APÊNDICE A – FORMULÁRIO DE CONSENTIMENTO

Formulário de Consentimento

Estudo
O objetivo deste estudo é analisar o apoio proporcionado pelo *toolkit* na criação e inclusão de novos recursos. Dentre estes recursos podem ser citadas as visões (metáforas visuais) e os filtros. Nesta caracterização também será avaliada a efetividade dos roteiros de extensão disponibilizados com o *toolkit*.

Idade
Eu declaro ter mais de 18 anos de idade e concordo em participar de um estudo conduzido por Arleson Nunes Silva na Universidade Salvador (UNIFACS).

Procedimento
Este estudo será composto de quatro etapas. Em todas as etapas, os participantes deverão executar um conjunto de atividades e responder a um questionário relacionado às atividades realizadas. Eu entendo que, uma vez o experimento tenha terminado, os trabalhos que desenvolvi serão estudados visando a entender a eficiência dos procedimentos e as técnicas propostas.

Confidencialidade
Toda informação coletada neste estudo é confidencial, e meu nome não será divulgado. Da mesma forma, me comprometo a não comunicar os meus resultados enquanto não terminar o estudo, bem como manter sigilo dos materiais e documentos apresentados e que fazem parte do experimento.

Benefícios e liberdade de desistência
Eu entendo que os benefícios que receberei deste estudo são limitados ao aprendizado do material que é distribuído e apresentado. Eu entendo que sou livre para realizar perguntas a qualquer momento ou solicitar que qualquer informação relacionada a minha pessoa não seja incluída no estudo. Eu entendo que participo de livre e espontânea vontade com o único intuito de contribuir para o avanço e desenvolvimento de recursos de visualização.

Pesquisador responsável
Arleson Nunes Silva
Programa de Mestrado em Sistemas e Computação - UNIFACS

Professor responsável
Profº. Glauco de Figueiredo Carneiro
Programa de Mestrado em Sistemas e Computação - UNIFACS

Assinatura:_____ **Data:**_____

APÊNDICE B - FORMULÁRIO DE CARACTERIZAÇÃO

Formulário de Caracterização

1) Formação acadêmica

( ) Graduação ( ) Graduando
( ) Mestrado ( ) Mestrando
( ) Doutorado ( ) Doutorando
( ) Pós-Graduação ( ) Pós-Graduando

Ano de ingresso: _____ Ano de conclusão (ou previsão de conclusão): _____

2) Formação geral

2.1) Por favor, indique o grau de sua experiência nesta seção seguindo a escala de 5 pontos abaixo:

0 = nenhum
1 = estudei em aula ou em livro
2 = pratiquei em projetos em sala de aula
3 = usei em projetos pessoais
4 = usei em projetos na indústria

2.1.1) Visualização da Informação _____
2.1.2) Ambiente de Desenvolvimento de Software (ADS) Eclipse _____
2.1.3) Interface Gráfica do Usuário (GUI) em Java (ex: SWT; draw2D) _____

2.3) Por favor, explique sua resposta. Inclua o número de semestres ou número de anos de experiência relevante nos itens citados acima. (Ex.: "Eu trabalhei por 2 anos como programador Java na indústria")

_____
_____
_____

Obrigado por sua colaboração!

APÊNDICE C – ETAPAS DO ESTUDO

Etapas do Estudo

Etapa 1

O objetivo desta etapa é realizar a configuração dos artefatos necessários para criação da nova visão. Para esta finalidade será necessária a instalação e configuração do *toolkit.* Assim, utilizando os manuais de instalação e configuração, instale e configure o *toolkit* adequadamente. Além disto, também será necessária a realização do planejamento para a criação da visão (metáfora visual). Desta forma, a partir dos principais conceitos da metáfora visual a ser criada, defina o algoritmo a ser utilizado para a composição da visão.

Etapa 2

O objetivo desta etapa consiste na criação da visão (metáfora visual) a partir do *toolkit*. Assim, a partir dos recursos disponibilizados pelo *toolkit,* realize a criação de uma nova visão. Para esta finalidade utilize o ponto de extensão do *toolkit* mais adequado para criação de metáforas visuais. Em seguida, utilizando o algoritmo definido na etapa anterior, crie os componentes visuais para compor e gerar a interface da nova visão.

Etapa 3

O objetivo desta etapa é realizar a interação da nova visão criada com as estruturas de dados do *toolkit*. Para esta finalidade é necessário primeiramente acessar os dados e as informações contidos nas estruturas de dados. A partir destes dados, identificar os elementos da aplicação a serem representados na visão. Em seguida, realizar o mapeamento das propriedades ou atributos reais presentes nos elementos, para atributos visuais tais como formas, cores e posições na tela. Desta forma, a nova visão criada será utilizada para representação visual dos elementos da aplicação e de seus atributos.

Etapa 4

O objetivo desta etapa é realizar a interação da nova visão criada com os recursos de filtragem do *toolkit*. Para esta finalidade é necessária a identificação e distinção do conjunto de elementos filtrados e do conjunto de elementos não filtrados da aplicação. Assim, utilizando os recursos disponibilizados pelo *toolkit,* realize a identificação e distinção destes conjuntos de elementos. Em seguida, defina a configuração da visão para a representação visual destes conjuntos. Desta forma, utilize atributos visuais para diferenciar os elementos filtrados dos elementos não filtrados na visão.

APÊNDICE D – FORMULÁRIO DE OBSERVAÇÃO

Formulário de Observação

Tempo dedicado nesta etapa: _____

1) Quais os pontos positivos do uso do toolkit e dos roteiros de extensão para realização desta etapa?

_____
_____
_____

2) Quais as dificuldades encontradas para realização desta etapa?

_____
_____
_____

Observações gerais (sugestões e críticas):

_____
_____
_____
_____

APÊNDICE E – MANUAIS

Manual de Instalação

1. Requisitos Básicos

Para perfeita instalação do toolkit AIMV é necessário estar utilizando o Eclipse SDK. Se sua versão do Eclipse já tiver Eclipse SDK é só seguir os próximos passos, caso contrário, é necessária a sua instalação. Para instalar o Eclipse SDK no seu Eclipse utilize a opção Install New Sotware... no menu Help do seu Eclipse.

2. Instalando o Toolkit AIMV no Eclipse

1. Baixe e descompacte o arquivo aimv.zip **(Figura 38)**
2. Copie todos os arquivos .jar da pasta descompactada para a pasta plugins do Eclipse **(Figura 39)**
3. Reinicie a Eclipse IDE.

