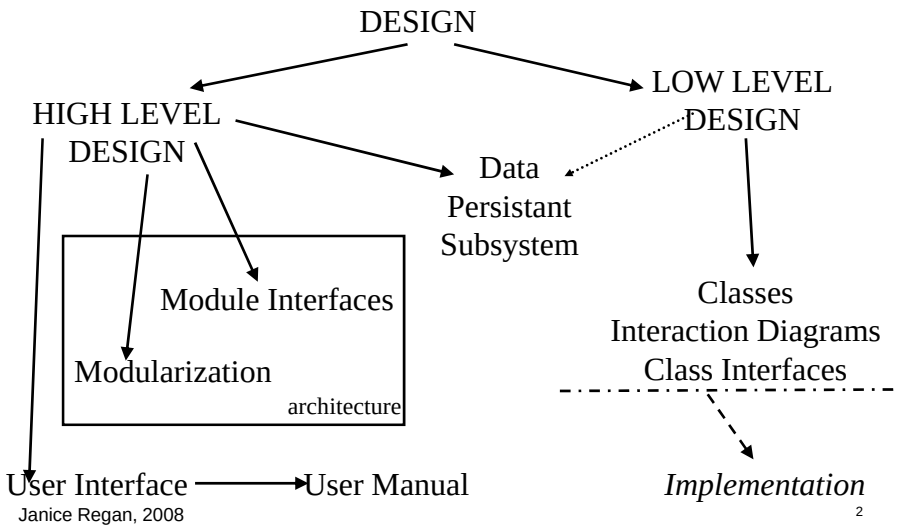
Sub-Phase

Low Level Design (cont)

Map of design phase



Janice Regan, 2008

Low Level Design

- Objective of Low Level Design
  - Refine representation (models) of software system to a level of detail that will allow resulting representation to be used as a blueprint for implementation and unit test planning phases

Interaction Diagrams

- Model dynamic aspects of the software system by specifying the interaction among objects to produce a particular behaviour
- For each use case
  - Show object interaction
  - Show how software system realizes a use case
- Help identify object operations (methods)

Interaction Diagrams

- Two types of interaction diagrams are defined in UML
  - Collaboration diagram: emphasizes the structural organization of objects that send and receive messages
  - Sequence diagram: emphasizes the time ordering of the messages passed between objects
- However, both diagrams are not computationally complete; they are not algorithms!
  -> They do not define the behaviour

Modeling at different phases

	Static	Dynamic
Requirements analysis	1. System Context Diagram 3. Class Diagram	2. Informal Scenarios 4. Use cases 5. Use cases Diagrams 6. Scenarios
High Level Design	7. Architecture (from 1,4,5,6)	8. Analyze data persistence 9. Refined use cases (from 4,7, 8) 10. Sequence Diagrams (from 9)
Low Level Design	11. Refined Class Diagram including attributes and methods (from 3, 12, 13)	12. Refined use cases and scenarios (from 9) 13. Refined Sequence Diagrams and collaboration diagrams (from 12)

UML Notational Elements of Collaboration Diagram

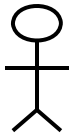
Link

Object

: Class

object : Class

Sequence of message



initiating actor

Message

[Condition] response := method(parameters)

Creating Collaboration Diagrams

- Summary:
  - 1 scenario (use case) per collaboration diagram
  - Identify participating classes (objects)
  - Determine messages to be sent (+ parameters) to carry out the behaviour by reading scenario/use case
    - Look at 1 scenario per use case
    - What changes for other scenarios?
  - Introduce solution for object persistence, if needed

Refined Scenario #1

- **Use Case Name:** CheckInResource (#7)
- **Scenario:** Student Patron Paul returns a book on time.
- **Preconditions:**
  - Librarian Eva has successfully gained access to the LMS.
  - LMS is ready to go (DB has been populated, network is up, and LMS has been initialized).
  - LMS screen with ***Check*** menu is displayed.

Refined Scenario #1

- **Use Case Name:** CheckInResource (#7)
- **Scenario:** Student Patron Paul returns a book on time.
- **Preconditions:**
  - Librarian Eva has successfully gained access to the LMS.
  - LMS is ready to go (DB has been populated, network is up, and LMS has been initialized).
  - LMS screen with ***Check*** menu is displayed.

