successfully, but with minor warnings returned from ODBC, may cause unpredictable results in EiffelStore.

FUTURE ENHANCEMENTS

The following enhancements should be considered in future development work and prototyping:

1. Extend the system prototype to include concurrent remote clients, with full authentication and privilege checking.

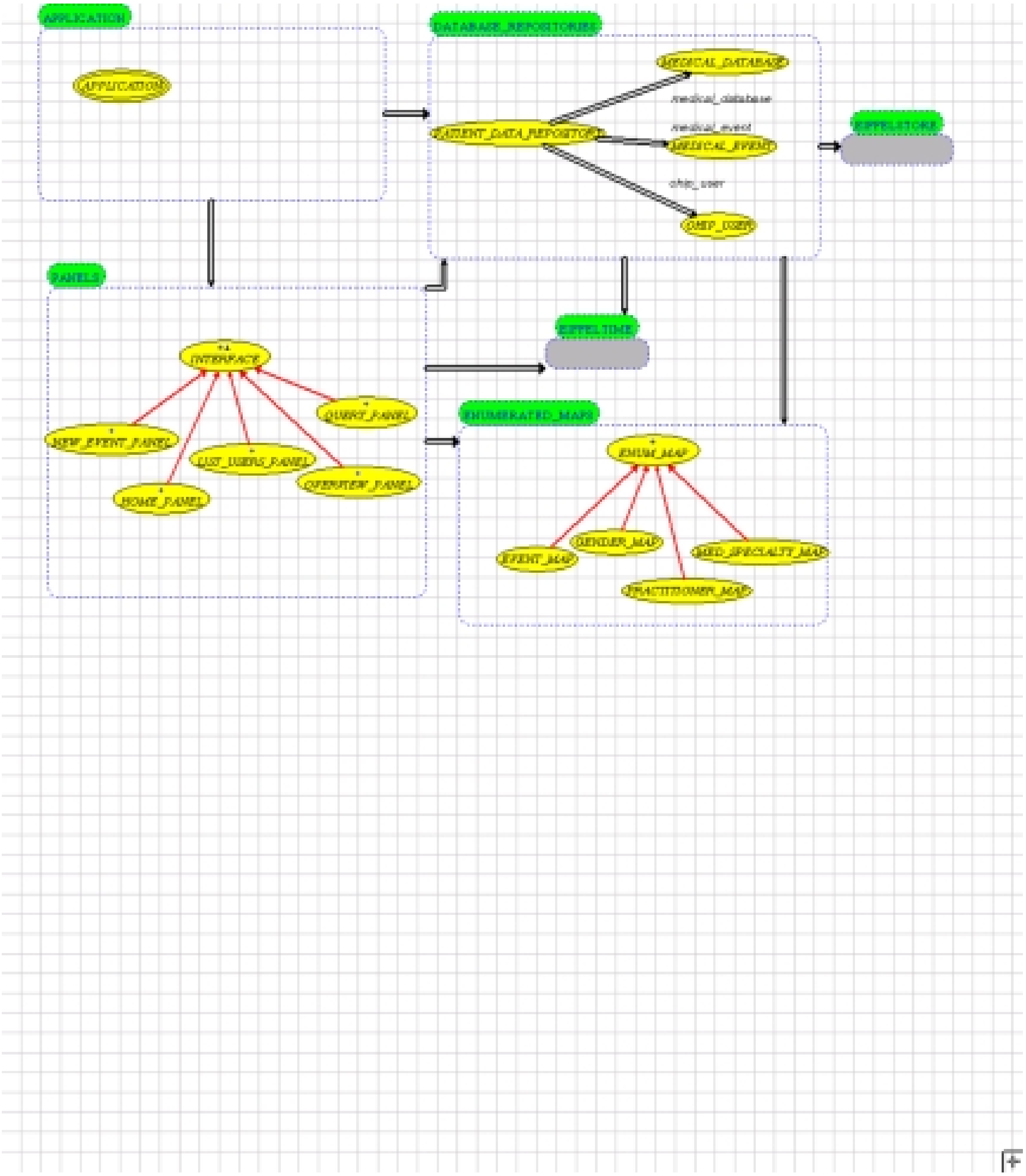
2. Expand the diagnosis consultation data into diagnosis by technical medical terminology and a separate field to proposed treatments.

3. Extend the consultation types (currently triage, intermediate consultation, and diagnosis) to include follow up consultations where treatment progress can be tracked.

4. The current system does not present any patient characteristics when presenting consultation recorsds for family members (i.e. name, gender, age, are masked). Despite the importance of patient privacy, there may be valid reasons to display patient age/gender when presenting this data. For example, if a patient has a heart condition at age 80, this may be far less significant than a patient demonstrating cardiac problem in their teens. Therefore, age and possibly gender may be required in order for user of MHDCS to make sense of the “patient’s relatives” medical history data.

5. Initial prototyping of the system using commercially available data mining software (several such packages are available for DB2 Universal Database today) should be attempted. This will provide an initial examination of the possible data mining value of this system. Initial testing will almost certainly surface recommendations for database definition changes (such as table structures) to enhance the data mining potential of MHDCS.

APPENDIX A: BUSINESS OBJECT NOTATION (BON) SYSTEM DIAGRAM



APPENDIX B: GLOSSARY OF CLASSES

Name of Class	Definition	Belongs to Cluster
APPLICATION	The root class for the medical DB project.	APPLICATION
DATE DATE_TIME DATE_TIME_DURATION	Classes for date and time handling.	EIFFELTIME
DB_CONTROL DB_REPOSITORY DB_RESULT DB_SELECTION DB_SESSION DB_STORE	Classes for database and relational table access.	EIFFELSTORE
ENUM_MAP	Super class for enum classes	ENUMERATED_MAPS
EVENT_MAP	Types of medical events, and their features	ENUMERATED_MAPS
GENDER_MAP	Mapping for gender	ENUMERATED_MAPS
HOME_PANEL	UI panel for the main screen.	PANELS
INTERFACE	Deferred class for standard I/O ops, used by app. Panels.	PANELS
LIST_USERS_PANEL	UI Panel for listing all ohip users in our system.	PANELS
MEDICAL_DATABASE	Database layer, for DB-level (not table) ops	DATABASE_REPOSITORIES
MEDICAL_EVENT	Mapping class, for repositories associated with table MEDICAL_EVENTS	DATABASE_REPOSITORIES
MED_SPECIALTY_MAP	Types of medical specialities we support	ENUMERATED_MAPS
NEW_EVENT_PANEL	Panel for user to enter new patient event data after a patient consultation.	PANELS
OHIP_USER	Mapping class for repositories related to table OHIP_USERS.	DATABASE_REPOSITORIES
OVERVIEW_PANEL	UI panel for displaying product overview	PANELS
PATIENT_DATA_REPOSITORY	Interface to medical repositories and their ops.	DATABASE_REPOSITORIES
PRACTITIONER_MAP	Types of practitioners.	ENUMERATED_MAPS
QUERY_PANEL	UI panel, provides selections for patient queiries.	PANELS

APPENDIX C: SHORT FORM OF SYSTEM CLASSES

APPLICATION

```
indexing

        description: "The root class for the medical DB project."
        author: "Sam Lightstone"
        date: "$Date: $"
        revision: "$Revision: $"

class interface
        APPLICATION

create
        make

feature

        db_problems: BOOLEAN
                        -- DB ok for this app?

        make
                        -- allocate app with n states and m
                        -- possible choices, instantiate transition and
                        -- state arrays, then traverse the state transitions
                        -- until the exit state is traversed.
                ensure
                        post1: application_ui_panels /= void implies state_number = 0

invariant

        inv1: patient_data_repository /= void;
        inv2: not db_problems implies home_panel /= void;
        inv3: not db_problems implies new_event_panel /= void;
        inv4: not db_problems implies query_panel /= void;
        inv5: not db_problems implies all_users_panel /= void;
        inv6: not db_problems implies overview_panel /= void;
        inv7: not db_problems implies panel_transitions /= void;
        inv8: not db_problems implies application_ui_panels /= void;
        inv9: not db_problems implies state_number > 0;
        inv10: not db_problems implies state_number <= max_state;

end -- class APPLICATION
```

ENUM_MAP

```
indexing

        description: "Super class for enum classes"

        author: "Sam Lightstone"

        date: "$Date: $"

        revision: "$Revision: $"


deferred class interface

        ENUM_MAP


feature {ANY}

        clear_map
                        -- unsets the enumerator
                ensure
                        is_set = false

        get_enum: INTEGER
                        -- returns an INTEGER enumerator
                require
                        is_set = true
                ensure
                        Result = enum

        get_formatted_string: STRING
                        -- Returns a human readable string
                        -- representing an enumerated type.
                require
                        is_set = true

        is_set: BOOLEAN
                        -- Has the enumerator been set?

        make
                        -- creation
                ensure
                        post1: is_set = false

        max_enumerator: INTEGER
                        -- the max enumerator supported
                        -- by each class of type ENUM_MAP

invariant

        is_set implies enum > 0;
        is_set implies string /= void;
        is_set implies string.count > 0;

end -- class ENUM_MAP
```

EVENT_MAP

```
Ancestor:

        ENUM_MAP



indexing

        description: "Types of medical events, and their features"
        author: "Sam Lightstone"
        date: "$Date: $"
        revision: "$Revision: $"



class interface

        EVENT_MAP



create

        make



feature {ANY}



        is_diagnosis: BOOLEAN

                        -- return TRUE if diagnosis

                require

                        is_set

                ensure
                        post1: enum = enum_diagnosis implies Result

        is_triage: BOOLEAN
                        -- return TRUE if TRIAGE
                require
                        per1: is_set
                ensure
                        psot1: enum = enum_triage implies Result

        Max_enumerator: INTEGER is 3
                        -- maximum enumerator for this class

        set_diagnosis
                        -- sets the event type to diagnosis (final consult)
                ensure
                        is_set = true;
                        enum = enum_diagnosis

        set_event_from_enum (enum_in: INTEGER)
                        -- sets the map data based on an input enumerator previously
                        -- generated by an onject of this class. This will also
```

```
                                -- set the external string representation for the object.
                        require
                                pre1: enum_in > 0;
                                pre2: enum_in <= max_enumerator
                        ensure
                                post1: enum = enum_in;
                                post2: is_set = true

                set_intermediate
                                -- sets the event type to intermediate (neither triage nor diagnosis)
                        ensure
                                is_set = true;
                                enum = enum_intermediate

                set_triage
                                -- sets the event type to triage
                        ensure
                                is_set = true;
                                enum = enum_triage

        invariant

                invariant_clause: enum <= max_enumerator;

        end -- class EVENT_MAP
```

GENDER_MAP

```
Ancestor:

        ENUM_MAP



indexing

        description: "Mapping for gender"

        author: "Sam Lightstone"

        date: "$Date: $"

        revision: "$Revision: $"



class interface

        GENDER_MAP

create
        make

feature {ANY}

        Max_enumerator: INTEGER is 2
                        -- max enumerator supported in this class

        set_female
                        -- sets the gender type to female
                ensure
                        enum = enum_female;
                        string = female_string;
                        is_set = true

        set_gender_from_enum (enum_in: INTEGER)
                        -- sets the gender type, based on
                        -- input enumerator. This will sanity
                        -- cehck the enum_in, as well as set the
                        -- formatted string representation.
                require
                        enum_in > 0;
                        enum_in <= max_enumerator
                ensure
                        is_set = true

        set_male
                        -- sets the gender type to male
                ensure
                        enum = enum_male;
                        string = male_string;
                        is_set = true

invariant

        invariant_clause: enum <= max_enumerator;

end -- class GENDER_MAP
```

HOME_PANEL

```
Ancestor:

        INTERFACE


indexing

        description: "UI panel for the main screen."

        author: "Sam Lightstone"

        date: "$Date: $"

        revision: "$Revision: $"


class interface

        HOME_PANEL


create

        make


feature


        make (repository_in: PATIENT_DATA_REPOSITORY)
                        -- Creation routine

        process_panel
                        -- Routine to run the panel
                ensure then
                        valid_panel_selection (panel_selection)

end -- class HOME_PANEL
```

INTERFACE

```
indexing

        description: "Deferred class for standard I/O ops, used by app. panels."
        author: "Sam Lightstone"
        date: "$Date: $"
        revision: "$Revision: $"

deferred class interface
        INTERFACE

feature {ANY}

        get_selection: INTEGER
                        -- returns the current user selection
                ensure
                        post1: Result = panel_selection

        make_interface (repository_in: PATIENT_DATA_REPOSITORY)
                        -- the generic make routine useful to all INTERFACE objects
                require
                        repository_in /= void
                ensure
                        post1: patient_data_repository = repository_in;
                        post2: selection = 0;
                        post3: panel_selection = 0

        process_panel
                        -- display a new UI panel

invariant

        inv1: patient_data_repository /= void;
        inv2: selection >= 0;

end -- class INTERFACE
```

LIST_USERS_PANEL

```
Ancestor:

        INTERFACE



indexing

        description: "UI Panel for listing all ohip users in our system."
        author: "Sam Lightstone"
        date: "$Date: $"
        revision: "$Revision: $"

class interface
        LIST_USERS_PANEL

create
        make

feature -- Initialization

        make (repository_in: PATIENT_DATA_REPOSITORY)
                        -- Creation routine

        process_panel
                        -- Routine to run the panel. displayus all known
                        -- OHIP users in the system, sorted by last then first name.
                ensure then
                        post1: patient_data_repository.db_query_in_prog = false;
                        post2: valid_panel_selection (panel_selection)

end -- class LIST_USERS_PANEL
```

MED_SPECIALTY_MAP

```
Ancestor:

        ENUM_MAP



indexing

        description: "Types of medical specialties we support"

        author: "Sam Lightstone"

        date: "$Date: $"

        revision: "$Revision: $"

class interface
        MED_SPECIALTY_MAP

create
        make

feature {ANY}

        Max_enumerator: INTEGER is 6
                        -- the max enumerator supported in the class

        set_cardiology
                        -- sets medical specialty to cardiology
                ensure
                        post1: is_set = true;
                        post2: enum = enum_cardiology

        set_obstetrics
                        -- sets medical specialty to obstetrics
                ensure
                        post1: is_set = true;
                        post2: enum = enum_obstetrics

        set_oncology
                        -- sets medical specialty to oncology
                ensure
                        post1: is_set = true;
                        post2: enum = enum_oncology

        set_ophthalmology
                        -- sets medical specialty to ophthalmology
                ensure
                        post1: is_set = true;
                        post2: enum = enum_ophthalmology

        set_psychiatry
                        -- sets medical specialty to psychiatry
                ensure
                        post1: is_set = true;
                        post2: enum = enum_psychiatry

        set_specialty_from_enum (enum_in: INTEGER)
                        -- sets medical specialty from an enum value. The
                        -- enum value passed should have been previously
                        -- obtained from features of this class. This will
                        -- also set the external string representation.
                require
                        pre1: enum_in > 0;
                        pre2: enum_in <= max_enumerator
                ensure
                        post1: enum = enum_in;
                        post2: is_set = true

        set_triage
                        -- sets medical specialty to triage
                ensure
                        post1: is_set = true;
                        post2: enum = enum_triage

invariant

        invariant_clause: enum <= max_enumerator;

end -- class MED_SPECIALTY_MAP
```