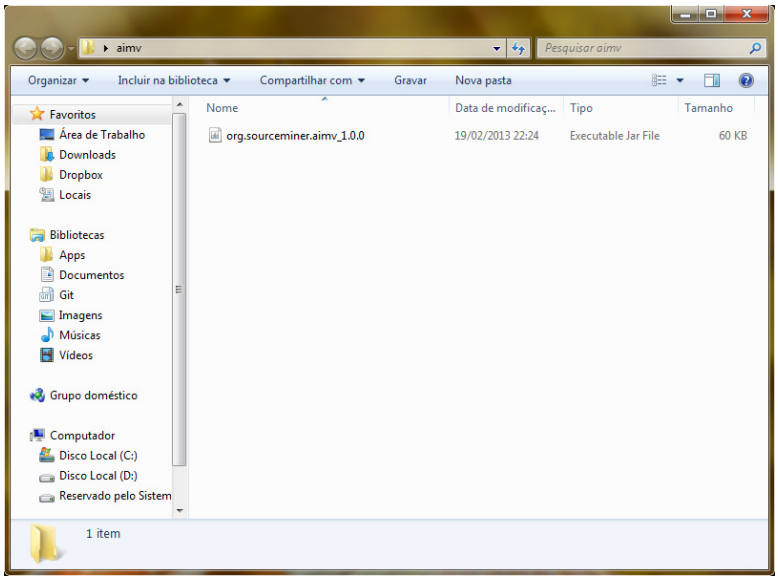

Figura 38: Arquivos .jar para Instalação do Toolkit

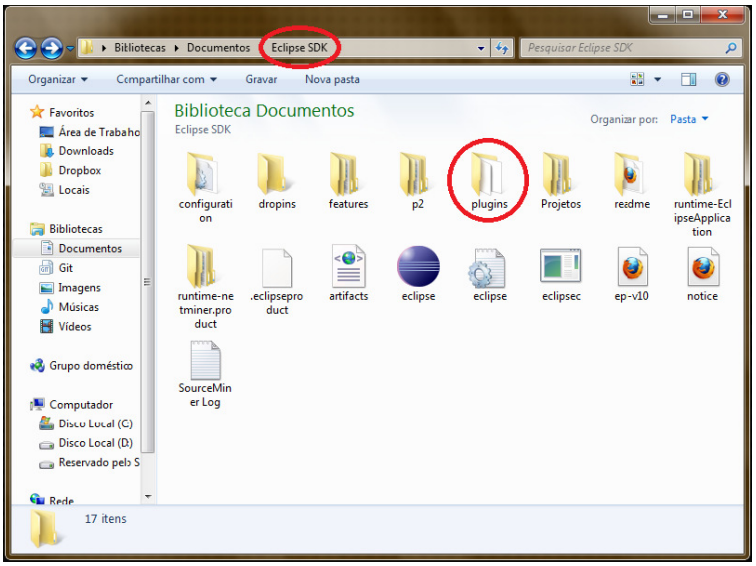

Figura 39: Pasta plugins do Eclipse

Manual de Configuração

1. Requisitos Básicos

Para utilizar os recursos e funcionalidades do toolkit AIMV é necessário antes a sua instalação. Além disso, é necessário estar utilizando o Eclipse SDK. Se o toolkit já está totalmente instalado no seu Eclipse é só seguir os próximos passos, caso contrário, utilize o **Manual de Instalação** para instalar o toolkit e em seguida siga os passos abaixo.

2. Criando um Novo Plug-in

No menu do Eclipse, selecione **File > New > Other** (ou pressione **Ctrl+N**), em seguida, selecione o assistente **Plug-in Development** e escolha **Plug-in Project.** Pressione Avançar. Na próxima tela, insira um nome de projeto. Pressione Avançar novamente. Na próxima tela, observe que o ID do plug-in corresponde ao nome do projeto. Usar o nome do projeto como o ID do plug-in minimiza as chances de esse plug-in entrar em conflito com o nome de outro plug-in. Clique em **Avançar** novamente. A próxima tela fornece a opção de criar manualmente o código de plug-in inicial ou executar um assistente de geração de código. Deixe o assistente de geração de código padrão, selecione **Hello World** e clique em **Avançar**.

3. Inserindo Dependências

Ao criar um novo plug-in será gerado um arquivo plugin.xml. Este arquivo contém informações descritivas que serão usadas pelo Eclipse para integrar o plug-in à estrutura. Na aba **Dependencies** do arquivo plugin.xml selecione a opção **Add..** e adicione o plug-in **org.sourceminer.aimv**, como mostra a Figura 40. Adicione também o plug-in **org.eclipse.core.runtime**.

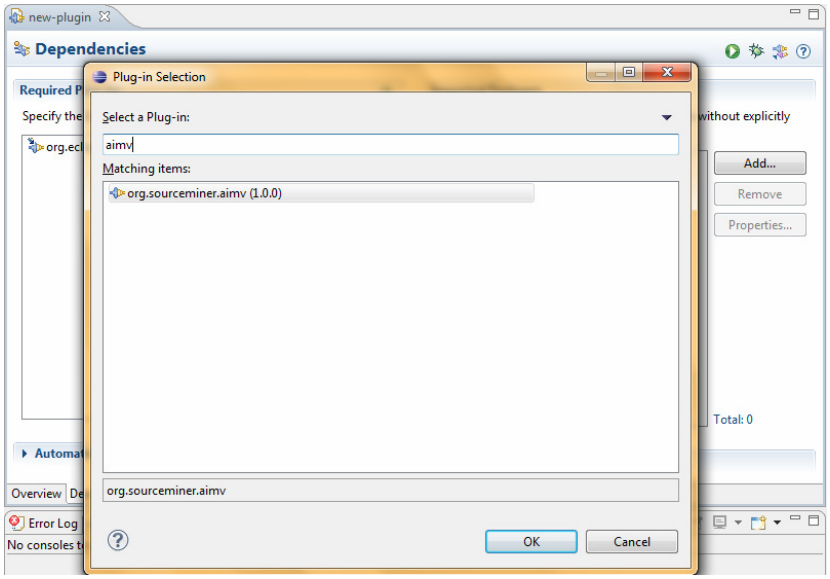

Figura 40: Inserindo Dependência do Plug-in aimv

Depois de inserir as dependências o plug-in criado poderá utilizar os pontos de extensão do toolkit. A partir destes pontos de extensão o usuário poderá criar os recursos e funcionalidades para o desenvolvimento dos seus ambientes interativos baseados em múltiplas visões (AIMVs).

APÊNDICE F – ROTEIROS DE EXTENSÃO

Roteiro 1 – Importação e Modelagem de Dados

1. Requisitos Básicos

Para utilizar os recursos de importação e modelagem de dados do toolkit é necessária antes a sua configuração. Se o toolkit já está totalmente instalado e configurado no seu Eclipse é só seguir os próximos passos, caso contrário, utilize o **Manual de Instalação** para instalação e o **Manual de Configuração** para configurar o toolkit e em seguida siga os passos abaixo.

2. Importando Dados

O toolkit disponibiliza uma ferramenta (Figura 41) que pode ser utilizada para importação de dados das aplicações AIMVs criadas a partir dele. Esta ferramenta permite a seleção de arquivos de diferentes formatos (Figura 42), que podem ser utilizados como fonte de dados para as aplicações AIMVs.

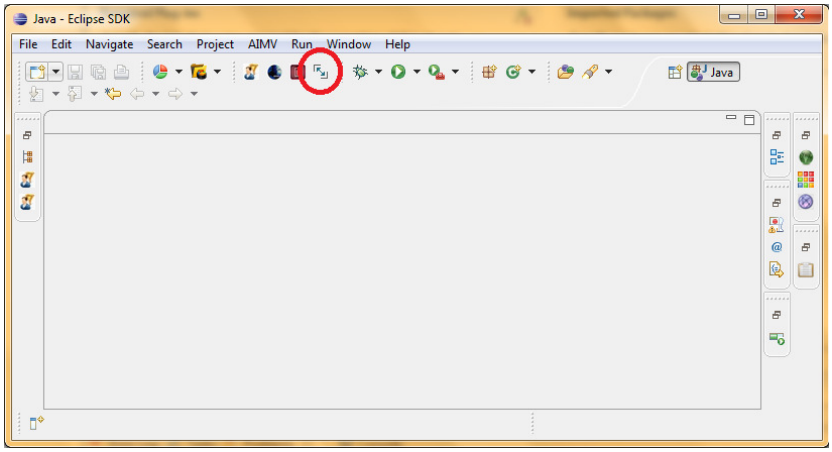

Figura 41: Ferramenta de Importação de Dados

Vale ressaltar, entretanto, que a importação de dados pode ser realizada através de outras ferramentas desenvolvidas por terceiros. Isto pode ser feito utilizando o **Roteiro 3 –**

Criação de Ferramentas. Para esta finalidade, deve ser utilizado o método **setFonte()**. Este método é disponibilizado pelo controlador central **AIMV** do toolkit e recebe como argumento um objeto do tipo **Object** que representa a fonte de dados. Na linguagem Java tudo é tratado como um objeto, logo o método **setFonte()** pode receber qualquer tipo de fonte de dados. A Figura 43 mostra um exemplo de utilização deste método. Neste caso, o usuário utiliza o método para passar um endereço de um banco de dados.