Refined Scenario #1

- **Main flow of events:**

1. Student Patron Paul *comes* up to the librarian counter to return the Quantum Physics book he borrowed last week.
2. Eva the Librarian *chooses* ***CheckInResource*** option from the LMS screen by selecting the ***In*** command option under the ***Check*** menu.
3. A window representing a Check In Form is then *displayed.*

Refined Scenario #1

- **Main flow of events (cont):**

4. Eva takes the book Paul is handing to her and *types in* its Dewey call number in the appropriate text field then presses the "Accept" button to commit the entry.
5. The Dewey call number for the book was entered successfully and it *was a valid call number*, information about the Quantum Physics book and the borrowing patron is *retrieved* from the Database and displayed on the Check In screen.

Refined Scenario #1

- **Main flow of events (cont):**
  - Since Student Patron Paul is returning the Quantum Physics book before its due date, there is no overdue charge. Also, no one is currently requesting the Quantum Physics book.
  - LMS completes the check-in process by
    - changing the status of the book to “reshelve“,
    - canceling its “due date” and “date of loan“,
    - updating its “date of return“ to today,
    - Clearing the borrowing patron ID,
    - removing the Quantum Physics book from the Student Patron Paul’s list of borrowed resources.

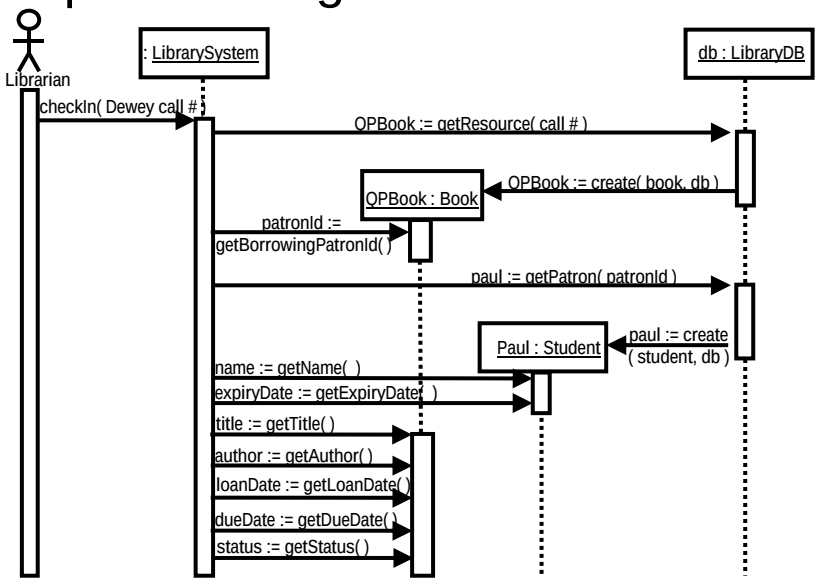
Refined Scenario #1

- **Main flow of events (cont):**
  - LMS updates the records for the Quantum Physics book and the borrowing Student Patron Paul in the Database.
  - LMS updates the screen showing the newly checked-in book along with the updated dates.
  - Eva verifies by looking at the screen that the book has been checked in properly, then presses the “Done” button.