MEDICAL_DATABASE

```
Ancestor:

        RDB_HANDLE



indexing

        description: "Database layer, for DB-level (not table) ops"
        author: "Sam Lightstone"
        date: "$Date: $"
        revision: "$Revision: $"

class interface
        MEDICAL_DATABASE

create
        make

feature

        db_cleanup
                        -- normal garbage collection isn't quite enough in this application.
                        -- If we are connect to a database, we need to disconnect explicitly.
                require
                        session_control /= void
                ensure
                        not session_control.is_connected

        is_connected: BOOLEAN

        make (db_name_in: STRING)

        session_control: DB_CONTROL

        session_ok: BOOLEAN
                require
                        session_control /= void

invariant

        session_control /= void;

end -- class MEDICAL_DATABASE
```

MEDICAL_EVENT

```
indexing

        description: "Mapping class, for repositories associated with table MEDICAL_EVENTS"

        author: "Sam Lightstone"
        date: "$Date: $"
        revision: "$Revision: $"

class interface
        MEDICAL_EVENT

create
        make

feature

        make

feature {ANY}

        absolute_date: DATE_TIME

        comment: STRING

        complaint: STRING

        event_type: INTEGER

        final_diagnosis: STRING

        hospital_name: STRING

        medical_specialty: INTEGER

        patient_ohip_num: INTEGER

        practitioner_name: STRING

        practitioner_type: INTEGER

        reset_patient_ohip_num
                        -- resets the patient_ohip_num attribute to 0
                ensure
                        patient_ohip_num = 0

        set_attributes (complaint_in, pract_name_in, hospital_name_in, diagnosis_in, comment_in: STRING;
ohip_num_in, specialty_in, pract_type_in, event_type_in: INTEGER; date_time_in: DATE_TIME)
                require
                        pre1: ohip_num_in > 0;
                        pre2: pract_name_in.count > 0;
                        pre3: hospital_name_in.count > 0;
                        pre4: event_type_in >= 0;
                        pre5: date_time_in /= void;
                        pre6: specialty_in > 0;
                        pre7: pract_type_in > 0
                ensure
                        post1: patient_ohip_num = ohip_num_in;
                        post2: complaint /= void;
                        post3: medical_specialty = specialty_in;
                        post4: practitioner_name /= void;
                        post5: practitioner_type = pract_type_in;
```

```
                    post6: absolute_date = date_time_in;
                    post7: hospital_name /= void;
                    post8: final_diagnosis /= void;
                    post9: comment /= void;
                    post10: event_type = event_type_in

invariant

       patient_ohip_num >= 0;
       absolute_date /= void;

end -- class MEDICAL_EVENT
```

NEW_EVENT_PANEL

```
Ancestor:

        INTERFACE



indexing

        description: "  Panel for user to enter new patient events data after a patient consultation."
        author: "Sam Lightstone"
        date: "$Date: $"
        revision: "$Revision: $"

class interface
        NEW_EVENT_PANEL

create
        make

feature -- Initialization

        make (repository_in: PATIENT_DATA_REPOSITORY)
                        -- Creation routine
                ensure then
                        post1: specialty_map /= void;
                        post2: event_map /= void;
                        post3: gender_map /= void;
                        post4: practitioner_map /= void;
                        post5: cur_date_time /= void

invariant

        specialty_map /= void;
        event_map /= void;
        gender_map /= void;
        practitioner_map /= void;

end -- class NEW_EVENT_PANEL
```

OHIP_USER

```
indexing

        description: "Mapping class for repositories related to table OHIP_USERS."

        author: "Sam Lightstone"

        date: "$Date: $"

        revision: "$Revision: $"


class interface
        OHIP_USER

create
        make

feature

        make

feature {ANY}

        dad_ohip_num: INTEGER

        date_of_birth: DATE_TIME

        gender: INTEGER

        given_names: STRING

        mom_ohip_num: INTEGER

        patient_ohip_num: INTEGER

        reset_patient_ohip_num
                        -- resets the patient_ohip_num attribute to 0
                ensure
                        patient_ohip_num = 0

        set_attributes (name_in, lastname_in: STRING; ohip_num_in, mom_num_in, dad_num_in, sex_in:
INTEGER; dob_in: DATE_TIME)
                require
                        pre1: ohip_num_in > 0;
                        pre2: name_in.count > 0;
                        pre3: lastname_in.count > 0;
                        pre4: name_in.count > 0;
                        pre5: dob_in /= void
                ensure
                        post1: given_names /= void;
                        post2: surname /= void;
                        post3: patient_ohip_num = ohip_num_in;
                        post4: mom_ohip_num = mom_num_in;
                        post5: dad_ohip_num = dad_num_in;
                        post6: gender = sex_in;
                        post7: date_of_birth = dob_in

        surname: STRING

invariant
```

```
      surname /= void;
      given_names /= void;
      date_of_birth /= void;
      patient_ohip_num >= 0;

end -- class OHIP_USER
```

OVERVIEW_PANEL

```
Ancestor:

        INTERFACE


indexing

        description: "UI panel for displaying product overview"

        author: "Sam Lightstone"

        date: "$Date: $"

        revision: "$Revision: $"


class interface

        OVERVIEW_PANEL


create
        make

feature -- Initialization

        make (repository_in: PATIENT_DATA_REPOSITORY)
                        -- creation routine

        process_panel
                        -- routine to run the panel
                ensure then
                        valid_panel_selection (panel_selection)

end -- class OVERVIEW_PANEL
```

PATIENT_DATA_REPOSITORY

```
indexing

       description: "Interface to medical repositories and their ops."

       author: "Sam Lightstone"

       date: "$Date: $"

       revision: "$Revision: $"


class interface

       PATIENT_DATA_REPOSITORY


create

       make


feature {ANY}

       absolute_date: DATE_TIME
                     -- Date and time of a medical consultation

       bind_out_last_ohip_user
                     -- bind out the last ohip_user we queried/inserted
                     -- if there was one. Otherwise, just NOP.
              ensure
                     post1: (ohip_user.patient_ohip_num > 0) implies is_valid_ohip_user_data;
                     post2: (ohip_user.patient_ohip_num > 0) implies (cur_patient_ohip_num = 
ohip_user.patient_ohip_num)

       comment: STRING
                     -- Comments from the medical practitioner

       complaint: STRING
                     -- patient's complaint that caused them to visit the hospital

       connected_ok: BOOLEAN
                     -- did we connect tot he database ok?

       dad_ohip_num: INTEGER
                     -- patient's father's OHIP #

       date_of_birth: DATE_TIME
                     -- Patient's date of birth

       db_query_in_prog: BOOLEAN
                     -- Is a query currently in progress?

       event_map: EVENT_MAP
                     -- type of consultation (triage, diagnosis, etc)

       final_diagnosis: STRING
                     -- The final diagnosis from the attending medical practitioner
```

```
gender_map: GENDER_MAP
              -- gender of the patient

given_names: STRING
              -- patient given names

hospital_name: STRING
              -- the name of the hospital where the current medical consultation took place

is_patient_in_ohip_repository (ohip_num: INTEGER): BOOLEAN
              -- determine if this user exists in the ohip
              -- repository or not
       require
              pre1: ohip_num > 0
       ensure
              ohip_user.patient_ohip_num > 0 implies Result = true

is_valid_event (event: MEDICAL_EVENT): BOOLEAN
              -- check if medicalevent data
              -- we've bound out is reasonable
       require
              event /= void

is_valid_ohip_user_data: BOOLEAN
              -- sanity check ohip user data.
       require
              pre1: given_names /= void;
              pre2: surname /= void;
              pre3: gender_map /= void

is_valid_ohip_user_obj (usr: OHIP_USER): BOOLEAN
              -- verify that usr obj reprsents a reasonable
              -- OHIP_USER.
       require
              usr /= void

last_query_had_result: BOOLEAN
              -- any values returned form last query?

make (db_name_in: STRING)
              -- Creation routine. Instantiate the medical repository
              -- and other reference objects.

mom_ohip_num: INTEGER
              -- patient's mother's OHIP #

new_medical_event (event: MEDICAL_EVENT)
              -- Enters a new medical event into the DB.
              -- Requires the patient exist in the ohip_repository
              -- Insert is amazingly easy using Eiffel
              -- object-to-repository mapping!
       require
              pre1: is_patient_in_ohip_repository (cur_patient_ohip_num);
              pre2: event.medical_specialty <= specialty_map.max_enumerator;
              pre3: event.practitioner_name.count > 0;
              pre4: event.practitioner_type <= practitioner_map.max_enumerator;
              pre5: event.absolute_date /= void;
              pre6: event.hospital_name.count > 0;
              pre7: event.event_type <= event_map.max_enumerator

new_patient (new_user: OHIP_USER)
              -- Enters a new patient in the ohip_users respository
              -- Insert is amazingly easy using Eiffel
              -- object-to-repository mapping!
       require
```

```
              pre1: is_valid_ohip_user_obj (new_user)
       ensure
              post1: update_ok implies is_patient_in_ohip_repository (cur_patient_ohip_num);
              post2: cur_patient_ohip_num = new_user.patient_ohip_num;
              post3: ohip_user.patient_ohip_num = new_user.patient_ohip_num

next_medical_event
              -- fetch the next medical_event in a query of
              -- the medical event table.
       require
              pre1: db_query_in_prog
       ensure
              post1: not results_exhausted implies db_query_in_prog;
              post2: not results_exhausted implies is_valid_event (medical_event)

next_ohip_user
              -- fetch the next ohip user in a query of ohip
              -- users table.
       require
              pre1: db_query_in_prog
       ensure
              post1: not results_exhausted implies db_query_in_prog;
              post2: not results_exhausted implies is_valid_ohip_user_obj (ohip_user)

patient_ohip_num: INTEGER
              -- patient OHIP #

practitioner_map: PRACTITIONER_MAP
              -- Type of practitioner (nurse, docutor etc)

practitioner_name: STRING
              -- The practitioner's name (Nurse or Doctor)

query_all_patients
              -- List all patients by name and OHIP number.
       ensure
              post1: query_ok

query_children
              -- Retrieve the set of children who have the current
              -- patient as a parent
       require
              pre1: is_patient_in_ohip_repository (cur_patient_ohip_num)
       ensure
              post1: query_ok

query_ok: BOOLEAN
              -- Did the last query  run ok?

query_patient_complete_history
              --List a complete medical history for a patient.
       require
              pre1: is_patient_in_ohip_repository (cur_patient_ohip_num)
       ensure
              post1: query_ok

query_patient_diagnosis_history
              -- Retrieve a patient's medical history, listing only
              -- the final diagnosis for each problem.
       require
              pre1: is_patient_in_ohip_repository (cur_patient_ohip_num)
       ensure
              post1: query_ok

query_patient_history_by_specialty (specialty_enum: MED_SPECIALTY_MAP; diagnosis_only: BOOLEAN)
              -- For a given medical specifalty, retrieve a patient's
```

```
            -- medical history, listing only the final diagnosis for each problem.
        require
            pre1: is_patient_in_ohip_repository (cur_patient_ohip_num)
        ensure
            query_ok

    query_patient_last_visit
            -- List the last recorded medical event for a patient in the database.
        require
            pre1: is_patient_in_ohip_repository (cur_patient_ohip_num)
        ensure
            post1: query_ok

    query_relatives_diagnosis_history_by_specialty
            -- For a given medical specialty, retrieve the medical histories
            -- listing only the final diagnosis for each problem for apatient's
            -- relatives (i.e. Parents & children)
        require
            pre1: is_patient_in_ohip_repository (cur_patient_ohip_num)
        ensure
            post1: query_ok

    query_siblings
            -- Retrieve the set of siblings for the current patient
        require
            pre1: is_patient_in_ohip_repository (cur_patient_ohip_num)
        ensure
            post1: query_ok

    repository_cleanup
            -- garbage collection may not handle DB
            -- teardown. We should add this explicitly

    results_exhausted: BOOLEAN
            -- Any remaining data to fetch for this query?

    specialty_map: MED_SPECIALTY_MAP
            -- Current medical specialty (obstetrics, oncology etc)

    surname: STRING
            -- patient surname

    tables_conform: BOOLEAN
            -- Does a required table exist?

    update_ok: BOOLEAN
            -- Was the last database INSERT/UPDATE successful?

invariant

    inv1: medical_database /= void;
    inv2: tables_conform implies base_selection /= void;
    inv3: tables_conform implies store /= void;
    inv4: tables_conform implies medical_event /= void;
    inv5: tables_conform implies ohip_user /= void;
    inv6: tables_conform implies gender_map /= void;
    inv7: tables_conform implies event_map /= void;
    inv8: tables_conform implies practitioner_map /= void;
    inv9: tables_conform implies specialty_map /= void;
    inv10: tables_conform implies my_cursor /= void;

end -- class PATIENT_DATA_REPOSITORY
```

PRACTITIONER_MAP

```
Ancestor:

       ENUM_MAP


indexing

       description: "Types of practitioners."

       author: "Sam Lightstone"

       date: "$Date: $"

       revision: "$Revision: $"


class interface

       PRACTITIONER_MAP


create

       make


feature



       Max_enumerator: INTEGER is 2
                     -- max enumerator allowed in this class.

       set_doctor
                     -- sets the prectitioner type to doctor
              ensure
                     enum = enum_doctor;
                     string = doctor_string;
                     is_set = true

       set_nurse
                     -- sets the practitioner type to nurese
              ensure
                     enum = enum_nurse;
                     string = nurse_string;
                     is_set = true

       set_practitioner_from_enum (enum_in: INTEGER)
                     -- sets the enumerated map given an enumerator.
                     -- this will do a sanity check ont he enum,
                     -- set the human readable format string, and mark the
                     -- object as being set ("is_set" = TRUE)
              require
                     enum_in > 0;
                     enum_in <= max_enumerator
               ensure
                     enum = enum_in;
                     is_set = true

 invariant

        invariant_clause: enum <= max_enumerator;

 end -- class PRACTITIONER_MAP
```

QUERY_PANEL

```
Ancestor:

        INTERFACE


indexing

        description: "UI panel, provides selections for patient queiries."

        author: "Sam Lightstone"

        date: "$Date: $"

        revision: "$Revision: $"


class interface

        QUERY_PANEL

create
        make

feature -- Initialization

        make (repository_in: PATIENT_DATA_REPOSITORY)
                        -- Creation routine
                ensure
                        post1: specialty_enum /= void

end -- class QUERY_PANEL
```