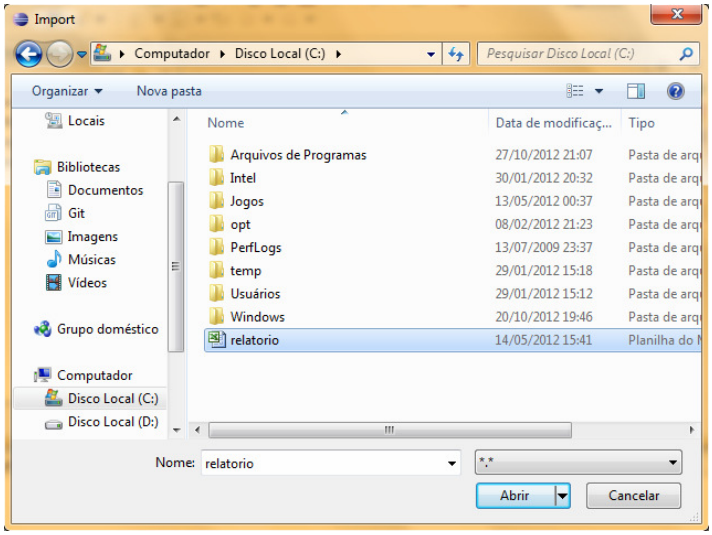

Figura 42: Seleção de Arquivo para Importação

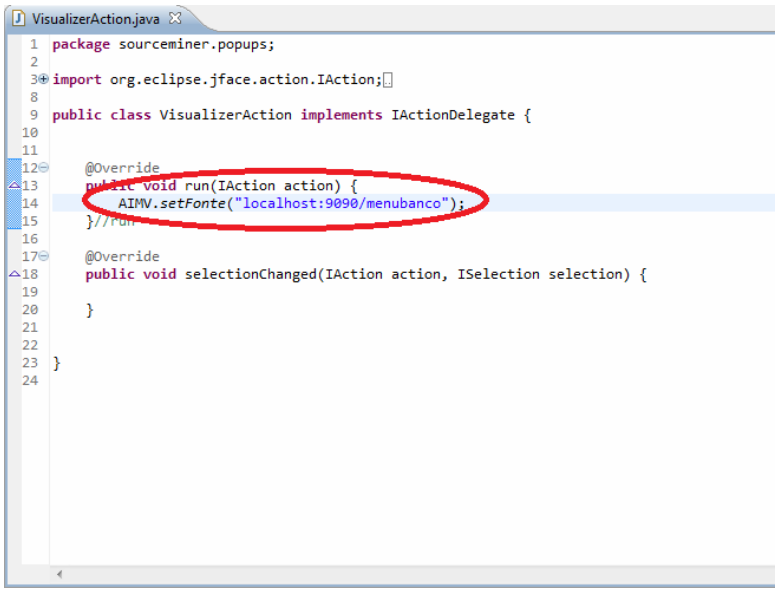

Figura 43: Utilizalação do Método setFonte()

3. Criando um Módulo de Importação

Os módulos de importação são os componentes responsáveis pela importação e modelagem dos dados nas estruturas de dados da aplicação. Eles recebem o objeto fonte dos dados e realiza um conjunto de transformações para modelar os dados na estruturas de dados da aplicação. Eles podem ser usados para receber diferentes objetos fonte de dados, de acordo com a aplicação do usuário. Para criar um módulo de importação é necessário utilizar o ponto de extensão **aimv.modules** disponibilizado pelo toolkit. Na aba **Extensions** do arquivo **plugin.xml** do seu plug-in selecione a opção **Add..** e adicione o ponto de extensão **aimv.modules** e clique em **Finish**, como mostra a Figura 44.

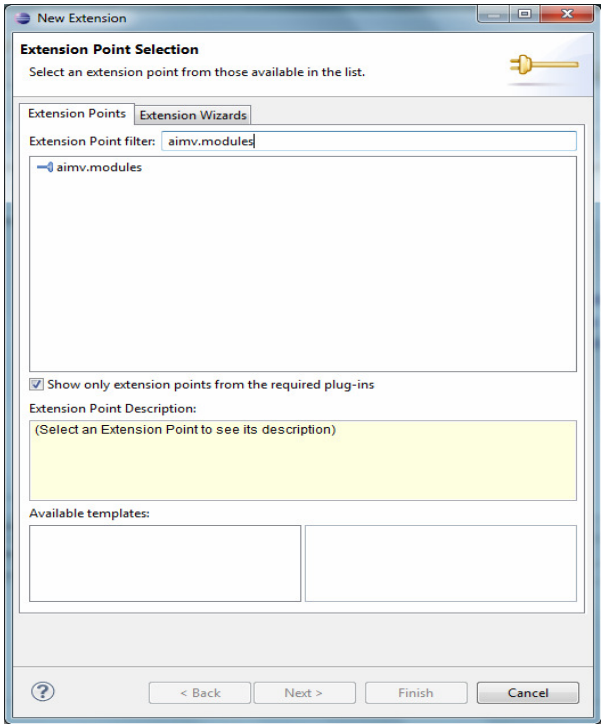

Figura 44: Adicionando Ponto de Extensão para Criação de Módulos de Importação

Depois de adicionar o ponto de extensão já é possível a criação de módulos. Para criar um novo módulo de importação, clique com o botão esquerdo no ponto de extensão adicionado e crie um novo módulo (Figura 45).

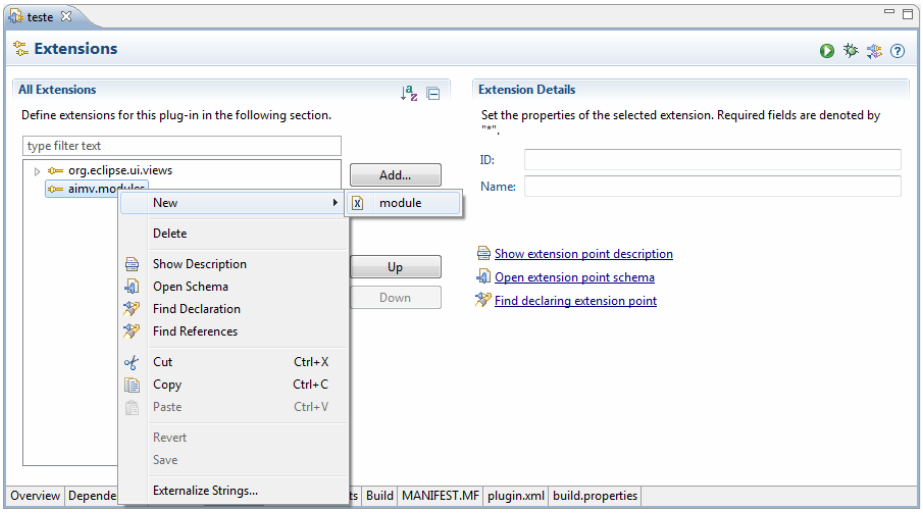

Figura 45: Criando Módulo de Importação

Após a criação do módulo de importação é necessário inserir algumas informações para identificar o módulo. Existem quatro atributos que devem ser preenchidos (Figura 46):

id usado para identificar e diferenciar o módulo dos outros módulos de importação