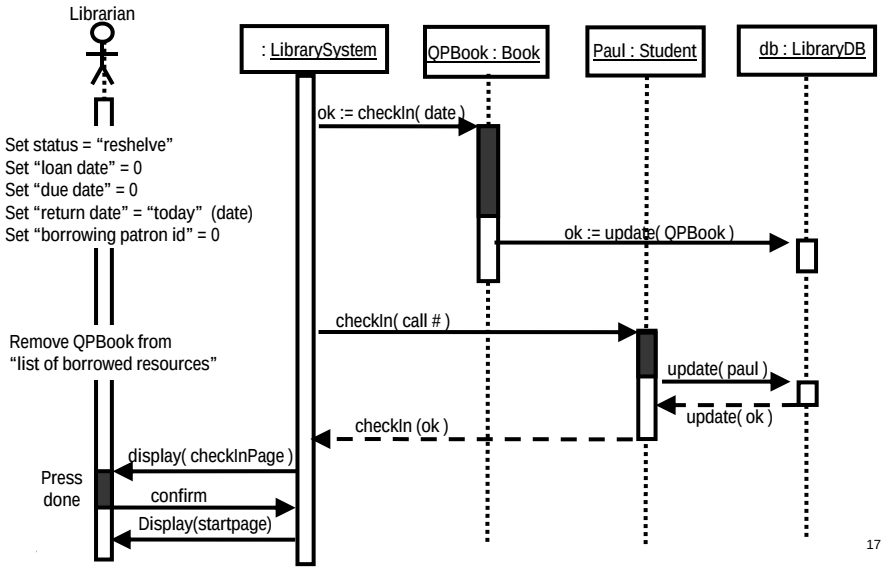
Refined Scenarios #1

- **Postconditions:**
  - Student Patron Paul's record is now showing that he is no longer borrowing the Quantum Physics book. The Quantum Physics book has now a status of "reshelve", today's date as a "date of return", "date of loan" has been cleared and so has the "due date".