APPENDIX D: COMPLETE SYSTEM SOURCE CODE

APPLICATION

```
indexing

        description: "The root class for the medical DB project.";

        author: "Sam Lightstone";

        date: "$Date: $";

        revision: "$Revision: $"


class APPLICATION


create

        make


feature


        make is

                        -- allocate app with n states and m

                        -- possible choices, instantiate transition and
                        -- state arrays, then traverse the state transitions
                        -- until the exit state is traversed.
                local
                        m: INTEGER;
                        db_name: STRING
                do
                        io.putstring (intro_text);
                        db_name := "";
                        io.new_line;
                        io.new_line;
                        io.putstring ("%N%NThis program requires a connection to a relation database.");
                        io.putstring ("%NPlease enter the name of the database you expect MHDCS to connect to:%N ");
                        io.readline;
                        db_name := clone (io.laststring);
                        create patient_data_repository.make (db_name);
                        if not patient_data_repository.connected_ok then
                                db_problems := true;
                                io.putstring ("%NUnable to connect to the specified database.");
                                io.putstring ("%NThe database may not exist, or may be locked by another
application.");

                                io.putstring ("%NPlease try again later.")
                        elseif not patient_data_repository.tables_conform then
                                db_problems := true;
                                io.putstring ("%NThe database tables do not conform the required specification.");
                                io.putstring ("%NPlease contact the database administrator for help.")
                        end;
                        create home_panel.make (patient_data_repository);
                        create new_event_panel.make (patient_data_repository);
                        create query_panel.make (patient_data_repository);
                        create all_users_panel.make (patient_data_repository);
                        create overview_panel.make (patient_data_repository);
                        create panel_transitions.make (max_state, max_panel_choices);
                        create application_ui_panels.make (1, max_state);
                        init_ui_panels;
                        init_panel_transitions;
                        select_initial_state (1);
                        if not db_problems then
                                execute
                        end
                ensure
                        post1: application_ui_panels /= void implies state_number = 0
                end;

        db_problems: BOOLEAN;
                        -- DB ok for this app?

feature {NONE}

        state_number: INTEGER;
                        -- the set of UI panels we will use
                        -- each of these is associated with
                        -- an application state.

        home_panel: HOME_PANEL;
                        -- Home panel UI

        new_event_panel: NEW_EVENT_PANEL;
                        -- New consultation UI

        query_panel: QUERY_PANEL;
                        -- Patient query UI

        all_users_panel: LIST_USERS_PANEL;
                        -- List fo OHIP users UI

        overview_panel: OVERVIEW_PANEL;
                        -- Product overvewi UI

        Max_state: INTEGER is 5;
                        -- max state allowed int he application

        Max_panel_choices: INTEGER is 8;
                        -- max num slections for any panel

        initial: INTEGER;
                        -- number of the initial state

        patient_data_repository: PATIENT_DATA_REPOSITORY;
                        -- highest level class for interface to DB operations

        execute is
                        -- perform a user session
                local
                        cur_panel: INTERFACE
                do
                        from
                                state_number := initial
                        invariant
                                0 <= state_number;
                                state_number <= max_state
                        until
                                state_number = 0
                        loop
                                cur_panel := application_ui_panels.item (state_number);
                                cur_panel.process_panel;
                                state_number := panel_transitions.item (state_number, cur_panel.get_selection)
                        end
                ensure
                        state_number = 0
                end;

        put_state (ref_panel: INTERFACE; state_num: INTEGER) is
                        -- enter state with index sn
                require
                        1 <= state_number;
                        state_num <= max_state
                do
                        application_ui_panels.put (ref_panel, state_num)
                end;

        select_initial_state (initial_state_num: INTEGER) is
                        -- define the state number for the intial
                        -- UI panel.
                require
                        1 <= initial_state_num;
                        initial_state_num <= max_state
                do
                        initial := initial_state_num
                ensure
                        initial = initial_state_num
                end;

        put_transition (source, target, label: INTEGER) is
                        -- enter transition label  from state number
                        -- source to state number target
                require
                        1 <= source;
                        source <= max_state;
                        0 <= target;
                        target <= max_state
                do
                        panel_transitions.put (source, label, target)
                end;

        panel_transitions: ARRAY2 [INTEGER];
                        -- 2 dimensional array of state transitions

        application_ui_panels: ARRAY [INTERFACE];
                        -- array of state objects

        init_panel_transitions is
                        -- initialize the two dimensional array
                        -- of panel state transitions. First element
                        -- is the target state, second element is
                        -- the current state, and the third element
                        -- is the selection index.
                        -- States are as follows:
                        -- State 1: HOME_PANEL
                        -- State 2: QUERY_PANEL
                        -- State 3:    NEW_EVENT_PANEL
                        -- State 4:    LIST_USERS_PANEL
                        -- State 5: OVERVIEW_PANEL
                do
                        panel_transitions.put (0, 1, 1)
                        panel_transitions.put (2, 1, 2)
                        panel_transitions.put (3, 1, 3)
                        panel_transitions.put (4, 1, 4)
                        panel_transitions.put (5, 1, 5)
                        panel_transitions.put (2, 2, 1)
                        panel_transitions.put (2, 2, 2)
                        panel_transitions.put (2, 2, 3)
                        panel_transitions.put (2, 2, 4)
                        panel_transitions.put (2, 2, 5)
                        panel_transitions.put (2, 2, 6)
                        panel_transitions.put (2, 2, 7)
                        panel_transitions.put (1, 2, 8)
                        panel_transitions.put (3, 3, 1)
                        panel_transitions.put (3, 3, 2)
                        panel_transitions.put (3, 3, 3)
                        panel_transitions.put (1, 3, 4)
                        panel_transitions.put (1, 4, 1)
                        panel_transitions.put (1, 5, 1)
                end;

        init_ui_panels is
                        -- this feature intializes the
                        -- application_panel_transition array
                        -- with a references to each panel
                        -- this app may need.
                do
```

```
                        application_ui_panels.enter (home_panel, 1)
                        application_ui_panels.enter (query_panel, 2)
                        application_ui_panels.enter (new_event_panel, 3)
                        application_ui_panels.enter (all_users_panel, 4)
                        application_ui_panels.enter (overview_panel, 5)
                ensure
                        application_ui_panels.item (1) = home_panel;
                        application_ui_panels.item (2) = query_panel;
                        application_ui_panels.item (3) = new_event_panel;
                        application_ui_panels.item (4) = all_users_panel;
                        application_ui_panels.item (5) = overview_panel
                end;

        get_db_name: STRING is
                        -- Prompt user for db name, and return db name string
                local
                        temp_string: STRING
                do
                        io.putstring ("%NPlease enter the name of the database you expect MHDCS to connect to: ");
                        io.readline;
                        temp_string := clone (io.laststring);
                        Result := temp_string
                end;

        start_db_session is
                        -- prompt user for DB name, and instantiate the
                        -- patient_data_repository object.
                local
                        db_name: STRING
                do
                        db_name := "";
                        db_name := get_db_name;
                        create patient_data_repository.make (db_name);
                        if patient_data_repository.connected_ok then
                                io.putstring ("%N%N Unable to connect to the database you specified!");
                                io.putstring ("%NPlease try again later ")
                        end
                end;

        Intro_text: STRING is "%N%N  *** EIFFEL PROTOTYPE OF *** %N  *** MEDICAL HISTORY DATABASE & CORRELATION
SYSTEM *** %N%N %N  Welcome to the Medical History and Data Correlation System (MHDCS)%T%N  This system is a proof
of concept prototype for capturing %N  medical consultation data in a relational database, and performing%N
subsequent data processing for medical histories of patients and their %N  relatives.";

invariant

        inv1: patient_data_repository /= void;
        inv2: not db_problems implies home_panel /= void;
        inv3: not db_problems implies new_event_panel /= void;
        inv4: not db_problems implies query_panel /= void;
        inv5: not db_problems implies all_users_panel /= void;
        inv6: not db_problems implies overview_panel /= void;
        inv7: not db_problems implies panel_transitions /= void;
        inv8: not db_problems implies application_ui_panels /= void;
        inv9: not db_problems implies state_number > 0;
        inv10: not db_problems implies state_number <= max_state;

end -- class APPLICATION
```

ENUM_MAP

```
indexing

        description: "Super class for enum classes";

        author: "Sam Lightstone";

        date: "$Date: $";
        revision: "$Revision: $"

deferred class ENUM_MAP

feature {ANY}

        make is
                        -- creation
                do
                        enum := 0
                        string := ""
                ensure
                        post1: is_set = false
                end;

        is_set: BOOLEAN;
                        -- Has the enumerator been set?

        clear_map is
                        -- unsets the enumerator
                do
                        is_set := false
                        string := void
                        enum := 0
                ensure
                        is_set = false
                end;

        get_enum: INTEGER is
                        -- returns an INTEGER enumerator
                require
                        is_set = true
                do
                        Result := enum
                ensure
                        Result = enum
                end;

        get_formatted_string: STRING is
                        -- Returns a human readable string
                        -- representing an enumerated type.
                require
                        is_set = true
                do
                        Result := string
                end;

        max_enumerator: INTEGER is
                        -- the max enumerator supported
                        -- by each class of type ENUM_MAP
                deferred
                end;

feature {NONE}

        enum: INTEGER;
                        -- The enumerated value

        string: STRING;
                        -- the human readable representation of the enumerated type

invariant
```

```
        is_set implies enum > 0;
        is_set implies string /= void;
        is_set implies string.count > 0;

end -- class ENUM_MAP
```

EVENT_MAP

```
Ancestor:

        ENUM_MAP


indexing

        description: "Types of medical events, and their features";

        author: "Sam Lightstone";

        date: "$Date: $";
        revision: "$Revision: $"

class EVENT_MAP

inherit
        ENUM_MAP

create
        make

feature {ANY}

        set_triage is
                        -- sets the event type to triage
                do
                        enum := enum_triage
                        string := triage_string
                        is_set := true
                ensure
                        is_set = true;
                        enum = enum_triage
                end;

        set_intermediate is
                        -- sets the event type to intermediate (neither triage nor diagnosis)
                do
                        enum := enum_intermediate
                        string := intermediate_string
                        is_set := true
                ensure
                        is_set = true;
                        enum = enum_intermediate
                end;

        set_diagnosis is
                        -- sets the event type to diagnosis (final consult)
                do
                        enum := enum_diagnosis
                        string := diagnosis_string
                        is_set := true
                ensure
                        is_set = true;
                        enum = enum_diagnosis
                end;

        set_event_from_enum (enum_in: INTEGER) is
                        -- sets the map data based on an input enumerator previously
                        -- generated by an onject of this class. This will also
                        -- set the external string representation for the object.
                require
                        pre1: enum_in > 0;
                        pre2: enum_in <= max_enumerator
                do
                        is_set := false
                        if enum_in = enum_triage then
                                set_triage
                        elseif enum_in = enum_intermediate then
                                set_intermediate
                        elseif enum_in = enum_diagnosis then
                                set_diagnosis
                        end
                ensure
                        post1: enum = enum_in;
                        post2: is_set = true
                end;

        is_triage: BOOLEAN is
                        -- return TRUE if TRIAGE
                require
                        per1: is_set
                do
                        if enum = enum_triage then
                                Result := true
                        end
                ensure
                        psot1: enum = enum_triage implies Result
                end;

        is_diagnosis: BOOLEAN is
                        -- return TRUE if diagnosis
                require
                        is_set
                do
                        if enum = enum_diagnosis then
                                Result := true
                        end
                ensure
                        post1: enum = enum_diagnosis implies Result
                end;

        Max_enumerator: INTEGER is 3;
                        -- maximum enumerator for this class

feature {NONE}

        Enum_triage: INTEGER is 1;
                        -- enumerated value for triage

        Enum_intermediate: INTEGER is 2;
                        -- enumerated value for intermediate

        Enum_diagnosis: INTEGER is 3;
                        -- enumerated value for diagnosis

        Triage_string: STRING is "Triage";
                        -- external string representation for triage

        Intermediate_string: STRING is "Consultation";
                        -- external string representation for intermediate consult

        Diagnosis_string: STRING is "Final diagnosis";
                        -- external string representation for final diagnosis

invariant

        invariant_clause: enum <= max_enumerator;

end -- class EVENT_MAP
```

GENDER_MAP

```
Ancestor:

        ENUM_MAP



indexing

        description: "Mapping for gender";
        author: "Sam Lightstone";
        date: "$Date: $";
        revision: "$Revision: $"

class GENDER_MAP

inherit
        ENUM_MAP

create
        make

feature {ANY}

        set_male is
                        -- sets the gender type to male
                do
                        enum := enum_male
                        string := male_string
                        is_set := true
                ensure
                        enum = enum_male;
                        string = male_string;
                        is_set = true
                end;

        set_female is
                        -- sets the gender type to female
                do
                        enum := enum_female
                        string := female_string
                        is_set := true
                ensure
                        enum = enum_female;
                        string = female_string;
                        is_set = true
                end;

        set_gender_from_enum (enum_in: INTEGER) is
                        -- sets the gender type, based on
                        -- input enumerator. This will sanity
                        -- cehck the enum_in, as well as set the
                        -- formatted string representation.
                require
                        enum_in > 0;
                        enum_in <= max_enumerator
                do
                        if enum_in = enum_male then
                                set_male
                        else
                                set_female
                        end
                ensure
                        is_set = true
                end;

        Max_enumerator: INTEGER is 2;
                        -- max enumerator supported in this class

feature {NONE} -- implementation
```

```
        Enum_male: INTEGER is 1;
                        -- enum value for male

        Enum_female: INTEGER is 2;
                        -- enum value for female

        Male_string: STRING is "Male";
                        -- external string representation for male

        Female_string: STRING is "Female";
                        -- external string representation for female

invariant

        invariant_clause: enum <= max_enumerator;

end -- class GENDER_MAP
```

HOME_PANEL

```
Ancestor:

        INTERFACE



indexing
        description: "UI panel for the main screen.";
        author: "Sam Lightstone";
        date: "$Date: $";
        revision: "$Revision: $"

class HOME_PANEL

inherit
        INTERFACE

create
        make

feature

        make (repository_in: PATIENT_DATA_REPOSITORY) is
                        -- Creation routine
                do
                        make_interface (repository_in)
                end;

        process_panel is
                        -- Routine to run the panel
                local
                        time_to_exit: BOOLEAN
                do
                        panel_selection := 0;
                        from
                                time_to_exit := false
                        until
                                time_to_exit = true
                        loop
                                clear_screen;
                                display_panel_header;
                                io.putstring (" 1. Exit this program%N");
                                io.putstring (" 2. Query patient medical information%N");
                                io.putstring (" 3. Enter patient medical event%N");
                                io.putstring (" 4. List all patients%N");
                                io.putstring (" 5. Product overview%N");
                                min_panel_selection_value := 1;
                                max_panel_selection_value := 5;
                                min_selection_value := 1;
                                max_selection_value := 5;
                                request_panel_selection;
                                if valid_panel_selection (panel_selection) then
                                        time_to_exit := true
                                end
                        end
                ensure then
                        valid_panel_selection (panel_selection)
                end;

end -- class HOME_PANEL
```