name usado para atribuir um nome para o módulo de importação

creation determina se é um módulo de criação

class usado para selecionar a classe que irá representar o módulo

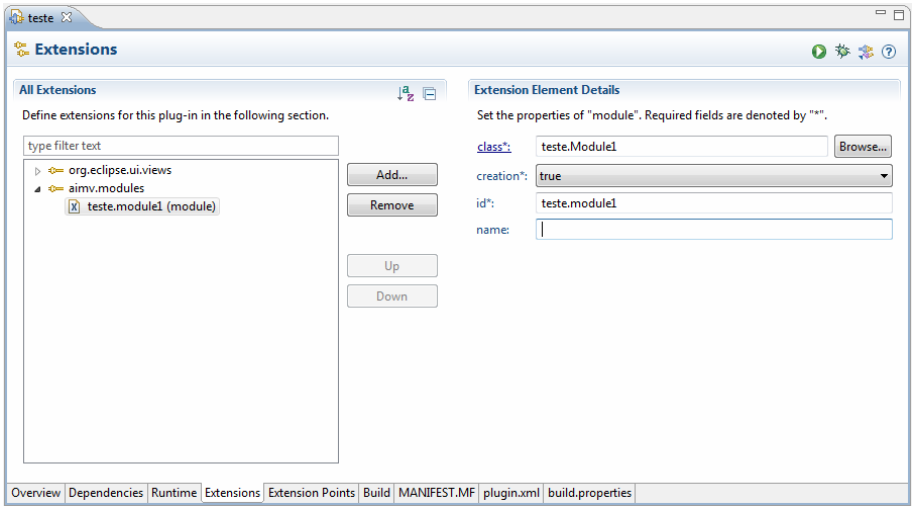

Figura 46: Inserindo informações no módulo de importação

O ponto de extensão para criação de módulos de importação disponibiliza uma superclasse chamada **aimv.modules.Module**. Esta superclasse fornece os métodos que devem ser implementados pelas classes que irão representar os módulos. Se a classe já estiver criada

então utilize a opção **Browse...** e selecione a classe, caso contrário clique no atributo **class**. Irá aparecer uma tela para criação da nova classe e o usuário pode preencher as informações necessárias da classe (Figura 47). A nova classe será criada com os métodos a serem implementados (Figura 48).

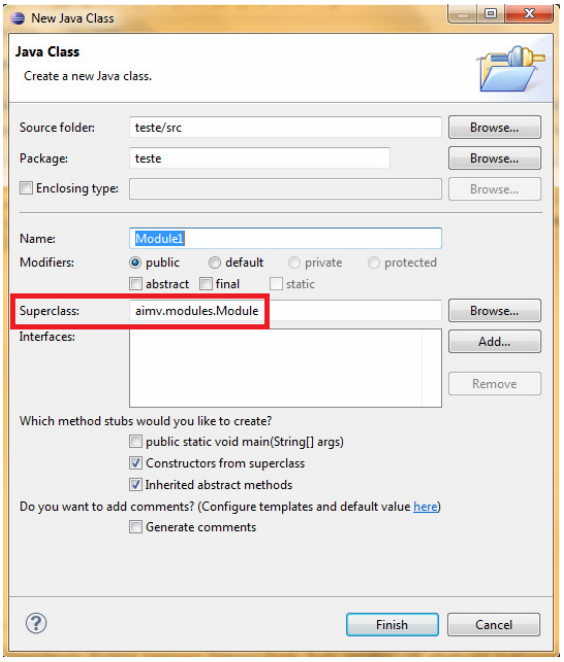

Figura 47: Criando Classe para Representar o Módulo de Importação

O objeto fonte de dados é armazenado no controlador central **AIMV**, como foi visto na seção anterior através do método **setFonte()**. Quando este método é acionado o controlador **AIMV** irá disponibilizar o objeto fonte de dados para todos os módulos de importação registrados na aplicação, através do método **start()**. Através deste método o usuário pode realizar a modelagem de dados utilizando as estruturas de dados disponibilizadas pelo toolkit.

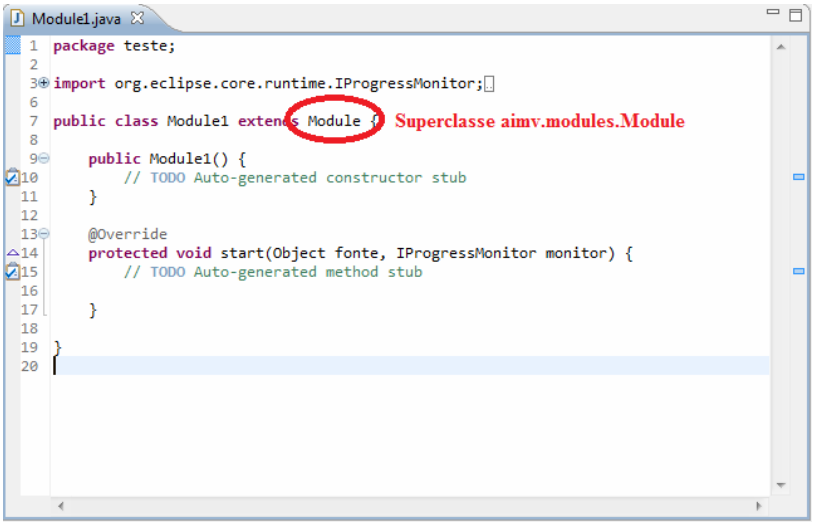

Figura 48: Exemplo de Classe que Representa um Módulo de Importação

4. Modelando Dados

As estruturas de dados são os componentes responsáveis pelo armazenamento dos dados tratados da fonte original pelos módulos de importação. Elas são utilizadas para facilitar e auxiliar o uso dos dados pelos demais componentes da aplicação.

O toolkit disponibiliza três tipos de estrutura de dados: **Group**, **Node** e **Relation**. A estrutura de dados **Node** pode ser utilizado para representar objetos ou entidades de um determinado domínio de dados. Ele contém um atributo denominado **ID** e uma lista de propriedades contendo as demais informações. O **ID** é único e é usado para diferenciar os objetos do tipo **Node**. Por exemplo, em um AIMV para representação visual de dados de redes sociais, pode utilizar o componente **Node** para representar as "pessoas" que participam da rede. Os atributos da entidade "pessoa", tais como, nome, idade e sexo podem ser armazenados como propriedades do **Node**. A estrutura **Relation** pode ser utilizada para relacionar duas entidades. Ela é criada a partir de dois objetos do tipo **Node**. Por exemplo, pode utilizar está estrutura para armazenar informações de amizade entre as entidades "pessoa" dentro de uma rede. Além disso, a estrutura **Relation** contém uma lista de propriedades, onde pode ser armazenadas informações das relações, como por exemplo, o tipo de amizade entre as entidades "pessoa" da rede. Depois de criados, os objetos do tipo **Relation** podem ser armazenados na lista de relações dos objetos do tipo **Node**. A estrutura **Group** pode ser utilizada para agrupar os objetos do tipo **Node**. Os objetos do tipo **Node** podem ser agrupados de acordo com determinada propriedade comum a estes objetos. Por

exemplo, pode agrupar os objetos do tipo **Node** que representam as entidades "pessoa" pela propriedade "sexo" comum a todos os objetos do tipo **Node** criados para aplicação.

A seguir será mostrado um exemplo de modelagem de dados utilizando um módulo de importação. Neste exemplo será utilizado um arquivo rede.txt (Figura 49) que contém informações de pessoas, tais como, a idade e o sexo. Este arquivo será importado e será disponibilizado para o módulo de importação criado especificamente para modelar os dados deste arquivo.