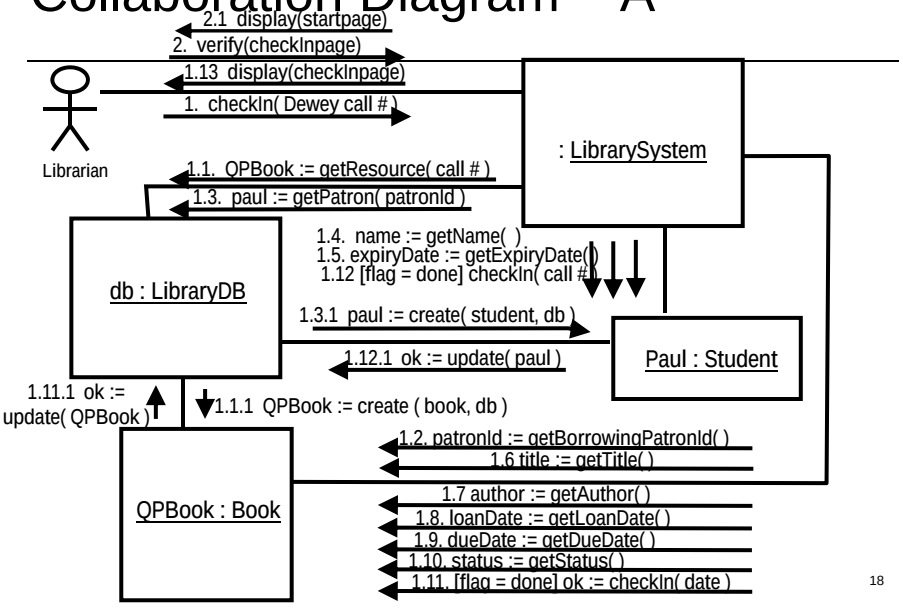
Sequence Diagram - 1



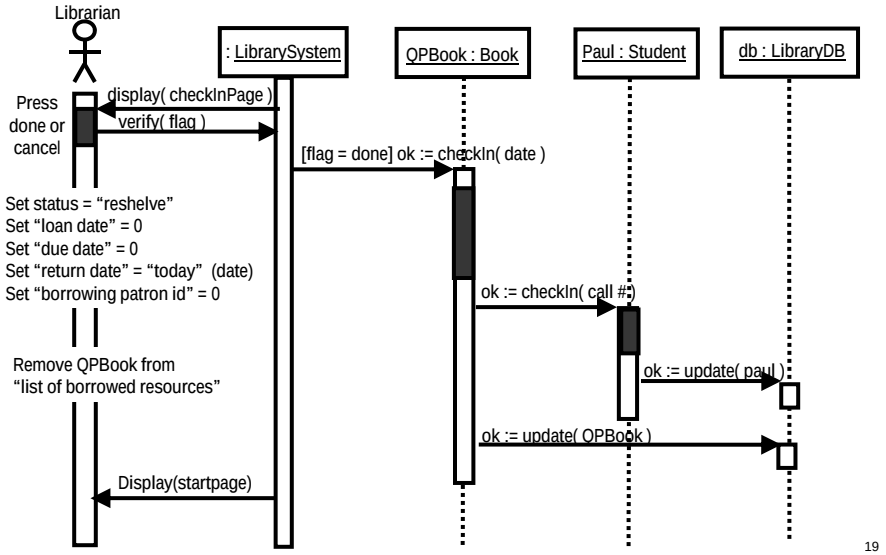
Sequence Diagram – 2 A



Collaboration Diagram – A

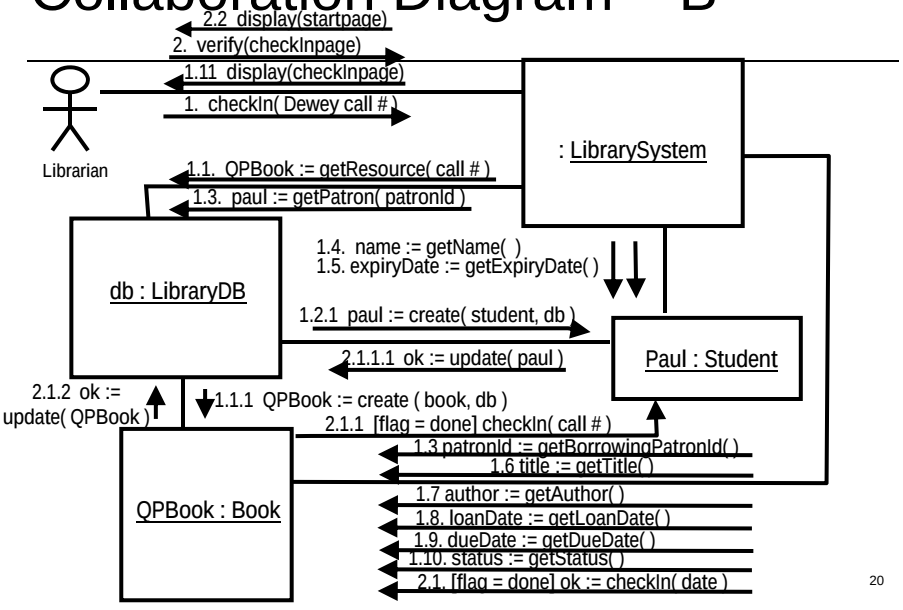


Sequence Diagram – 2 B



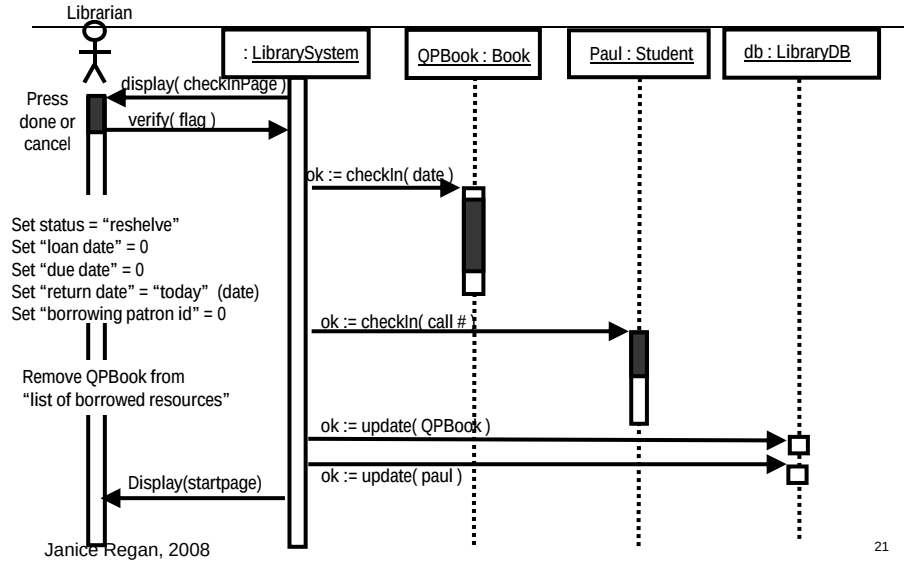
19

Collaboration Diagram – B



20

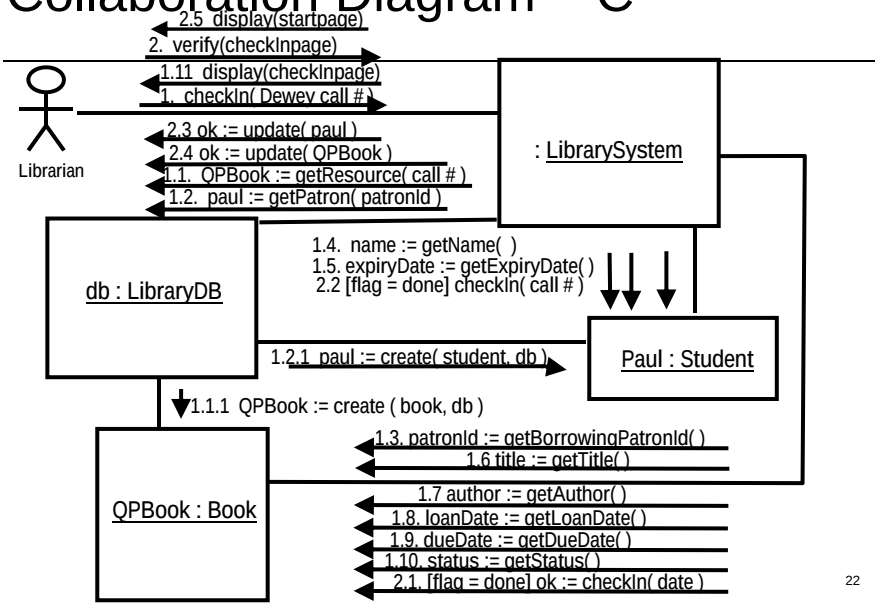
Sequence Diagram – 2 C



Janice Regan, 2008

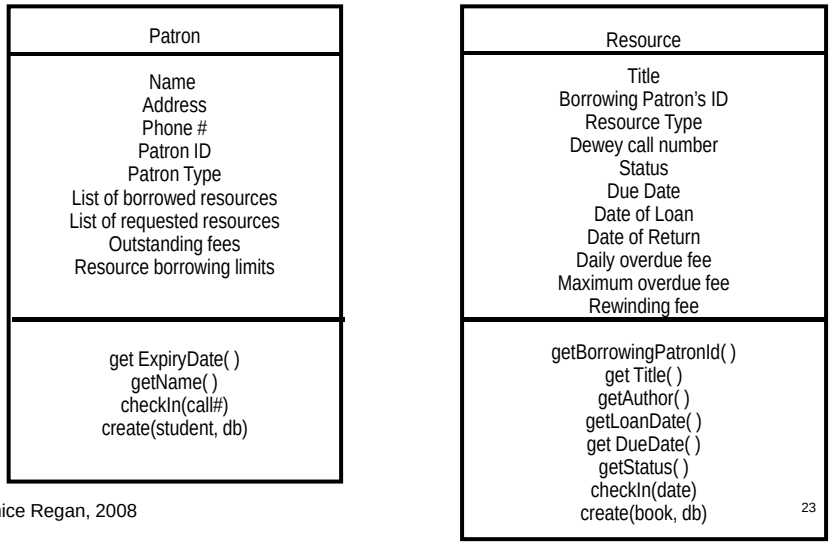
21

Collaboration Diagram – C



22

Patron and Resource class attributes



Discovering more methods

- To be sure we have all the methods for a class we need to consider all use cases that involve that class and all the scenarios associate with each of these use cases
  - Remember that although there is one class diagram for the system, there is at least one collaboration diagram or sequence diagram for each use case.
  - A collaboration diagram or sequence diagram can contain information for more than one scenario associated with a particular use case.
  - Usage of discovered methods must be consistent between all use cases and scenarios

Creating interaction diagrams

- Note that either a sequence diagram or a collaboration diagram can be derived directly from your use case and scenarios
- It is also possible to translate a collaboration diagram into a sequence diagram or a sequence diagram into a collaboration diagram
- The development of interaction diagrams represent an evolution of your model from abstract (class diagram) to more concrete
- Need to summarize the addition information back to the class diagram.
  - Add more detail to the methods and attributes on the class diagram