INTERFACE

```
indexing

        description: "Deferred class for standard I/O ops, used by app. panels.";
        author: "Sam Lightstone";
        date: "$Date: $";
        revision: "$Revision: $"

deferred class INTERFACE

feature {ANY}

        make_interface (repository_in: PATIENT_DATA_REPOSITORY) is
                        -- the generic make routine useful to all INTERFACE objects
                require
                        repository_in /= void
                do
                        patient_data_repository := repository_in
                        selection := 0
                        panel_selection := 0
                        selection_prompt_string := request_selection_string
                ensure
                        post1: patient_data_repository = repository_in;
                        post2: selection = 0;
                        post3: panel_selection = 0
                end;

        get_selection: INTEGER is
                        -- returns the current user selection
                do
                        Result := panel_selection
                ensure
                        post1: Result = panel_selection
                end;

        process_panel is
                        -- display a new UI panel
                deferred
                end;

feature {NONE}

        selection: INTEGER;
                        -- the user selection from any request.

        panel_selection: INTEGER;
                        -- the user selection from a UI panel

        min_selection_value: INTEGER;
                        -- minimum expected selection

        max_selection_value: INTEGER;
                        -- nmax expected selection

        min_panel_selection_value: INTEGER;
                        -- minimum expected selection

        max_panel_selection_value: INTEGER;
                        -- nmax expected selection

        patient_data_repository: PATIENT_DATA_REPOSITORY;
                        -- reference to object providing basic DB operations

        display_header_string (string: STRING) is
                        -- prints a string to stdio in
                        -- a colour designated for headers.
                        -- Not currently used since ANSI colour not supported on all NT
                        -- configurations
                local
                        control_string: STRING;
                        esc_char: CHARACTER;
                        zero_char, black_char, white_char, yellow_char: CHARACTER
                do
                        control_string := "        ";
                        zero_char := '0';
                        esc_char := zero_char - 21;
                        black_char := '0';
                        white_char := '7';
                        yellow_char := '3';
                        control_string.put (esc_char, 1);
                        control_string.put ('[', 2);
                        control_string.put ('3', 3);
                        control_string.put (yellow_char, 4);
                        control_string.put (';', 5);
                        control_string.put ('4', 6);
                        control_string.put (black_char, 7);
                        control_string.put ('m', 8);
                        io.putstring (control_string);
                        io.putstring (string);
                        control_string.put (esc_char, 1);
                        control_string.put ('[', 2);
                        control_string.put ('3', 3);
                        control_string.put (white_char, 4);
                        control_string.put (';', 5);
                        control_string.put ('4', 6);
                        control_string.put (black_char, 7);
                        control_string.put ('m', 8);
                        io.putstring (control_string)
                end;

        get_any_key is
                        -- mapping routine for getch() since
                        -- Eiffel has no comparable routines.
                        -- note that "io.readchar" is busted
                external
                        "C  |<stdio.h>,<conio.h>"
                alias
                        "getch"
                end;

        determine_age (date_time: DATE_TIME): INTEGER is
                        -- given a start date, calculates the the
                        -- current age, in years.
                require
                        date_time /= void
                local
                        current_sys_time: DATE_TIME;
                        duration: DATE_TIME_DURATION
                do
                        create current_sys_time.make (1999, 1, 20, 9, 0, 0);
                        create duration.make (1999, 1, 20, 9, 5, 45);
                        current_sys_time.make_now;
                        duration := current_sys_time.relative_duration (date_time);
                        Result := duration.year
                ensure
                        post1: Result < 135;
                        post2: Result >= 0
                end;

        years_between_dates (date1, date2: DATE_TIME): INTEGER is
                        -- given two dates, caculate the num years
                        -- between them
                require
                        pre1: date1 /= void;
                        pre2: date2 /= void
                local
                        duration: DATE_TIME_DURATION
                do
                        create duration.make (1999, 1, 20, 9, 5, 45);
                        duration := date2.relative_duration (date1);
                        Result := duration.year
                ensure
```

```
                        post1: Result < 500;
                        post2: Result >= 0
                end;

        clear_screen is
                        -- clear the screen
                local
                        icounter: INTEGER
                do
                        from
                                io.new_line
                        until
                                icounter > 100
                        loop
                                io.new_line;
                                icounter := icounter + 1
                        end
                end;

        display_panel_header is
                        -- A header for all user panels that wish to use it.
                do
                        io.new_line
                        io.new_line
                        io.putstring ("         *** EIFFEL PROTOTYPE OF PATIENT DATABASE ***%N%N%N")
                end;

        display_ohip_user is
                        -- displays an OHIP user's meta-characteristics
                        -- in viewable form to screen.
                require
                        pre1: patient_data_repository.is_valid_ohip_user_data;
                        pre2: valid_ohip_num (patient_data_repository.patient_ohip_num)
                local
                        pad_spaces: INTEGER;
                        name_chars: INTEGER;
                        width_of_name_field: INTEGER;
                        counter: INTEGER
                do
                        width_of_name_field := 30;
                        name_chars := patient_data_repository.surname.count +
patient_data_repository.given_names.count;
                        io.putstring ("%N");
                        io.putstring (patient_data_repository.surname);
                        io.putstring (", ");
                        io.putstring (patient_data_repository.given_names);
                        if name_chars < width_of_name_field then
                                pad_spaces := width_of_name_field - name_chars;
                                from
                                        counter := 0
                                until
                                        counter = pad_spaces - 1
                                loop
                                        io.putstring (" ");
                                        counter := counter + 1
                                end
                        end;
                        io.putstring ("OHIP #: ");
                        io.putint (patient_data_repository.patient_ohip_num);
                        io.putstring ("   Age: ");
                        io.putint (determine_age (patient_data_repository.date_of_birth))
                end;

        is_valid_event_data: BOOLEAN is
                        -- sanity check medical event data
                local
                        valid: BOOLEAN
                do
                        if patient_data_repository.absolute_date /= void and patient_data_repository.date_of_birth /=
void and patient_data_repository.practitioner_name /= void and patient_data_repository.practitioner_map /= void and
patient_data_repository.gender_map /= void and patient_data_repository.gender_map.is_set and
patient_data_repository.practitioner_map.is_set = true and patient_data_repository.event_map /= void and
patient_data_repository.event_map.is_set = true then
                                valid := true
                end;
                Result := valid
        end;

selection_prompt_string: STRING;
                -- prompt string to use with request_user_selection

request_user_value is
                -- geta  numeric value from the user.
                -- reuse code from "request_user_selection", but
                -- change the prompt string. Remember to put the
                -- default string back when we're done
        require
                request_value_string /= void;
                request_selection_string /= void
        do
                selection_prompt_string := request_value_string
                request_user_selection
                selection_prompt_string := request_selection_string
        ensure
                selection_prompt_string = request_selection_string
        end;

request_user_selection is
                -- routine to prompt a user for a choice selection.
                -- the user is expected to enter an integer representing one
                -- of the posted choices.
        require
                pre1: min_selection_value >= 0;
                pre2: max_selection_value >= min_selection_value
        local
                input_ok: BOOLEAN
        do
                from
                        input_ok := false
                until
                        input_ok = true
                loop
                        io.putstring (selection_prompt_string);
                        io.new_line;
                        io.new_line;
                        io.readint;
                        selection := io.lastint;
                        if not valid_selection (selection) then
                                io.putstring (bad_selection_string)
                        else
                                input_ok := true
                        end
                end;
                clear_screen
        ensure
                post1: valid_selection (selection)
        rescue
                io.putstring (bad_selection_string);
                retry
        end;

request_panel_selection is
                -- request user selection to UI panel.
        do
                request_user_selection
                panel_selection := selection
        ensure
                valid_selection (panel_selection);
                panel_selection = selection
        end;

valid_selection (selection_in: INTEGER): BOOLEAN is
                -- check whether the current user selection
                -- is valid
        local
                valid: BOOLEAN
        do
                if (selection_in >= min_selection_value) and (selection_in <= max_selection_value) then
```

```
                                valid := true
                        end;
                        Result := valid
                end;

        valid_panel_selection (panel_selection_in: INTEGER): BOOLEAN is
                        -- check whether the current user's panel selection
                        -- is valid
                local
                        valid: BOOLEAN
                do
                        if (panel_selection_in >= min_panel_selection_value) and (panel_selection_in <=
max_panel_selection_value) then
                                valid := true
                        end;
                        Result := valid
                end;

        display_event (hide_identity: BOOLEAN) is
                        -- displays a single medical event
                        -- to the user interface. The client
                        -- is required to hide data not appropriate
                        -- for viewing by marking attributes as
                        -- void/empty.
                        -- output format is based on the following
                        -- template:
                        -- Date: <date>               patient: <name>
                        -- Patient age: <age>, OHIP #: <ohip #> Gender: <gender>
                        -- Consultation type: <consult string>, Area: <med_specialty>
                        -- Patient concern: <complaint>
                        -- Practitioner: <name>, <type string>, Hospital: <hosp name>
                        -- Practitioner comments: <comments>
                        -- Practitioner diagnosis: <diagnosis>
                        -- <blank line>
                require
                        pre1: is_valid_event_data;
                        pre2: not hide_identity implies patient_data_repository.is_valid_ohip_user_data
                local
                        age: INTEGER;
                        month: INTEGER
                do
                        io.putstring ("%NDate: ");
                        io.putint (patient_data_repository.absolute_date.year);
                        io.putstring (", ");
                        month := patient_data_repository.absolute_date.month;
                        io.putstring (patient_data_repository.absolute_date.months_text.item (month));
                        io.putstring (" ");
                        io.putint (patient_data_repository.absolute_date.day);
                        if not hide_identity then
                                if patient_data_repository.surname /= void then
                                        io.putstring ("%NName: ");
                                        io.putstring (patient_data_repository.given_names);
                                        io.putstring (" ");
                                        io.putstring (patient_data_repository.surname)
                                end;
                                age := years_between_dates (patient_data_repository.date_of_birth,
patient_data_repository.absolute_date);
                                io.putstring ("%NAge: ");
                                io.putint (age);
                                if patient_data_repository.surname /= void then
                                        io.putstring ("     OHIP #: ");
                                        io.putint (patient_data_repository.patient_ohip_num)
                                end
                        end;
                        io.putstring ("%NConsultation type: ");
                        io.putstring (patient_data_repository.event_map.get_formatted_string);
                        io.putstring ("    Area: ");
                        io.putstring (patient_data_repository.specialty_map.get_formatted_string);
                        if patient_data_repository.complaint /= void and patient_data_repository.complaint.count > 1
then
                                io.putstring ("%NPatient concern: ");
                                io.putstring (patient_data_repository.complaint)
                        end;
                        io.putstring ("%NPractitioner: ");
                        io.putstring (patient_data_repository.practitioner_name);
                        io.putstring (", ");
                        io.putstring (patient_data_repository.practitioner_map.get_formatted_string);
                        io.putstring ("%NHospital: ");
                        io.putstring (patient_data_repository.hospital_name);
                        if patient_data_repository.comment /= void and patient_data_repository.comment.count > 1 then
                                io.putstring ("%NPractitioner comments: ");
                                io.putstring (patient_data_repository.comment)
                        end;
                        if patient_data_repository.final_diagnosis /= void and
patient_data_repository.final_diagnosis.count > 1 then
                                io.putstring ("%NDiagnosis: ");
                                io.putstring (patient_data_repository.final_diagnosis)
                        end;
                        io.putstring ("%N")
                end;

        please_continue is
                        -- prompt the user with a please hit any key to continue msg.
                do
                        io.putstring ("%N(Hit any key to continue...)%N")
                        get_any_key
                end;

        request_ohip_num: INTEGER is
                        -- prompt the user for an OHIP number, and return it.
                local
                        ohip_num: INTEGER;
                        input_ok: BOOLEAN
                do
                        io.putstring ("%NPlease enter the patient%'s OHIP #, then hit enter.%N");
                        Result := fetch_ohip_num (true)
                ensure
                        post1: valid_ohip_num (Result)
                end;

        fetch_ohip_num (must_be_known: BOOLEAN): INTEGER is
                        -- fetch a valid OHIP number from the user.
                local
                        ohip_num: INTEGER;
                        input_ok: BOOLEAN
                do
                        from
                                input_ok := false
                        until
                                input_ok = true
                        loop
                                io.readint;
                                ohip_num := io.lastint;
                                input_ok := valid_ohip_num (ohip_num);
                                if not input_ok then
                                        if must_be_known then
                                                io.putstring ("%NI%'m sorry, this is not a valid OHIP number.");
                                                io.putstring ("%NA valid OHIP number is a 9 digit number that does not
start with 0");

                                                io.new_line;
                                                io.putstring ("%NPlease enter a vlid OHIP number");
                                                io.new_line
                                        elseif ohip_num = 0 then
                                                input_ok := true
                                        end
                                end
                        end;
                        Result := ohip_num
                ensure
                        post1: must_be_known implies valid_ohip_num (Result);
                        post2: Result = 0 implies must_be_known = false
                end;

        request_medical_specialty (specialty_enum: MED_SPECIALTY_MAP) is
                        -- prompt user for medical specialty
                        -- returns an enum for medical specialty.
                do
                        io.putstring ("%NPlease select a medical specialty from the following list:")
                        io.putstring ("%N1. Obstetrics, 2. Cardiology, 3. Ophthalmology")
                        io.putstring ("%N4. Psychiatry, 5. Oncology, 6. Emergency")
                        io.new_line
                        io.new_line
                        min_selection_value := 1
                        max_selection_value := 6
                        request_user_selection
                        if selection = 1 then
                                specialty_enum.set_obstetrics
                        elseif selection = 2 then
                                specialty_enum.set_cardiology
                        elseif selection = 3 then
                                specialty_enum.set_ophthalmology
                        elseif selection = 4 then
                                specialty_enum.set_psychiatry
                        elseif selection = 5 then
                                specialty_enum.set_oncology
                        else
                                specialty_enum.set_triage
                        end
                ensure
                        post1: selection > 0;
                        post2: selection <= 6;
                        post3: specialty_enum.is_set
                end;

        valid_ohip_num (ohip_num_in: INTEGER): BOOLEAN is
                        -- test whether an ohip number is
                        -- syntactically valid. All ohip #'s must
                        -- be 9 digit integers, which means the
                        -- valid range is 100000000 to 999999999.
                local
                        valid: BOOLEAN
                do
                        if ohip_num_in > 99999999 then
                                if ohip_num_in < 1000000000 then
                                        valid := true
                                end
                        end;
                        Result := valid
                end;

        display_ohip_user_query_results is
                        -- display the results of a query for
                        -- ohip users
                require
                        pre1: patient_data_repository.query_ok
                local
                        counter: INTEGER
                do
                        clear_screen;
                        display_panel_header;
                        patient_data_repository.next_ohip_user;
                        from
                        until
                                patient_data_repository.results_exhausted = true
                        loop
                                display_ohip_user;
                                counter := counter + 1;
                                if counter = 18 then
                                        counter := 0;
                                        please_continue;
                                        clear_screen;
                                        display_panel_header
                                end;
                                patient_data_repository.next_ohip_user
                        end
                end;

        display_event_query_results (hide_identity: BOOLEAN) is
                        -- display the results of a query for
                        -- medical events
                require
                        pre1: patient_data_repository.query_ok
                local
                    counter: INTEGER
                do
                    clear_screen;
                    display_panel_header;
                    patient_data_repository.next_medical_event;
                    from
                    until
                        patient_data_repository.results_exhausted = true
                    loop
                        display_event (hide_identity);
                        counter := counter + 1;
                        if counter = 2 then
                            counter := 0;
                            please_continue;
                            clear_screen;
                            display_panel_header
                        end;
                        patient_data_repository.next_medical_event
                    end
                end;

            invalid_ohip_user_msg is
                    -- write a message indicating patient not
                    -- found in our DB.
                do
                    io.putstring (bad_ohip_user_string)
                end;

            request_practitioner_name: STRING is
                local
                    temp_string: STRING
                do
                    temp_string := "";
                    io.putstring ("%NPlease enter you full professional title:");
                    io.new_line;
                    temp_string := get_user_string;
                    Result := temp_string
                ensure
                    post1: Result.count >= 0
                end;

            request_practitioner_type (practitioner_map: PRACTITIONER_MAP) is
                    -- prompt the user for their practitioner type.
                require
                    practitioner_map /= void
                do
                    io.putstring ("%NIf please enter 1 if you are a doctor, or 2 if you are a nurse")
                    io.new_line
                    min_selection_value := 1
                    max_selection_value := 2
                    request_user_selection
                    if selection = 1 then
                        practitioner_map.set_doctor
                    else
                        practitioner_map.set_nurse
                    end
                ensure
                    practitioner_map.is_set
                end;

            request_comment: STRING is
                    -- prompt the user to enter a comment.
                local
                    temp_string: STRING
                do
                    temp_string := "";
                    io.putstring ("%NPlease enter any comments you have for this consultation:");
                    io.new_line;
                    temp_string := get_user_string;
                    Result := temp_string
                end;

            request_diagnosis: STRING is
```

```
                        -- prompt the user to enter a diagnosis string
                local
                        temp_string: STRING
                do
                        temp_string := "";
                        io.putstring ("%NPlease enter your official diagnosis, and treatment:");
                        io.new_line;
                        temp_string := get_user_string;
                        Result := temp_string
                end;

        request_complaint: STRING is
                        -- prompt the user to enter the patient's medical
                        -- complaint.
                local
                        temp_string: STRING
                do
                        temp_string := "";
                        io.putstring ("%NPlease enter the patient%'s complaint/symptom:");
                        io.new_line;
                        temp_string := get_user_string;
                        Result := temp_string
                end;

        request_hospital_name: STRING is
                        -- prompt the user to enter the name of the hospital
                        -- where the met the patient for the consultation.
                local
                        temp_string: STRING
                do
                        temp_string := "";
                        io.putstring ("%NPlease enter the name of your hospital:");
                        io.new_line;
                        temp_string := get_user_string;
                        Result := temp_string
                end;

        get_user_string: STRING is
                        -- get a string from the user.
                local
                        temp_string: STRING
                do
                        io.readline;
                        temp_string := clone (io.laststring);
                        Result := temp_string
                end;

        Bad_selection_string: STRING is "%NYour selection is out of range. Please try again...%N";

        Bad_ohip_user_string: STRING is "%NI%'m sorry, this OHIP user is not found in our database. %NPlease double
check the number and try again.";

        Request_selection_string: STRING is "Please enter your selection, then hit enter.";

        Request_value_string: STRING is "Please enter a numeric value, then hit enter.";

invariant

        inv1: patient_data_repository /= void;
        inv2: selection >= 0;

end -- class INTERFACE
```