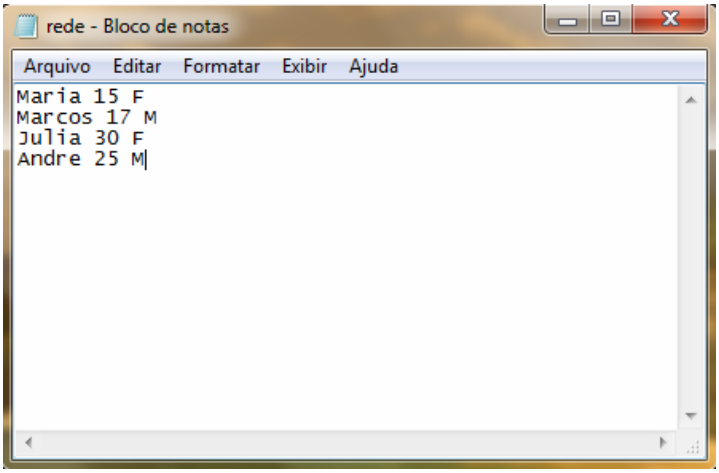

Figura 49: Exemplo de Entrada com Dados de uma Rede Social

O módulo de importação recebe o objeto fonte de dados, nesse caso, o arquivo rede.txt selecionado através da ferramenta de importação. Em seguida o arquivo é lido e as informações vão sendo armazenadas nas estruturas de dados, como mostra a Figura 50. Inicialmente são criados dois grupos através dos objetos do tipo **Group**. Esses grupos serão utilizados para agrupar as pessoas de acordo com o atributo sexo. Em seguida estes grupos são adicionados ao controlador de grupos **Nodes** através do método **addNode()** para poderem ser acessados pelos outros componentes da aplicação. Depois são criados objetos do tipo **Node** para representar cada entidade "pessoa" e seus atributos são armazenados como propriedades do **Node** através do método **setProperty()**. Por último, o objeto do tipo **Node** é adicionado ao seu respectivo grupo de acordo com o atributo "sexo" da pessoa.

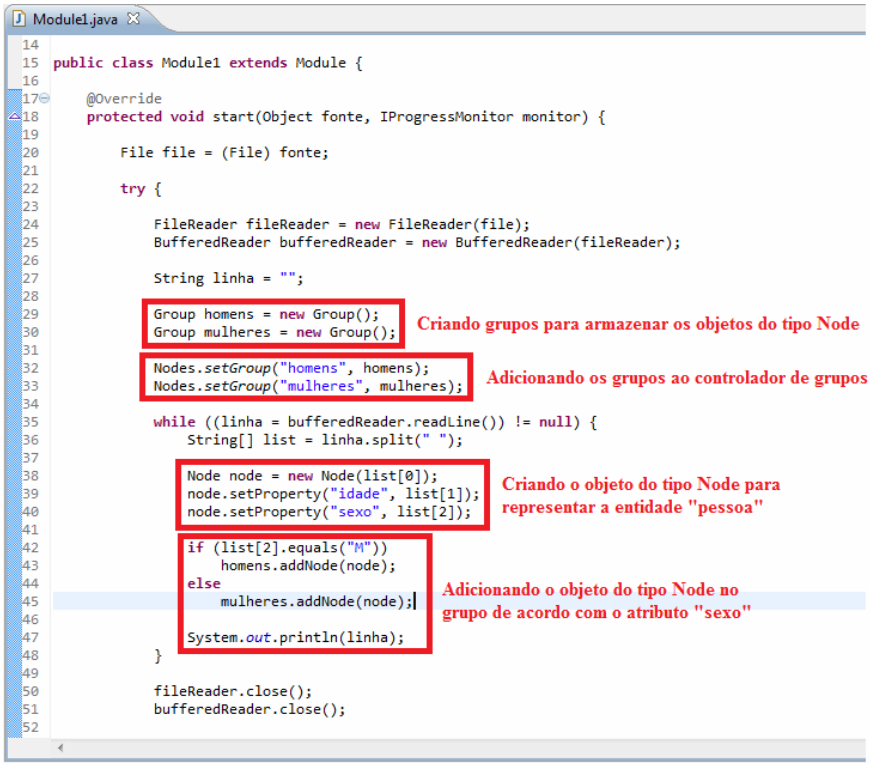

Figura 50: Modelagem Utilizando as Estruturas de Dados do Toolkit

Roteiro 2 – Criação e Aplicação de Filtros

1. Requisitos Básicos

Para utilizar os recursos de filtragem disponibilizados pelo toolkit AIMV é necessário antes a sua configuração. Se o toolkit já está totalmente instalado e configurado no seu Eclipse é só seguir os próximos passos, caso contrário, utilize o **Manual de Instalação** para instalação e o **Manual de Configuração** para configurar o toolkit e em seguida siga os passos abaixo.

2. Criação de Filtros

Os filtros são criados a partir do ponto de extensão de filtros **aimv.filters** disponibilizado pelo toolkit. Para adicionar este ponto de extensão, utilize a aba **Extensions** do arquivo **plugin.xml** do seu plug-in e selecione a opção **Add..,** em seguida, adicione o ponto de extensão **aimv.filters** e clique em **Finish**, como mostra a Figura 51.

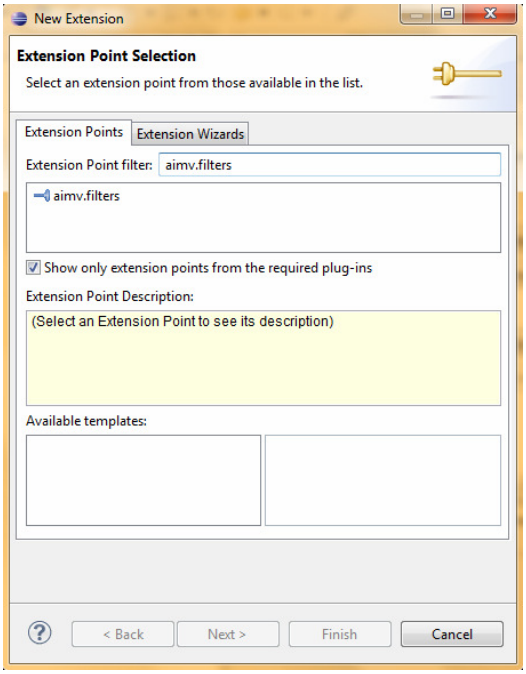

Figura 51: Adicionando Ponto de Extensão para Criação de Filtros

Depois de adicionar o ponto de extensão já é possível a criação de filtros. Para criar um novo filtro, clique com o botão esquerdo no ponto de extensão adicionado e selecione a opção **New > filter** (Figura 52).

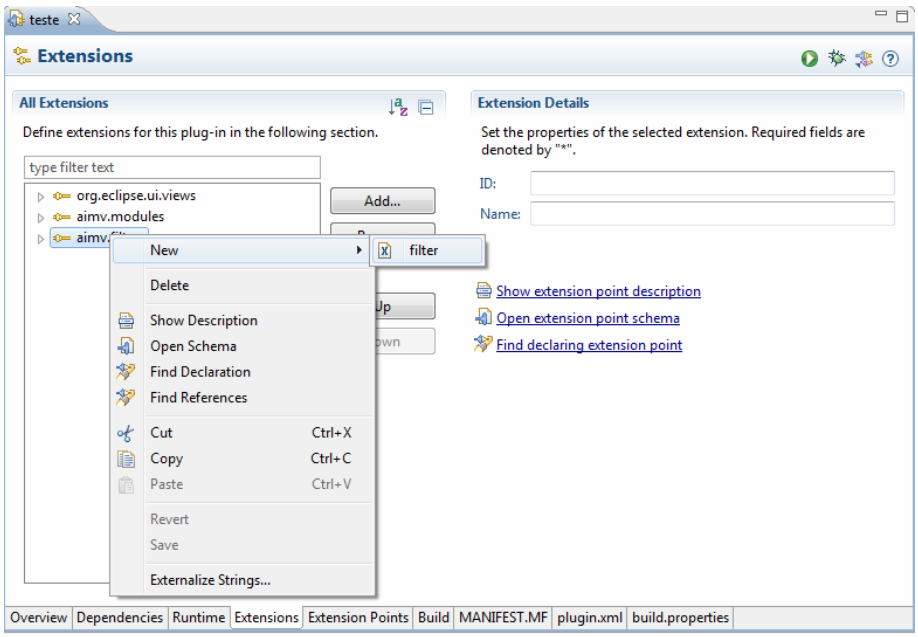

Figura 52: Criando Novo Filtro

Após a criação do filtro é necessário inserir algumas informações para sua identificação. Existem três atributos que são utilizados para caracterizar os filtros, como mostra a Figura 53:

id usado para identificar e diferenciar o filtro dos outros filtros;

name usado para atribuir um nome para o filtro;

class usado para selecionar a classe que irá representar o filtro.

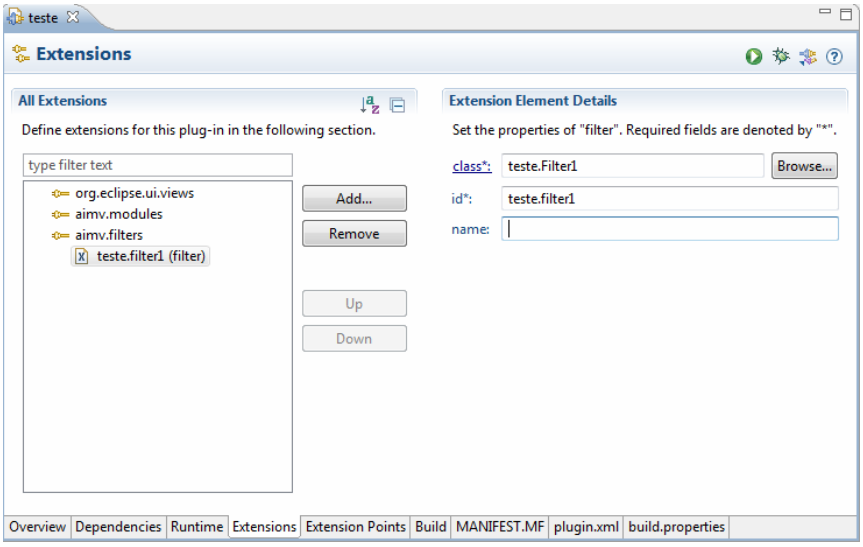

Figura 53: Inserindo Informações do Novo Filtro

O ponto de extensão de filtros utiliza como base para criação de novos filtros a classe **aimv.filters.Filter**. Esta classe fornece os métodos necessários para implementação das funcionalidades de cada filtro. Desta forma, ao criar um novo filtro é necessário criar uma classe que faça herança com a classe **aimv.filters.Filter**. Esta classe será utilizada para representar o filtro. Se a classe já estiver criada então utilize a opção **Browse...** e selecione a classe, caso contrário clique no atributo **class**. Irá aparecer uma tela para criação da nova classe e o usuário pode preencher as informações necessárias da classe (Figura 54).

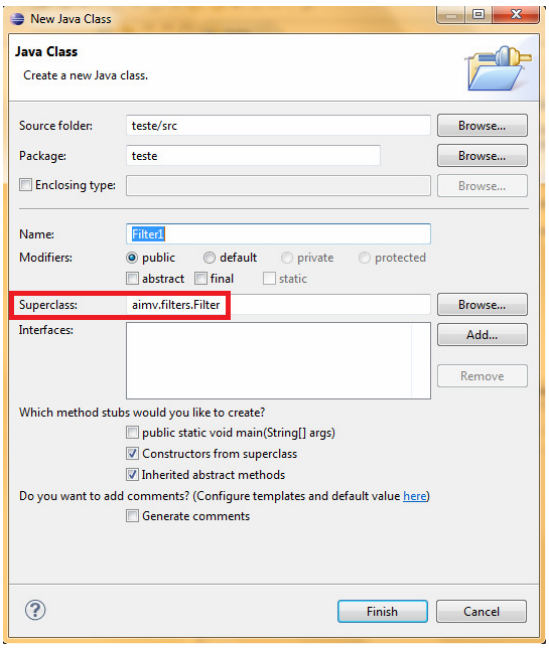

Figura 54: Criando Classe para Representar o Filtro

A classe será criada com os métodos a serem implementados, como mostra a Figura 55. Os métodos fornecidos pela classe **aimv.filters.Filter** são apresentados a seguir.

applyFilter(...): método utilizado para aplicação do filtro. Este método recebe uma lista de argumentos que são utilizados para aplicar o filtro nos elementos.

removeFilter(...): método utilizado para remoção do filtro. Este método recebe uma lista de argumentos que são utilizados para remover o filtro nos elementos.

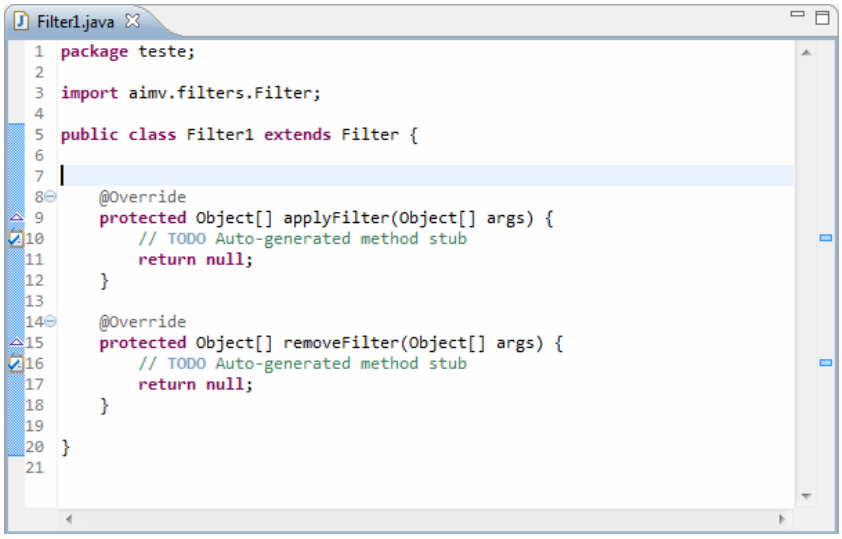

Figura 55: Exemplo de Classe para Representação de um Filtro

3. Criação de Visões de Filtragem

As visões de filtragem são criadas a partir do ponto de extensão **org.eclipse.ui.views**. Para adicionar este ponto de extensão, utilize a aba **Extensions** do arquivo **plugin.xml** do seu plug-in e selecione a opção **Add..,** em seguida, adicione o ponto de extensão **org.eclipse.ui.views** e clique em **Finish**, como mostra a Figura 56.

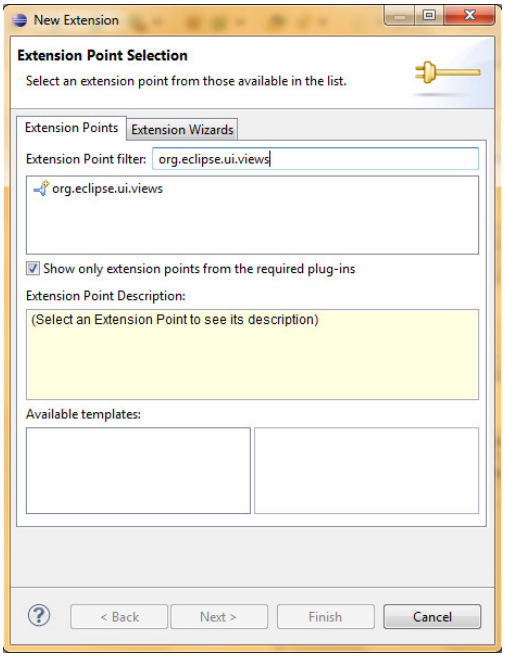

Figura 56: Adicionando Ponto de Extensão para Criação de Visões de Filtragem

Depois de adicionar o ponto de extensão já é possível a criação das visões de filtragem. Para criar uma nova visão de filtragem, clique com o botão esquerdo no ponto de extensão adicionado e selecione a opção **New > view** (Figura 57).