LIST_USERS_PANEL

Ancestor:

INTERFACE

```
indexing

        description: "UI Panel for listing all ohip users in our system.";
        author: "Sam Lightstone";
        date: "$Date: $";
        revision: "$Revision: $"

class LIST_USERS_PANEL

inherit
        INTERFACE

create
        make

feature -- Initialization

        make (repository_in: PATIENT_DATA_REPOSITORY) is
                        -- Creation routine
                do
                        make_interface (repository_in)
                end;

        process_panel is
                        -- Routine to run the panel. displayus all known
                        -- OHIP users in the system, sorted by last then first name.
                local
                        counter: INTEGER;
                        is_found: BOOLEAN;
                        new_user: OHIP_USER;
                        dob: DATE_TIME
                do
                        counter := 0;
                        max_panel_selection_value := 1;
                        min_panel_selection_value := 1;
                        max_selection_value := 1;
                        min_selection_value := 1;
                        panel_selection := 1;
                        clear_screen;
                        display_panel_header;
                        patient_data_repository.query_all_patients;
                        patient_data_repository.next_ohip_user;
                        from
                        until
                                patient_data_repository.results_exhausted
                        loop
                                display_ohip_user;
                                counter := counter + 1;
                                if counter = 18 then
                                        counter := 0;
                                        please_continue;
                                        clear_screen;
                                        display_panel_header;
                                        io.put_string (list_of_users_header)
                                end;
                                patient_data_repository.next_ohip_user
                        end;
                        please_continue
                ensure then
                        post1: patient_data_repository.db_query_in_prog = false;
                        post2: valid_panel_selection (panel_selection)
                end;

feature {NONE} -- Implementation
```

```
        List_of_users_header: STRING is "%N%N   list of known OHIP users registered in our system";
                       -- Header text for listing all known OHIP users.
                       -- Your invariant here

end -- class LIST_USERS_PANEL
```

MED_SPECIALTY_MAP

```
Ancestor:

        ENUM_MAP


indexing

        description: "Types of medical specialties we support";

        author: "Sam Lightstone";

        date: "$Date: $";
        revision: "$Revision: $"

class MED_SPECIALTY_MAP

inherit
        ENUM_MAP

create
        make

feature {ANY}

        set_obstetrics is
                        -- sets medical specialty to obstetrics
                do
                        enum := enum_obstetrics
                        string := obstetrics_string
                        is_set := true
                ensure
                        post1: is_set = true;
                        post2: enum = enum_obstetrics
                end;

        set_cardiology is
                        -- sets medical specialty to cardiology
                do
                        enum := enum_cardiology
                        string := cardiology_string
                        is_set := true
                ensure
                        post1: is_set = true;
                        post2: enum = enum_cardiology
                end;

        set_ophthalmology is
                        -- sets medical specialty to ophthalmology
                do
                        enum := enum_ophthalmology
                        string := ophthalmology_string
                        is_set := true
                ensure
                        post1: is_set = true;
                        post2: enum = enum_ophthalmology
                end;

        set_psychiatry is
                        -- sets medical specialty to psychiatry
                do
                        enum := enum_psychiatry
                        string := psychiatry_string
                        is_set := true
                ensure
                        post1: is_set = true;
                        post2: enum = enum_psychiatry
                end;
        set_oncology is
                        -- sets medical specialty to oncology
                do
                        enum := enum_oncology
                        string := oncology_string
                        is_set := true
                ensure
                        post1: is_set = true;
                        post2: enum = enum_oncology
                end;

        set_triage is
                        -- sets medical specialty to triage
                do
                        enum := enum_triage
                        string := triage_string
                        is_set := true
                ensure
                        post1: is_set = true;
                        post2: enum = enum_triage
                end;

        set_specialty_from_enum (enum_in: INTEGER) is
                        -- sets medical specialty from an enum value. The
                        -- enum value passed should have been previously
                        -- obtained from features of this class. This will
                        -- also set the external string representation.
                require
                        pre1: enum_in > 0;
                        pre2: enum_in <= max_enumerator
                do
                        is_set := false
                        if enum_in = enum_obstetrics then
                                set_obstetrics
                        elseif enum_in = enum_cardiology then
                                set_cardiology
                        elseif enum_in = enum_ophthalmology then
                                set_ophthalmology
                        elseif enum_in = enum_psychiatry then
                                set_psychiatry
                        elseif enum_in = enum_oncology then
                                set_oncology
                        elseif enum_in = enum_triage then
                                set_triage
                        end
                ensure
                        post1: enum = enum_in;
                        post2: is_set = true
                end;

        Max_enumerator: INTEGER is 6;
                        -- the max enumerator supported in the class

feature {NONE}

        Enum_obstetrics: INTEGER is 1;
                        -- enum value for obstetrics

        Enum_cardiology: INTEGER is 2;
                        -- enum value for cardiology

        Enum_ophthalmology: INTEGER is 3;
                        -- enum value for ophthalmology

        Enum_psychiatry: INTEGER is 4;
                        -- enum value for psychiatry

        Enum_oncology: INTEGER is 5;
                        -- enum value for oncology

        Enum_triage: INTEGER is 6;
                        -- enum value for triage
        Obstetrics_string: STRING is "Obstetrics";
                        -- external string representation for obstetrics

        Cardiology_string: STRING is "Cardiology";
                        -- external string representation for cardiology

        Ophthalmology_string: STRING is "Ophthalmology";
                        -- external string representation for ophthalmology

        Psychiatry_string: STRING is "Psychiatry";
                        -- external string representation for psychiatry

        Oncology_string: STRING is "Oncology";
                        -- external string representation for oncology

        Triage_string: STRING is "Emergency (ER)";
                        -- external string representation for triage

invariant

        invariant_clause: enum <= max_enumerator;

end -- class MED_SPECIALTY_MAP
```

MEDICAL_DATABASE

```
Ancestor:

        RDB_HANDLE



indexing

        description: "Database layer, for DB-level (not table) ops";
        author: "Sam Lightstone";
        date: "$Date: $";
        revision: "$Revision: $"

class MEDICAL_DATABASE

inherit
        RDB_HANDLE

create
        make

feature

        make (db_name_in: STRING) is
                local
                        tmp_string: STRING
                do
                        set_data_source (db_name_in);
                        login ("", "");
                        set_base;
                        create session_control.make;
                        session_control.connect;
                        if session_control.is_connected then
                                is_connected := true
                        end
                end;

        is_connected: BOOLEAN;

        session_control: DB_CONTROL;

        db_cleanup is
                        -- normal garbage collection isn't quite enough in this application.
                        -- If we are connect to a database, we need to disconnect explicitly.
                require
                        session_control /= void
                do
                        if session_control.is_connected then
                                session_control.disconnect
                        end
                ensure
                        not session_control.is_connected
                end;

        session_ok: BOOLEAN is
                require
                        session_control /= void
                do
                        Result := session_control.is_ok
                end;

invariant

        session_control /= void;

end -- class MEDICAL_DATABASE
```

MEDICAL_EVENT

```
indexing

        description: "Mapping class, for repositories associated with table MEDICAL_EVENTS";

        author: "Sam Lightstone";

        date: "$Date: $";

        revision: "$Revision: $"


class MEDICAL_EVENT


create

        make

feature

        make is
                do
                        complaint := ""
                        practitioner_name := ""
                        hospital_name := ""
                        final_diagnosis := ""
                        comment := ""
                        patient_ohip_num := 0
                        create absolute_date.make (1999, 1, 20, 2, 30, 0)
                end;

feature {ANY}

        patient_ohip_num: INTEGER;

        complaint: STRING;

        medical_specialty: INTEGER;

        practitioner_name: STRING;

        practitioner_type: INTEGER;

        absolute_date: DATE_TIME;

        hospital_name: STRING;

        final_diagnosis: STRING;

        comment: STRING;

        event_type: INTEGER;

        reset_patient_ohip_num is
                        -- resets the patient_ohip_num attribute to 0
                do
                        patient_ohip_num := 0
                ensure
                        patient_ohip_num = 0
                end;

        set_attributes (complaint_in, pract_name_in, hospital_name_in, diagnosis_in, comment_in: STRING; ohip_num_in,
specialty_in, pract_type_in, event_type_in: INTEGER; date_time_in: DATE_TIME) is
                require
                        pre1: ohip_num_in > 0;
                        pre2: pract_name_in.count > 0;
                        pre3: hospital_name_in.count > 0;
                        pre4: event_type_in >= 0;
                        pre5: date_time_in /= void;
                        pre6: specialty_in > 0;
                        pre7: pract_type_in > 0
                do
                        patient_ohip_num := ohip_num_in
                        complaint := clone (complaint_in)
                        medical_specialty := specialty_in
                        practitioner_name := clone (pract_name_in)
                        practitioner_type := pract_type_in
                        absolute_date := date_time_in
                        hospital_name := clone (hospital_name_in)
                        final_diagnosis := clone (diagnosis_in)
                        comment := clone (comment_in)
                        event_type := event_type_in
                ensure
                        post1: patient_ohip_num = ohip_num_in;
                        post2: complaint /= void;
                        post3: medical_specialty = specialty_in;
                        post4: practitioner_name /= void;
                        post5: practitioner_type = pract_type_in;
                        post6: absolute_date = date_time_in;
                        post7: hospital_name /= void;
                        post8: final_diagnosis /= void;
                        post9: comment /= void;
                        post10: event_type = event_type_in
                end;

invariant

        patient_ohip_num >= 0;
        absolute_date /= void;

end -- class MEDICAL_EVENT
```