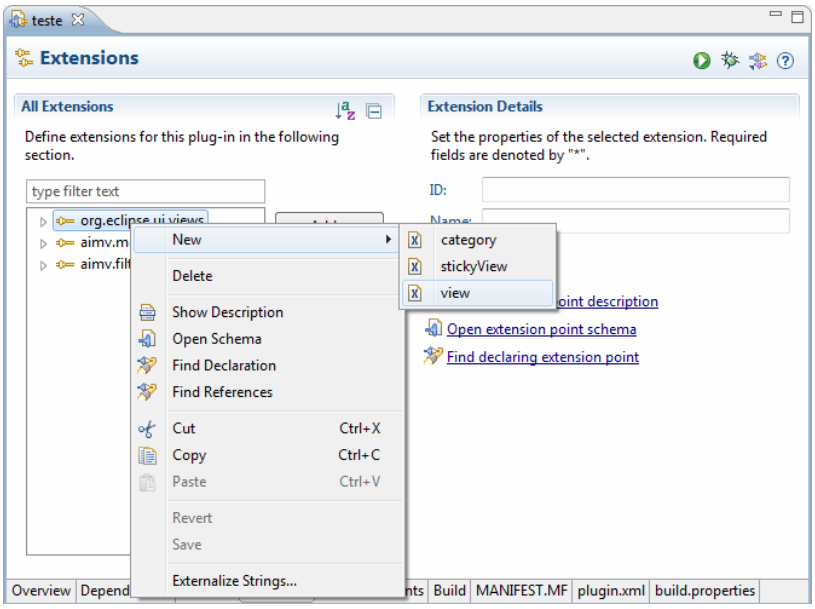

Figura 57: Criando Visão de Filtragem

Após a criação da visão é necessário inserir algumas informações para sua identificação. Os atributos que são utilizados para caracterizar as visões são ilustrados na Figura 58.

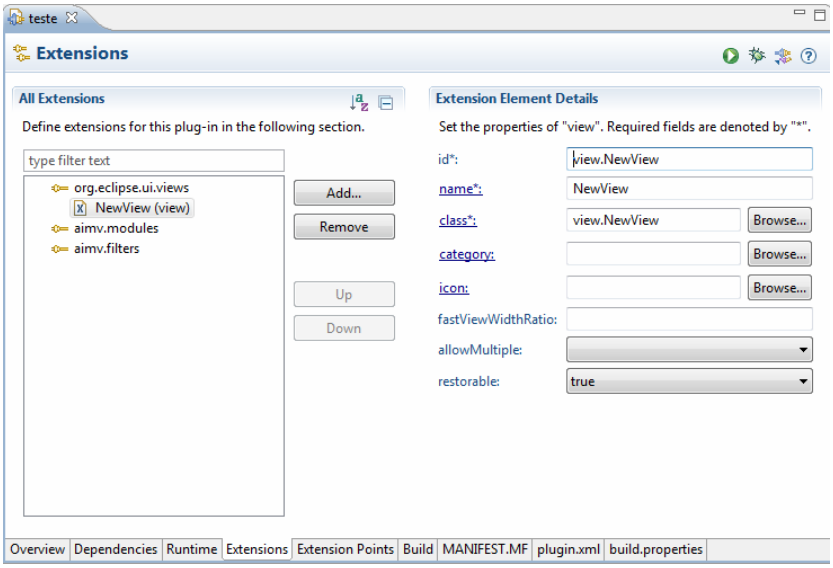

Figura 58: Inserindo Informações da Visão

A criação de visões de filtragem tem como base a classe **aimv.views.FilterView**. Esta classe fornece os métodos necessários para implementação das funcionalidades de cada visão de filtragem. Desta forma, ao criar uma nova visão de filtragem é necessário criar uma classe que faça herança com a classe **aimv.views.FilterView**. Esta classe será utilizada para representar a visão. Se a classe já estiver criada então utilize a opção **Browse...** e selecione a classe, caso contrário clique no atributo **class**. Irá aparecer uma tela para criação da nova classe e o usuário pode preencher as informações necessárias da classe, como mostra a Figura 59.

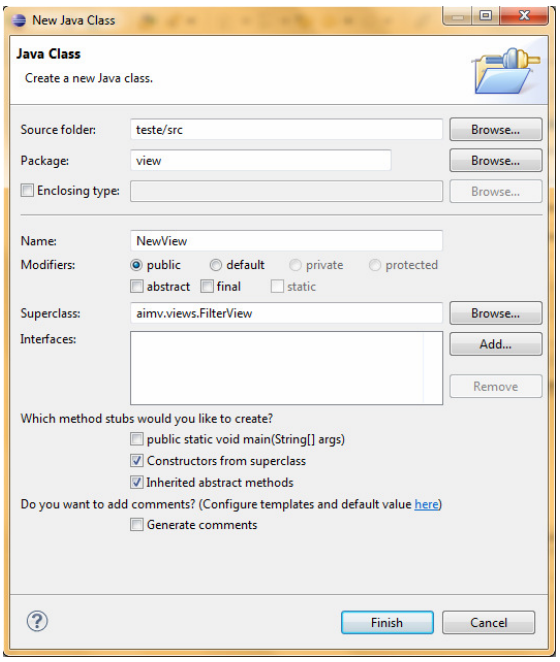

Figura 59: Criando Classe para Representar a Visão de Filtragem

A classe será criada com os métodos a serem implementados, como mostra a Figura 60. Os métodos fornecidos pela classe **aimv.views.FilterView** são apresentados a seguir.

createLayout(): método utilizado para criação dos componentes visuais da visão de filtragem.

open(): método utilizado para definir as ações a serem tomadas quando a visão de filtragem for aberta.

closed(): método utilizado para definir as ações a serem tomadas quando a visão de filtragem for fechada.

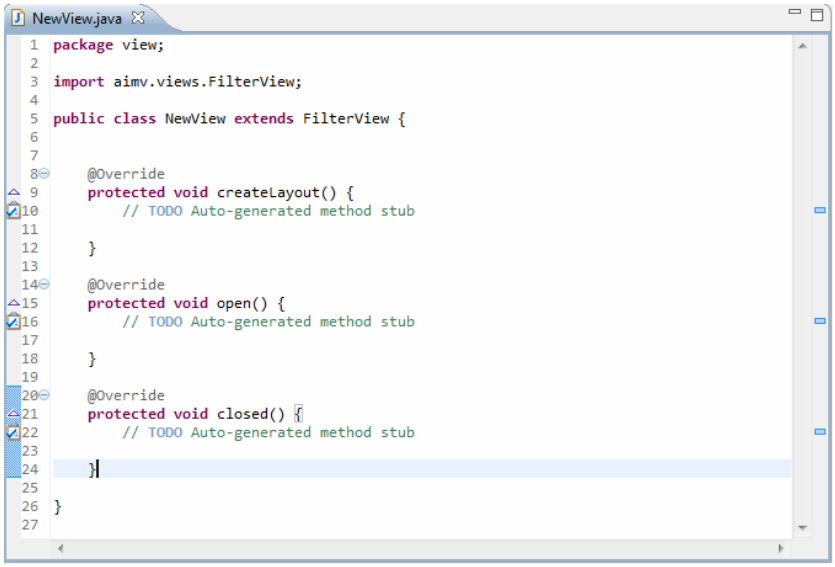

Figura 60: Exemplo de classe para Representação de uma Visão de Filtragem

4. Aplicação e Remoção de Filtros

Os filtros atuam diretamente nos elementos do tipo **Node** e **Relation** da aplicação. Eles são adicionados e removidos diretamente nestes elementos através dos métodos **addFilter()** e **removeFilter()**, como mostra a Figura 61. Cada elemento possui uma lista de filtros. Ao adicionar um filtro a um elemento, este filtro será adicionado a sua lista de filtros, da mesma forma, ao remover o filtro este será removido da lista. Se a lista de filtros de um elemento estiver vazia, indica que o elemento não está filtrado, caso contrário, indica que o elemento está filtrado. Os elementos filtrados podem ser identificados através do método **isFiltered()** presente em cada elemento.