NEW_EVENT_PANEL

```
Ancestor:

        INTERFACE


indexing
        description: "  Panel for user to enter new patient events data after a patient consultation.";
        author: "Sam Lightstone";
        date: "$Date: $";
        revision: "$Revision: $"

class NEW_EVENT_PANEL

inherit
        INTERFACE

create
        make

feature -- Initialization

        make (repository_in: PATIENT_DATA_REPOSITORY) is
                        -- Creation routine
                do
                        make_interface (repository_in)
                        create specialty_map.make
                        create event_map.make
                        create gender_map.make
                        create practitioner_map.make
                        create cur_date_time.make (1999, 1, 20, 2, 30, 0)
                        create new_user_dob2.make (1999, 1, 20, 2, 30, 0)
                ensure then
                        post1: specialty_map /= void;
                        post2: event_map /= void;
                        post3: gender_map /= void;
                        post4: practitioner_map /= void;
                        post5: cur_date_time /= void
                end;

feature {NONE} -- Implementation

        specialty_map: MED_SPECIALTY_MAP;
                        -- Map object for medical specialty enumerated type

        event_map: EVENT_MAP;
                        -- Map object for consultation type

        gender_map: GENDER_MAP;
                        -- Map objec for gender type

        practitioner_map: PRACTITIONER_MAP;
                        -- Map object for practitioner type

        process_panel is
                        --      Routine to run the pane. Displays the panel for
                        -- entering new consultation data. This may have the
                        -- side effect of entering a new OHIP user in the
                        -- system.
                require else
                        pre1: patient_data_repository.db_query_in_prog = false
                local
                        btimetoexit: BOOLEAN
                do
                        clear_screen;
                        display_panel_header;
                        io.putstring (" 1. Enter triage data%N");
                        io.putstring (" 2. Enter intermediate consultation%N");
                        io.putstring (" 3. Enter diagnosis and treatment%N");
                        io.putstring (" 4. Back to home%N");
                        min_panel_selection_value := 1;
                        max_panel_selection_value := 4;
                        min_selection_value := 1;
                        max_selection_value := 4;
                        request_panel_selection;
                        if panel_selection /= 4 then
                                if selection = 1 then
                                        event_map.set_triage
                                elseif selection = 2 then
                                        event_map.set_intermediate
                                elseif selection = 3 then
                                        event_map.set_diagnosis
                                end;
                                establish_ohip_user;
                                request_event_data;
                                insert_new_event_data
                        end
                ensure then
                        post1: patient_data_repository.db_query_in_prog = false;
                        post2: valid_panel_selection (panel_selection)
                end;

        practitioner_name: STRING;

        hospital_name: STRING;

        diagnosis_string: STRING;

        comment_string: STRING;

        complaint_string: STRING;

        cur_date_time: DATE_TIME;

        cur_ohip_num: INTEGER;

        establish_ohip_user is
                        -- establish which ohip user we are entering data
                        -- for. This may be a new user! Redundancy: Prompt user
                        -- for typos.
                require
                        pre1: patient_data_repository.db_query_in_prog = false
                local
                        user_happy: BOOLEAN;
                        new_ohip_user: BOOLEAN;
                        ohip_user_exists: BOOLEAN
                do
                        from
                        until
                                user_happy
                        loop
                                io.putstring ("%N%NPlease enter the patient%'s OHIP number:%N");
                                cur_ohip_num := fetch_ohip_num (true);
                                io.new_line;
                                ohip_user_exists := patient_data_repository.is_patient_in_ohip_repository
(cur_ohip_num);
                                if not ohip_user_exists then
                                        new_ohip_user := prompt_for_new_user;
                                        if new_ohip_user then
                                                request_ohip_user_info;
                                                insert_new_user_data;
                                                patient_data_repository.bind_out_last_ohip_user;
                                                user_happy := true
                                        end
                                else
                                        patient_data_repository.bind_out_last_ohip_user;
                                        io.putstring ("%NYou are updating records for the following patient: ");
                                        io.putstring ("%N   ");
                                        io.putstring (patient_data_repository.given_names);
                                        io.putstring (" ");
                                        io.putstring (patient_data_repository.surname);
                                        io.putstring ("%NIf this is correct enter 1, otherwise enter 2:%N");
                                        min_selection_value := 1;
                                        max_selection_value := 2;
                                        request_user_selection;
                                        if selection = 1 then
                                                user_happy := true
                                        end
                                end
                        end
                ensure
                        post1: patient_data_repository.db_query_in_prog = false
                end;

        prompt_for_new_user: BOOLEAN is
                        -- ask user if the ohip number they requested, which we
                        -- did not find in the DB is a new user, or a typo.
                        -- return TRUE if they intend to add a new user to the db.
                do
                        io.putstring ("%N%NThe ohip user you requested was not found in the database")
                        io.putstring ("%NType 1 if this is a new ohip user, or 2 if you%'d like to try again:%N")
                        min_selection_value := 1
                        max_selection_value := 2
                        request_user_selection
                        if selection = 1 then
                                Result := true
                        end
                ensure
                        Result implies (selection = 1)
                end;

        request_event_data is
                        -- fetch a pile of event characteristics from the user.
                do
                        request_medical_specialty (specialty_map)
                        clear_screen
                        practitioner_name := request_practitioner_name
                        clear_screen
                        request_practitioner_type (practitioner_map)
                        clear_screen
                        hospital_name := request_hospital_name
                        clear_screen
                        comment_string := request_comment
                        clear_screen
                        diagnosis_string := ""
                        complaint_string := ""
                        if event_map.is_diagnosis then
                                diagnosis_string := request_diagnosis
                                clear_screen
                        elseif event_map.is_triage then
                                complaint_string := request_complaint;
                                clear_screen
                        end
                        cur_date_time.make_now
                ensure
                        post1: practitioner_name.count > 0;
                        post2: event_map.is_diagnosis implies diagnosis_string.count > 3;
                        post3: event_map.is_triage implies complaint_string.count > 3
                end;

        new_user_last_name: STRING;

        new_user_first_names: STRING;

        new_user_dad_ohip_num: INTEGER;

        new_user_mom_ohip_num: INTEGER;

        new_user_dob: DATE;

        new_user_dob2: DATE_TIME;

        new_user_gender: GENDER_MAP;

        day_of_birth: INTEGER;

        month_of_birth: INTEGER;

        year_of_birth: INTEGER;
```

```
        request_ohip_user_info is
                        -- query info about a new ohip user
                        -- currently not foudn in our DB.
                local
                        day, month, year: INTEGER
                do
                        clear_screen;
                        create new_user_gender.make;
                        io.putstring ("%NYou will now be asked to enter information about this OHIP user.");
                        io.putstring ("%NPlease enter the patients last name%N");
                        new_user_last_name := get_user_string;
                        io.putstring ("%NPlease enter the patients first names%N");
                        new_user_first_names := get_user_string;
                        io.putstring ("%NThe patient%'s date of bith is required.%N");
                        request_date_of_birth;
                        io.putstring ("%NPlease enter the patient%'s mother%'s OHIP number, %Nor enter 0 to indicate
it is not known%N");
                        new_user_mom_ohip_num := fetch_ohip_num (false);
                        io.putstring ("%NPlease enter the patient%'s father%'s OHIP number, %Nor enter 0 to indicate
it is not known%N");
                        new_user_dad_ohip_num := fetch_ohip_num (false);
                        io.putstring ("%NPlease enter 1 if the patient is male, or 2 if the patient is female%N");
                        min_selection_value := 1;
                        max_selection_value := 2;
                        request_user_selection;
                        if selection = 1 then
                                new_user_gender.set_male
                        else
                                new_user_gender.set_female
                        end;
                        io.putstring ("%NThe new user information you have entered is as follows:");
                        io.putstring ("%N");
                        io.putstring (new_user_first_names);
                        io.putstring ("  ");
                        io.putstring (new_user_last_name);
                        io.putstring ("%NDate of birth: ");
                        io.putstring (new_user_dob.out);
                        io.putstring ("%NOHIP # : ");
                        io.putint (cur_ohip_num);
                        io.putstring ("%NMother%'s OHIP #: ");
                        io.putint (new_user_mom_ohip_num);
                        io.putstring ("%NFather%'s OHIP #: ");
                        io.putint (new_user_dad_ohip_num);
                        io.putstring ("%NGender: ");
                        io.putstring (new_user_gender.get_formatted_string);
                        io.putstring ("%N%NIf this is correct, please type 1 otherwise type 2 %N");
                        min_selection_value := 1;
                        max_selection_value := 2;
                        request_user_selection;
                        if selection = 2 then
                                request_ohip_user_info
                        end
                ensure
                        post1: new_user_last_name.count > 1;
                        post2: new_user_first_names.count > 1;
                        post3: valid_ohip_num (cur_ohip_num);
                        post4: new_user_dad_ohip_num >= 0;
                        post5: new_user_mom_ohip_num >= 0;
                        post6: new_user_dob /= void;
                        post7: new_user_dob2 /= void;
                        post8: new_user_gender.is_set
                end;

        insert_new_user_data is
                        -- insert a new OHIP user into the db!
                require
                        pre1: patient_data_repository.db_query_in_prog = false
                local
                        new_user: OHIP_USER
                do
                        create new_user.make;
                        new_user.set_attributes (new_user_first_names, new_user_last_name, cur_ohip_num,
new_user_mom_ohip_num, new_user_dad_ohip_num, new_user_gender.get_enum, new_user_dob2);
```

```
                        patient_data_repository.new_patient (new_user)
                ensure
                        post1: patient_data_repository.db_query_in_prog = false
                end;

        insert_new_event_data is
                require
                        patient_data_repository.patient_ohip_num /= 0;
                        patient_data_repository.db_query_in_prog = false
                local
                        event: MEDICAL_EVENT
                do
                        create event.make;
                        event.set_attributes (complaint_string, practitioner_name, hospital_name, diagnosis_string,
comment_string, patient_data_repository.patient_ohip_num, specialty_map.get_enum, practitioner_map.get_enum,
event_map.get_enum, cur_date_time);
                        patient_data_repository.new_medical_event (event)
                ensure
                        post1: patient_data_repository.db_query_in_prog = false
                end;

        request_date_of_birth is
                        -- prompt a user for date of birth information.
                require
                        new_user_dob2 /= void
                do
                        io.putstring ("%NPlease enter the year as a four digit value.%N")
                        min_selection_value := 1850
                        max_selection_value := 2200
                        request_user_value
                        year_of_birth := selection
                        io.putstring ("%NPlease enter the month, as a number between 1 and 12 %N")
                        min_selection_value := 1
                        max_selection_value := 12
                        request_user_value
                        month_of_birth := selection
                        io.putstring ("Please enter the day of the month as a number between 1 and 31%N")
                        min_selection_value := 1
                        max_selection_value := 31
                        request_user_value
                        day_of_birth := selection
                        create new_user_dob.make_day_month_year (day_of_birth, month_of_birth, year_of_birth)
                        new_user_dob2.set_date (new_user_dob)
                ensure
                        new_user_dob /= void;
                        new_user_dob2 /= void;
                        determine_age (new_user_dob2) < 135
                rescue
                        io.putstring ("%NThe date value you have provided is not valid.");
                        retry
                end;

invariant

        specialty_map /= void;
        event_map /= void;
        gender_map /= void;
        practitioner_map /= void;

end -- class NEW_EVENT_PANEL
```

OHIP_USER

```
indexing

        description: "Mapping class for repositories related to table OHIP_USERS.";
        author: "Sam Lightstone";
        date: "$Date: $";
        revision: "$Revision: $"

class OHIP_USER

create
        make

feature

        make is
                do
                        surname := ""
                        given_names := ""
                        patient_ohip_num := 0
                        create date_of_birth.make (1999, 1, 20, 2, 30, 0)
                end;

feature {ANY}

        given_names: STRING;

        surname: STRING;

        patient_ohip_num: INTEGER;

        mom_ohip_num: INTEGER;

        dad_ohip_num: INTEGER;

        date_of_birth: DATE_TIME;

        gender: INTEGER;

        reset_patient_ohip_num is
                        -- resets the patient_ohip_num attribute to 0
                do
                        patient_ohip_num := 0
                ensure
                        patient_ohip_num = 0
                end;

        set_attributes (name_in, lastname_in: STRING; ohip_num_in, mom_num_in, dad_num_in, sex_in: INTEGER; dob_in:
DATE_TIME) is
                require
                        pre1: ohip_num_in > 0;
                        pre2: name_in.count > 0;
                        pre3: lastname_in.count > 0;
                        pre4: name_in.count > 0;
                        pre5: dob_in /= void
                do
                        given_names := clone (name_in)
                        surname := clone (lastname_in)
                        patient_ohip_num := ohip_num_in
                        mom_ohip_num := mom_num_in
                        dad_ohip_num := dad_num_in
                        gender := sex_in
                        date_of_birth := dob_in
                ensure
                        post1: given_names /= void;
                        post2: surname /= void;
                        post3: patient_ohip_num = ohip_num_in;
                        post4: mom_ohip_num = mom_num_in;
                        post5: dad_ohip_num = dad_num_in;
                        post6: gender = sex_in;
                        post7: date_of_birth = dob_in
```

```
               end;

invariant

        surname /= void;
        given_names /= void;
        date_of_birth /= void;
        patient_ohip_num >= 0;

end -- class OHIP_USER
```

OVERVIEW_PANEL

```
Ancestor:

        INTERFACE



indexing

        description: "UI panel for displaying product overview";

        author: "Sam Lightstone";

        date: "$Date: $";

        revision: "$Revision: $"



class OVERVIEW_PANEL

inherit
        INTERFACE

create
        make

feature -- Initialization

        make (repository_in: PATIENT_DATA_REPOSITORY) is
                        -- creation routine
                do
                        make_interface (repository_in)
                end;

        process_panel is
                        -- routine to run the panel
                do
                        min_panel_selection_value := 1
                        max_panel_selection_value := 1
                        min_selection_value := 1
                        max_selection_value := 1
                        panel_selection := 1
                        display_program_overview
                ensure then
                        valid_panel_selection (panel_selection)
                end;

feature {NONE} -- Implementation

        display_program_overview is
                        -- routine to display the product overview text
                require
                        overview_text1.count > 0;
                        overview_text2.count > 0;
                        overview_text3.count > 0
                do
                        clear_screen
                        display_panel_header
                        io.new_line
                        io.putstring (overview_text1)
                        io.new_line
                        please_continue
                        clear_screen
                        io.putstring (overview_text2)
                        io.new_line
                        please_continue
                        clear_screen
                        io.putstring (overview_text3)
                        io.new_line
                        please_continue
```

```
              end;

    Overview_text1: STRING is
            -- Overview text part I

        "%N%N%N%N%N%N%N%N%N%N%N%N%
         %This program is a prototype.  Its aim is to show how        %N%
         %patient information from regional hospitals can be organized %N%
         %in a computer database yielding a number of important       %N%
         %benefits to patient care.  These benefits include:  %N%N%N%N%N%N%N%N%N%N%N"

    Overview_text2: STRING is
            -- Overview text part II

        "1. The ability to maintain and present patient histories    %N%
        %across regional hospitals.                                 %N%
        %                                                            %N%
        %2. The ability to query patient histories in entirety       %N%
        %or by medical specialty.                                    %N%
        %                                                            %N%
        %3. The ability to correlate patient medical histories with   %N%
        %those of close family members, while keeping the full       %N%
        %identity of these family members anonymous.  Close relatives %N%
        %can share common genetic and possibly environmental        %N%
        %characteristics. However, medical histories may not be       %N%
        %known between family members. By storing and correlating       %N%
        %family medical histories in a database it is possible to       %N%
        %discover medical relationships unknown to the patients     %N%
        %themsleves.   %N%
        %                                                            %N%
        %4. The ability to query a patient's last known              %N%
        %hospital visit anywhere in a defined medical region;         %N%
        %reported in nursing literature as a useful feature for      %N%
        %patient triage.                                            "

    Overview_text3: STRING is
            -- Overview text part III
        "%N%N%N%%N%N%N%
        %This system provides for both patient query, as well as      %N%
        %insert of new patients, and new consultation data.           %N%
        %                                                              %N%
        %In a production system, patients' personal OHIP information is %N%
        %encoded on the magnetic strip of the patient's OHIP card. After %N%
        %admission to the hospital, the data is replicated in a bar code %N%
        %located on the patient's wrist band.                         %N%N%
        %In the long run this system may form the backbone of a large %N%
        %medical datamining data warehouse, which could be used      %N%
        %to discover medical relationships between diseases          %N%
        %previously unrecognized in professional medical research. %N%
        %  %N%N%N%N%N%N%N%N"


end -- class OVERVIEW_PANEL
```

PATIENT_DATA_REPOSITORY

```
indexing

        description: "Interface to medical repositories and their ops.";

        author: "Sam Lightstone";

        date: "$Date: $";

        revision: "$Revision: $"


class PATIENT_DATA_REPOSITORY


create

        make


feature {ANY}


        tables_conform: BOOLEAN;

                        -- Does a required table exist?

        db_query_in_prog: BOOLEAN;
                        -- Is a query currently in progress?

        query_ok: BOOLEAN;
                        -- Did the last query  run ok?

        update_ok: BOOLEAN;
                        -- Was the last database INSERT/UPDATE successful?

        connected_ok: BOOLEAN;
                        -- did we connect tot he database ok?

        make (db_name_in: STRING) is
                        -- Creation routine. Instantiate the medical repository
                        -- and other reference objects.
                do
                        date_time_string := ""
                        create medical_database.make (db_name_in)
                        if medical_database.is_connected then
                                connected_ok := true
                        end
                        if connected_ok then
                                create base_selection.make
                                create store.make
                                create medical_event.make
                                create ohip_user.make
                                create gender_map.make
                                create event_map.make
                                create practitioner_map.make
                                create specialty_map.make
                                create event_repository.make (event_table_name)
                                create ohip_repository.make (ohip_table_name)
                                create my_cursor.make
                                event_repository.load
                                ohip_repository.load
                                if event_repository.exists then
                                        tables_conform := true
                                else
                                        tables_conform := false
                                end
```

```
                                    if ohip_repository.exists then
                                            tables_conform := true
                                    else
                                            tables_conform := false
                                    end
                            end
                    end;

    feature {ANY}

            last_query_had_result: BOOLEAN;
                            -- any values returned form last query?

            given_names: STRING;
                            -- patient given names

            surname: STRING;
                            -- patient surname

            patient_ohip_num: INTEGER;
                            -- patient OHIP #

            mom_ohip_num: INTEGER;
                            -- patient's mother's OHIP #

            dad_ohip_num: INTEGER;
                            -- patient's father's OHIP #

            date_of_birth: DATE_TIME;
                            -- Patient's date of birth

            complaint: STRING;
                            -- patient's complaint that caused them to visit the hospital

            practitioner_name: STRING;
                            -- The practitioner's name (Nurse or Doctor)

            absolute_date: DATE_TIME;
                            -- Date and time of a medical consultation

            hospital_name: STRING;
                            -- the name of the hospital where the current medical consultation took place

            final_diagnosis: STRING;
                            -- The final diagnosis from the attending medical practitioner

            comment: STRING;
                            -- Comments from the medical practitioner

            specialty_map: MED_SPECIALTY_MAP;
                            -- Current medical specialty (obstetrics, oncology etc)

            event_map: EVENT_MAP;
                            -- type of consultation (triage, diagnosis, etc)

            practitioner_map: PRACTITIONER_MAP;
                            -- Type of practitioner (nurse, docutor etc)

            gender_map: GENDER_MAP;
                            -- gender of the patient

            results_exhausted: BOOLEAN;
                            -- Any remaining data to fetch for this query?

            repository_cleanup is
                            -- garbage collection may not handle DB
                            -- teardown. We should add this explicitly
                    do
                            medical_database.db_cleanup
                    end;

            bind_out_last_ohip_user is
                            -- bind out the last ohip_user we queried/inserted
                            -- if there was one. Otherwise, just NOP.
```

```
                do
                        if ohip_user.patient_ohip_num > 0 then
                                bind_out_ohip_user
                                cur_patient_ohip_num := ohip_user.patient_ohip_num
                        end
                ensure
                        post1: (ohip_user.patient_ohip_num > 0) implies is_valid_ohip_user_data;
                        post2: (ohip_user.patient_ohip_num > 0) implies (cur_patient_ohip_num =
ohip_user.patient_ohip_num)
                end;

        is_valid_ohip_user_data: BOOLEAN is
                        -- sanity check ohip user data.
                require
                        pre1: given_names /= void;
                        pre2: surname /= void;
                        pre3: gender_map /= void
                local
                        valid: BOOLEAN
                do
                        if given_names.count > 0 and surname.count > 0 and mom_ohip_num >= 0 and dad_ohip_num >= 0 and
date_of_birth /= void and patient_ohip_num > 0 and gender_map.is_set then
                                valid := true
                        end;
                        Result := valid
                end;

        is_patient_in_ohip_repository (ohip_num: INTEGER): BOOLEAN is
                        -- determine if this user exists in the ohip
                        -- repository or not
                require
                        pre1: ohip_num > 0
                do
                        ohip_user.reset_patient_ohip_num
                        select_string := select_user_from_ohip_repository
                        base_selection.set_map_name (ohip_num, "predicate_ohip_num")
                        run_query
                        if not base_selection.exhausted then
                                my_cursor.fill_in
                                base_selection.object_convert (ohip_user)
                                base_selection.cursor_to_object
                                cur_patient_ohip_num := ohip_num
                        end
                        base_selection.reset_cursor (my_cursor)
                        base_selection.terminate
                        medical_database.session_control.commit
                        results_exhausted := true
                        db_query_in_prog := false
                        if ohip_user.patient_ohip_num > 0 then
                                last_query_had_result := true
                        end
                        base_selection.unset_map_name ("predicate_ohip_num")
                        Result := last_query_had_result
                ensure
                        ohip_user.patient_ohip_num > 0 implies Result = true
                end;

        new_patient (new_user: OHIP_USER) is
                        -- Enters a new patient in the ohip_users respository
                        -- Insert is amazingly easy using Eiffel
                        -- object-to-repository mapping!
                require
                        pre1: is_valid_ohip_user_obj (new_user)
                do
                        update_ok := false
                        store.set_repository (ohip_repository)
                        store.put (new_user)
                        if medical_database.session_ok then
                                medical_database.session_control.commit
                                if medical_database.session_ok then
                                        update_ok := true
                                end
                        end
                        cur_patient_ohip_num := new_user.patient_ohip_num
                        ohip_user := clone (new_user)
                ensure
                        post1: update_ok implies is_patient_in_ohip_repository (cur_patient_ohip_num);
                        post2: cur_patient_ohip_num = new_user.patient_ohip_num;
                        post3: ohip_user.patient_ohip_num = new_user.patient_ohip_num
                end;

        new_medical_event (event: MEDICAL_EVENT) is
                        -- Enters a new medical event into the DB.
                        -- Requires the patient exist in the ohip_repository
                        -- Insert is amazingly easy using Eiffel
                        -- object-to-repository mapping!
                require
                        pre1: is_patient_in_ohip_repository (cur_patient_ohip_num);
                        pre2: event.medical_specialty <= specialty_map.max_enumerator;
                        pre3: event.practitioner_name.count > 0;
                        pre4: event.practitioner_type <= practitioner_map.max_enumerator;
                        pre5: event.absolute_date /= void;
                        pre6: event.hospital_name.count > 0;
                        pre7: event.event_type <= event_map.max_enumerator
                do
                        update_ok := false
                        store.set_repository (event_repository)
                        store.put (event)
                        if medical_database.session_ok then
                                update_ok := true
                        end
                end;

        query_patient_diagnosis_history is
                        -- Retrieve a patient's medical history, listing only
                        -- the final diagnosis for each problem.
                require
                        pre1: is_patient_in_ohip_repository (cur_patient_ohip_num)
                do
                        base_selection.set_map_name (cur_patient_ohip_num, "predicate_ohip_num")
                        select_string := select_patient_diagnosis_history
                        run_query
                        base_selection.unset_map_name ("predicate_ohip_num")
                ensure
                        post1: query_ok
                end;

        query_patient_history_by_specialty (specialty_enum: MED_SPECIALTY_MAP; diagnosis_only: BOOLEAN) is
                        -- For a given medical specifalty, retrieve a patient's
                        -- medical history, listing only the final diagnosis for each problem.
                require
                        pre1: is_patient_in_ohip_repository (cur_patient_ohip_num)
                local
                        spec_enum_val: INTEGER
                do
                        spec_enum_val := specialty_enum.get_enum;
                        base_selection.set_map_name (cur_patient_ohip_num, "predicate_ohip_num");
                        base_selection.set_map_name (spec_enum_val, "predicate_specialty_enum");
                        if diagnosis_only then
                                select_string := select_patient_diagnosis_history_by_specialty
                        else
                                select_string := select_patient_history_by_specialty
                        end;
                        run_query;
                        base_selection.unset_map_name ("predicate_ohip_num");
                        base_selection.unset_map_name ("predicate_specialty_enum")
                ensure
                        query_ok
                end;

        query_siblings is
                        -- Retrieve the set of siblings for the current patient
                require
                        pre1: is_patient_in_ohip_repository (cur_patient_ohip_num)
                do
                        if mom_ohip_num = default_ohip_num then
                                mom_ohip_num := invalid_ohip_num
                        end
                        if dad_ohip_num = default_ohip_num then
                                dad_ohip_num := invalid_ohip_num
                        end
                        base_selection.set_map_name (mom_ohip_num, "mom_predicate_ohip_num")
                        base_selection.set_map_name (dad_ohip_num, "dad_predicate_ohip_num")
                        select_string := select_siblings_from_ohip_repository
                        run_query
                        base_selection.unset_map_name ("mom_predicate_ohip_num")
                        base_selection.unset_map_name ("dad_predicate_ohip_num")
                ensure
                        post1: query_ok
                end;

        query_children is
                        -- Retrieve the set of children who have the current
                        -- patient as a parent
                require
                        pre1: is_patient_in_ohip_repository (cur_patient_ohip_num)
                local
                        ohip1: INTEGER;
                        ohip2: INTEGER
                do
                        ohip1 := cur_patient_ohip_num;
                        ohip2 := cur_patient_ohip_num;
                        base_selection.set_map_name (ohip1, "predicate_ohip_num");
                        select_string := select_offspring_from_ohip_repository;
                        run_query;
                        base_selection.unset_map_name ("predicate_ohip_num")
                ensure
                        post1: query_ok
                end;

        query_relatives_diagnosis_history_by_specialty is
                        -- For a given medical specialty, retrieve the medical histories
                        -- listing only the final diagnosis for each problem for apatient's
                        -- relatives (i.e. Parents & children)
                require
                        pre1: is_patient_in_ohip_repository (cur_patient_ohip_num)
                do
                ensure
                        post1: query_ok
                end;

        query_patient_last_visit is
                        -- List the last recorded medical event for a patient in the database.
                require
                        pre1: is_patient_in_ohip_repository (cur_patient_ohip_num)
                do
                        select_string := select_patient_last_visit
                        base_selection.set_map_name (cur_patient_ohip_num, "predicate_ohip_num")
                        run_query
                        if not base_selection.exhausted then
                                my_cursor.fill_in
                                base_selection.object_convert (medical_event)
                                base_selection.cursor_to_object
                                bind_out_medical_event
                                last_query_had_result := true
                        end
                        base_selection.reset_cursor (my_cursor)
                        base_selection.terminate
                        medical_database.session_control.commit
                        results_exhausted := true
                        db_query_in_prog := false
                        base_selection.unset_map_name ("predicate_ohip_num")
                ensure
                        post1: query_ok
                end;

        query_patient_complete_history is
                        --List a complete medical history for a patient.
                require
                        pre1: is_patient_in_ohip_repository (cur_patient_ohip_num)
                do
                        base_selection.set_map_name (cur_patient_ohip_num, "predicate_ohip_num")
```

```
                        select_string := select_patient_complete_history
                        run_query
                        base_selection.unset_map_name ("predicate_ohip_num")
                ensure
                        post1: query_ok
                end;

        query_all_patients is
                        -- List all patients by name and OHIP number.
                do
                        select_string := select_all_patients
                        run_query
                ensure
                        post1: query_ok
                end;

        next_medical_event is
                        -- fetch the next medical_event in a query of
                        -- the medical event table.
                require
                        pre1: db_query_in_prog
                do
                        if not base_selection.exhausted then
                                last_query_had_result := true
                                my_cursor.fill_in
                                base_selection.object_convert (medical_event)
                                base_selection.cursor_to_object
                                bind_out_medical_event
                                base_selection.next
                                if fetch_once then
                                        medical_database.session_control.commit
                                        fetch_once := false
                                end
                        else
                                results_exhausted := true
                                db_query_in_prog := false
                        end
                ensure
                        post1: not results_exhausted implies db_query_in_prog;
                        post2: not results_exhausted implies is_valid_event (medical_event)
                end;

        next_ohip_user is
                        -- fetch the next ohip user in a query of ohip
                        -- users table.
                require
                        pre1: db_query_in_prog
                do
                        if not base_selection.exhausted then
                                last_query_had_result := true
                                my_cursor.fill_in
                                base_selection.object_convert (ohip_user)
                                base_selection.cursor_to_object
                                bind_out_ohip_user
                                base_selection.next
                        else
                                results_exhausted := true
                                db_query_in_prog := false
                        end
                ensure
                        post1: not results_exhausted implies db_query_in_prog;
                        post2: not results_exhausted implies is_valid_ohip_user_obj (ohip_user)
                end;

        is_valid_ohip_user_obj (usr: OHIP_USER): BOOLEAN is
                        -- verify that usr obj reprsents a reasonable
                        -- OHIP_USER.
                require
                        usr /= void
                local
                        is_ok: BOOLEAN
                do
```

```
                        if usr.given_names.count > 1 and usr.surname.count > 1 and usr.patient_ohip_num > 0 and
usr.mom_ohip_num >= 0 and usr.dad_ohip_num >= 0 and usr.date_of_birth /= void and usr.gender > 0 and usr.gender <=
gender_map.max_enumerator then
                                is_ok := true
                        end;
                        Result := is_ok
                end;

        is_valid_event (event: MEDICAL_EVENT): BOOLEAN is
                        -- check if medicalevent data
                        -- we've bound out is reasonable
                require
                        event /= void
                local
                        is_ok: BOOLEAN
                do
                        if event.medical_specialty <= specialty_map.max_enumerator and event.practitioner_name.count >
0 and event.practitioner_type <= practitioner_map.max_enumerator and event.absolute_date /= void and
event.hospital_name.count > 0 and event.event_type <= event_map.max_enumerator then
                                is_ok := true
                        end;
                        Result := is_ok
                end;

feature {NONE}

        fetch_once: BOOLEAN;

        base_selection: DB_SELECTION;
                        -- selection class

        event_repository: DB_REPOSITORY;
                        -- Eiffel repositoryy (interface to relational table)

        ohip_repository: DB_REPOSITORY;
                        -- Eiffel repositoryy (interface to relational table)

        store: DB_STORE;
                        -- Store object for DB INSERT

        my_cursor: DB_RESULT;
                        -- Cursor for DB selects

        ohip_user: OHIP_USER;
                        -- Mapping class for repository operations with the OHIP_USERS table

        medical_event: MEDICAL_EVENT;
                        -- Mapping class for repository operations with the MEDICAL_EVENTS table

        cur_patient_ohip_num: INTEGER;
                        -- OHIP # of current patient

        select_string: STRING;
                        -- string containing SQL syntax for selects

        date_time_string: STRING;
                        -- Date and time in external string format

        medical_database: MEDICAL_DATABASE;
                        -- A high level database class for DB (not table) operations

        run_query is
                        -- This command executes a query specified
                        -- in the "select_string" string.
                require
                        pre1: db_query_in_prog = false
                do
                        last_query_had_result := false
                        query_ok := false
                        results_exhausted := true
                        base_selection.reset_cursor (my_cursor)
                        base_selection.terminate
                        base_selection.object_convert (ohip_repository)
                        base_selection.query (select_string)
```

```
            if base_selection.is_ok then
                query_ok := true
                db_query_in_prog := true
                base_selection.reset_cursor (my_cursor)
                results_exhausted := false
            end
        ensure
            post1: not results_exhausted implies db_query_in_prog;
            post2: not results_exhausted implies query_ok
        end;

    bind_out_medical_event is
            -- move values in medical_event to externally viewable
            -- features for the class. Note, clients of this feature are
            -- expected to validate the bound-out data after bindout.
            -- this routine intentionally does a "blind" bind out i.e.
            -- it does not check for correctness of the data.Thereforem, the
            -- assertions are are intentionally weak.
        require
            medical_event /= void
        do
            complaint := clone (medical_event.complaint)
            practitioner_name := clone (medical_event.practitioner_name)
            absolute_date := clone (medical_event.absolute_date)
            hospital_name := clone (medical_event.hospital_name)
            final_diagnosis := clone (medical_event.final_diagnosis)
            comment := clone (medical_event.comment)
            event_map.set_event_from_enum (medical_event.event_type)
            practitioner_map.set_practitioner_from_enum (medical_event.practitioner_type)
            specialty_map.set_specialty_from_enum (medical_event.medical_specialty)
        end;

    bind_out_ohip_user is
            -- move values in ohip_user to externally viewable
            -- features for the class. Note, clients of this feature are
            -- expected to validate the bound-out data after bindout.
            -- this routine intentionally does a "blind" bind-out i.e.
            -- it does not check for correctness of the data. Therefore
            -- the assertions are intentionally weak.
        do
            given_names := clone (ohip_user.given_names)
            surname := clone (ohip_user.surname)
            mom_ohip_num := ohip_user.mom_ohip_num
            dad_ohip_num := ohip_user.dad_ohip_num
            date_of_birth := clone (ohip_user.date_of_birth)
            patient_ohip_num := ohip_user.patient_ohip_num
            gender_map.set_gender_from_enum (ohip_user.gender)
        end;

    Select_all_patients: STRING is "select * from %"ohip_users%"  order by %"SURNAME%", %"GIVEN_NAMES%"";
            -- select string for querying all known OHIP users

    Select_patient_complete_history: STRING is "select * from %"medical_events%" where PATIENT_OHIP_NUM =
:predicate_ohip_num  order by ABSOLUTE_DATE";
            -- select string for querying the complete medical history of a patient

    Select_patient_history_by_specialty: STRING is "select * from %"medical_events%" where PATIENT_OHIP_NUM =
:predicate_ohip_num and MEDICAL_SPECIALTY = :predicate_specialty_enum  order by ABSOLUTE_DATE";
            -- select string for querying a patient history n a given medical specialty

    Select_patient_diagnosis_history_by_specialty: STRING is "select * from %"medical_events%" where EVENT_TYPE =
3 and PATIENT_OHIP_NUM = :predicate_ohip_num and MEDICAL_SPECIALTY = :predicate_specialty_enum  order by
ABSOLUTE_DATE";
            -- select string for querying a patient history n a given medical specialty

    Select_patient_last_visit: STRING is "select * from %"medical_events%" where PATIENT_OHIP_NUM =
:predicate_ohip_num and EVENT_TYPE = 1 order by ABSOLUTE_DATE DESC";
            -- select string for querying a patient's last hospital visit

    Select_patient_diagnosis_history: STRING is "select * from %"medical_events%" where EVENT_TYPE = 3 and
PATIENT_OHIP_NUM = :predicate_ohip_num  order by ABSOLUTE_DATE";
            -- select string for querying a patient's medical history, diagnosis only
        Select_offspring_from_ohip_repository: STRING is "select * from %"ohip_users%" where MOM_OHIP_NUM =
:predicate_ohip_num or DAD_OHIP_NUM = :predicate_ohip_num order by DATE_OF_BIRTH ";
                       -- select string for querying all of a patient's children
                       -- "select * from %"ohip_users%" where DAD_OHIP_NUM = :predicate_ohip_num order by
DATE_OF_BIRTH "

        Select_siblings_from_ohip_repository: STRING is "select * from %"ohip_users%" where MOM_OHIP_NUM =
:mom_predicate_ohip_num or DAD_OHIP_NUM = :dad_predicate_ohip_num order by %"DATE_OF_BIRTH%" ";
                       -- select string for querying all of a patient's brothers and sisters.
                       -- "select * from %"ohip_users%" where DAD_OHIP_NUM = :dad_predicate_ohip_num order by
%"DATE_OF_BIRTH%" "

        Select_user_from_ohip_repository: STRING is "select * from %"ohip_users%" where %"PATIENT_OHIP_NUM%" =
:predicate_ohip_num";
                       -- select string for querying a single user from known OHIP users

        Event_table_name: STRING is "medical_events";
                       -- Name of the relational tabl holding event data for
                       -- medical consultations

        Ohip_table_name: STRING is "ohip_users";
                       -- Name of the relational table holding data for
                       -- the list of known OHIP users.

        Default_ohip_num: INTEGER is 0;
                       -- ohip value in DB when unknown

        Invalid_ohip_num: INTEGER is 999999999;
                       -- guranteed not to match in DB.
                       -- event_repository.conforms( medical_event ) -- EiffelStore bug. Still not working
                       -- ohip_repository.conforms( ohip_user ) -- EiffelStore bug. Still not working

invariant

        inv1: medical_database /= void;
        inv2: tables_conform implies base_selection /= void;
        inv3: tables_conform implies store /= void;
        inv4: tables_conform implies medical_event /= void;
        inv5: tables_conform implies ohip_user /= void;
        inv6: tables_conform implies gender_map /= void;
        inv7: tables_conform implies event_map /= void;
        inv8: tables_conform implies practitioner_map /= void;
        inv9: tables_conform implies specialty_map /= void;
        inv10: tables_conform implies my_cursor /= void;

end -- class PATIENT_DATA_REPOSITORY
```

PRACTITIONER_MAP

```
Ancestor:

        ENUM_MAP

indexing
        description: "Types of practitioners.";
        author: "Sam Lightstone";
        date: "$Date: $";
        revision: "$Revision: $"

class PRACTITIONER_MAP

inherit
        ENUM_MAP

create
        make

feature

        set_doctor is
                        -- sets the prectitioner type to doctor
                do
                        enum := enum_doctor
                        string := doctor_string
                        is_set := true
                ensure
                        enum = enum_doctor;
                        string = doctor_string;
                        is_set = true
                end;

        set_nurse is
                        -- sets the practitioner type to nurese
                do
                        enum := enum_nurse
                        string := nurse_string
                        is_set := true
                ensure
                        enum = enum_nurse;
                        string = nurse_string;
                        is_set = true
                end;

        set_practitioner_from_enum (enum_in: INTEGER) is
                        -- sets the enumerated map given an enumerator.
                        -- this will do a sanity check ont he enum,
                        -- set the human readable format string, and mark the
                        -- object as being set ("is_set" = TRUE)
                require
                        enum_in > 0;
                        enum_in <= max_enumerator
                do
                        if enum_in = enum_doctor then
                                set_doctor
                        else
                                set_nurse
                        end
                ensure
                        enum = enum_in;
                        is_set = true
                end;

        Max_enumerator: INTEGER is 2;
                        -- max enumerator allowed in this class.

feature {NONE}

        Enum_doctor: INTEGER is 1;
                        -- enum value for doctors
        Enum_nurse: INTEGER is 2;
                        -- enum value for nurses

        Nurse_string: STRING is "NURSE";
                        -- string representation for nurses

        Doctor_string: STRING is "DOCTOR";
                        -- string representation for doctors

invariant

        invariant_clause: enum <= max_enumerator;

end -- class PRACTITIONER_MAP
```

QUERY_PANEL

```
Ancestor:

        INTERFACE


indexing

        description: "UI panel, provides selections for patient queiries.";

        author: "Sam Lightstone";

        date: "$Date: $";

        revision: "$Revision: $"


class QUERY_PANEL

inherit
        INTERFACE

create
        make

feature -- Initialization

        make (repository_in: PATIENT_DATA_REPOSITORY) is
                        -- Creation routine
                do
                        make_interface (repository_in)
                        create specialty_enum.make
                ensure
                        post1: specialty_enum /= void
                end;

feature {NONE} -- Implementation

        specialty_enum: MED_SPECIALTY_MAP;

        current_ohip_num: INTEGER;

        process_panel is
                        -- Routine to run the panel
                local
                        ohip_user_exists: BOOLEAN
                do
                        panel_selection := 0;
                        clear_screen;
                        display_panel_header;
                        io.putstring (" 1. Display a patient%'s last hospital visit%N");
                        io.putstring (" 2. Complete patient history%N");
                        io.putstring (" 3. Patient history by diagnosis%N");
                        io.putstring (" 4. Patient history by diagnosis for medical specialty%N");
                        io.putstring (" 5. Patient history by medical specialty%N");
                        io.putstring (" 6. Family history by diagnosis%N");
                        io.putstring (" 7. Family history by diagnosis for medical specialty%N");
                        io.putstring (" 8. Back to home%N");
                        min_panel_selection_value := 1;
                        max_panel_selection_value := 8;
                        min_selection_value := 1;
                        max_selection_value := 8;
                        request_panel_selection;
                        if panel_selection /= 8 then
                                current_ohip_num := request_ohip_num;
                ohip_user_exists := patient_data_repository.is_patient_in_ohip_repository
(current_ohip_num);
                if not ohip_user_exists then
                    invalid_ohip_user_msg;
                    io.putstring ("%N%N*****%N%N");
                    please_continue
                else
                    patient_data_repository.bind_out_last_ohip_user
                end
            end;
            if ohip_user_exists then
                if selection = 1 then
                    display_last_visit;
                    please_continue
                elseif selection = 2 then
                    display_complete_history;
                    please_continue
                elseif selection = 3 then
                    display_history_by_diagnosis (false);
                    please_continue
                elseif selection = 4 then
                    request_medical_specialty (specialty_enum);
                    display_history_for_specialty (false, true);
                    please_continue
                elseif selection = 5 then
                    request_medical_specialty (specialty_enum);
                    display_history_for_specialty (false, false);
                    please_continue
                elseif selection = 6 then
                    display_family_history_by_diagnosis;
                    please_continue
                elseif selection = 7 then
                    request_medical_specialty (specialty_enum);
                    display_family_history_for_specialty;
                    please_continue
                end
            end
        ensure then
            valid_panel_selection (panel_selection)
        end;

    display_last_visit is
            -- display info on this patients
            -- last regional hospital visit
        require
            patient_data_repository.db_query_in_prog = false
        local
            found_event: BOOLEAN
        do
            patient_data_repository.query_patient_last_visit;
            if patient_data_repository.last_query_had_result then
                clear_screen;
                display_panel_header;
                display_event (false)
            else
                io.putstring ("%NNo consultation history for this patient in the current database")
            end
        ensure
            patient_data_repository.db_query_in_prog = false
        end;

    display_complete_history is
            -- display the complete medical
            -- history for this patient
        require
            patient_data_repository.db_query_in_prog = false
        do
            patient_data_repository.query_patient_complete_history
            display_event_query_results (false)
        ensure
            patient_data_repository.db_query_in_prog = false
        end;

    display_history_by_diagnosis (hide_identity: BOOLEAN) is
```

```
                -- display a patient's medical
                -- history by diagnosis
        require
                patient_data_repository.db_query_in_prog = false
        do
                patient_data_repository.query_patient_diagnosis_history
                display_event_query_results (hide_identity)
        ensure
                patient_data_repository.db_query_in_prog = false
        end;

display_history_for_specialty (hide_identity, diagnosis_only: BOOLEAN) is
                -- display a patient's medical
                -- history in a given medical
                -- specialty
        require
                patient_data_repository.db_query_in_prog = false
        do
                patient_data_repository.query_patient_history_by_specialty (specialty_enum, diagnosis_only)
                display_event_query_results (hide_identity)
        ensure
                patient_data_repository.db_query_in_prog = false
        end;

display_family_history_by_diagnosis is
                -- display the patient histories
                -- (diagnosis only) for close
                -- relatives of a patient.
        require
                patient_data_repository.db_query_in_prog = false
        local
                i: INTEGER
        do
                build_list_of_family_ohip_numbers;
                from
                        i := 1
                until
                        i > num_family_members
                loop
                        set_current_ohip_user (family_ohip_nums @ i);
                        display_history_by_diagnosis (true);
                        i := i + 1
                end
        ensure
                patient_data_repository.db_query_in_prog = false
        end;

display_family_history_for_specialty is
                -- display the patient histories in a medical specialty
                -- (diagnosis only) for close relatives
                -- of a patient.
        require
                patient_data_repository.db_query_in_prog = false
        local
                i: INTEGER
        do
                build_list_of_family_ohip_numbers;
                from
                        i := 1
                until
                        i > num_family_members
                loop
                        set_current_ohip_user (family_ohip_nums @ i);
                        display_history_for_specialty (true, true);
                        i := i + 1
                end
        ensure
                patient_data_repository.db_query_in_prog = false
        end;

build_list_of_family_ohip_numbers is
                -- build a linked list of OHIP numbers for
                -- all close relatives of our current patient
        require
```

```
                patient_data_repository.db_query_in_prog = false
        local
                index: INTEGER
        do
                index := 1;
                if family_ohip_nums = void then
                        create family_ohip_nums.make (1, max_num_family_ohip_nums)
                end;
                if patient_data_repository.mom_ohip_num > 0 then
                        family_ohip_nums.put (patient_data_repository.mom_ohip_num, index);
                        index := index + 1
                end;
                if patient_data_repository.dad_ohip_num > 0 then
                        family_ohip_nums.put (patient_data_repository.dad_ohip_num, index);
                        index := index + 1
                end;
                patient_data_repository.query_children;
                patient_data_repository.next_ohip_user;
                from
                until
                        patient_data_repository.results_exhausted = true
                loop
                        patient_data_repository.bind_out_last_ohip_user;
                        family_ohip_nums.put (patient_data_repository.patient_ohip_num, index);
                        index := index + 1;
                        patient_data_repository.next_ohip_user
                end;
                set_current_ohip_user (current_ohip_num);
                patient_data_repository.query_siblings;
                patient_data_repository.next_ohip_user;
                from
                until
                        patient_data_repository.results_exhausted = true
                loop
                        patient_data_repository.bind_out_last_ohip_user;
                        if current_ohip_num /= patient_data_repository.patient_ohip_num then
                                family_ohip_nums.put (patient_data_repository.patient_ohip_num, index);
                                index := index + 1
                        end;
                        patient_data_repository.next_ohip_user
                end;
                set_current_ohip_user (current_ohip_num);
                patient_data_repository.bind_out_last_ohip_user;
                num_family_members := index - 1
        ensure
                patient_data_repository.db_query_in_prog = false
        end;

family_ohip_nums: ARRAY [INTEGER];
                -- constant for max number of family members.
                -- We will collect ohip numbers for parents, siblings
                -- and children. 200 is more than a safe upper bound for this.
                -- Contracts will protect us if the scope of collected
                -- ohip numbers changes.

Max_num_family_ohip_nums: INTEGER is 200;

num_family_members: INTEGER;
                -- the number of family members we find.

set_current_ohip_user (user_ohip_num: INTEGER) is
                -- set the current ohip user in the patient_data_repository
                -- to be "user_ohip_num" and bind out his attributes.
        require
                pre1: valid_ohip_num (user_ohip_num)
        local
                user_exists: BOOLEAN
        do
                user_exists := patient_data_repository.is_patient_in_ohip_repository (user_ohip_num);
                patient_data_repository.bind_out_last_ohip_user
        ensure
                post1: patient_data_repository.is_valid_ohip_user_data
        end;
```

```
end -- class QUERY_PANEL
```