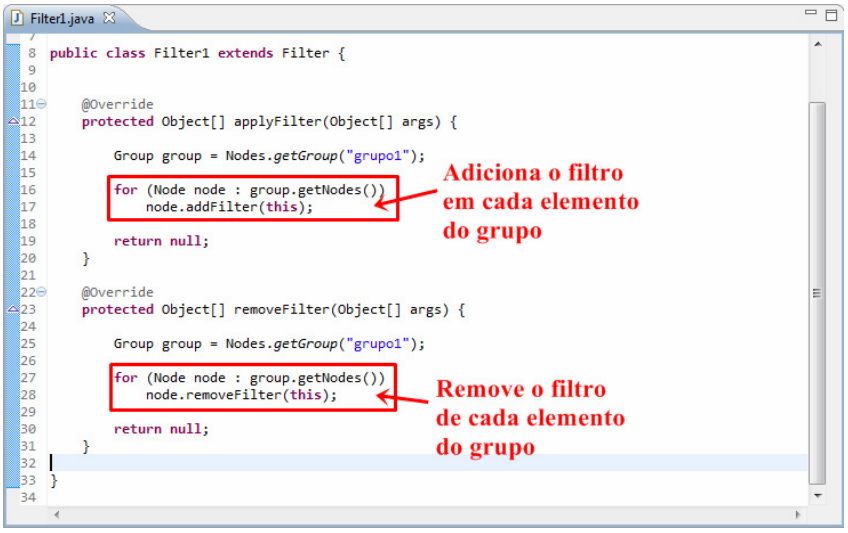

Figura 61: Aplicação e Remoção de um Filtro

Os filtros são acionados através das visões de filtragem. A Figura 62 mostra um exemplo de uma visão de filtragem. Nesta visão foram criados dois botões, um para aplicação e o outro para remoção do filtro. A Figura 63 mostra a interface da visão de filtragem criada. Ao selecionar os botões **Aplicar** e **Remover**, o filtro será acionado e aplicado nos elementos da aplicação.

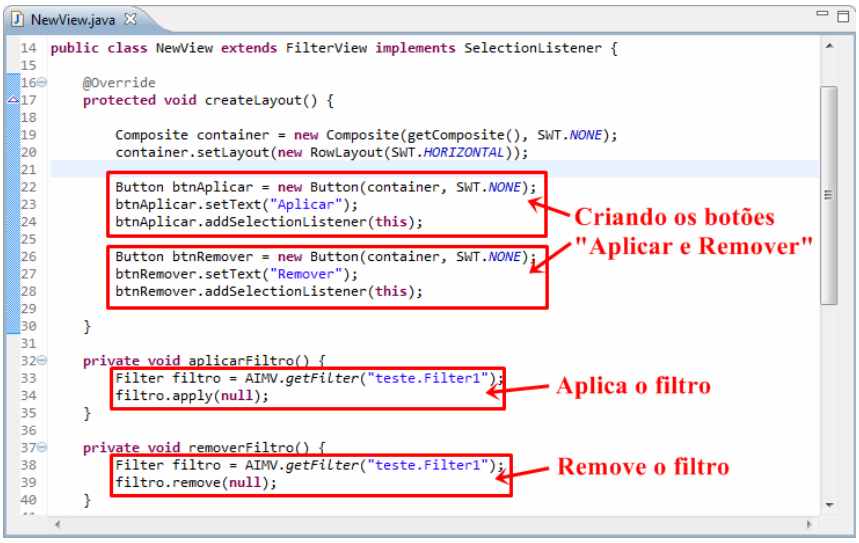

Figura 62: Visão de Filtragem

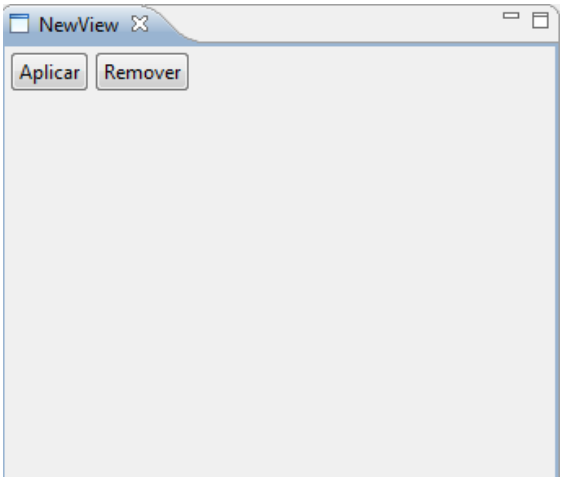

Figura 63: Interface da Visão de Filtragem

Roteiro 3 – Criação de Ferramentas

1. Requisitos Básicos

Para utilizar os recursos para criação de ferramentas disponibilizados pelo toolkit AIMV é necessário antes a sua configuração. Se o toolkit já está totalmente instalado e configurado no seu Eclipse é só seguir os próximos passos, caso contrário, utilize o **Manual de Instalação** para instalação e o **Manual de Configuração** para configurar o toolkit e em seguida siga os passos abaixo.

2. Criação de Ferramentas

As ferramentas são criadas a partir do ponto de extensão de ferramentas **aimv.tools** disponibilizado pelo toolkit. Para adicionar este ponto de extensão, utilize a aba **Extensions** do arquivo **plugin.xml** do seu plug-in e selecione a opção **Add..,** em seguida, adicione o ponto de extensão **aimv.tools** e clique em **Finish**, como mostra a Figura 64.

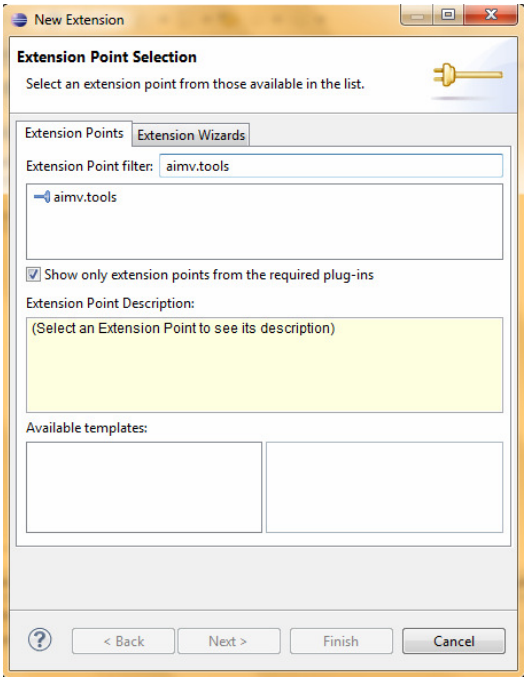

Figura 64: Adicionando Ponto de Extensão para Criação de Ferramentas

Depois de adicionar o ponto de extensão já é possível a criação de ferramentas. Para criar uma nova ferramenta, clique com o botão esquerdo no ponto de extensão adicionado e selecione a opção **New > tool** (Figura 65).

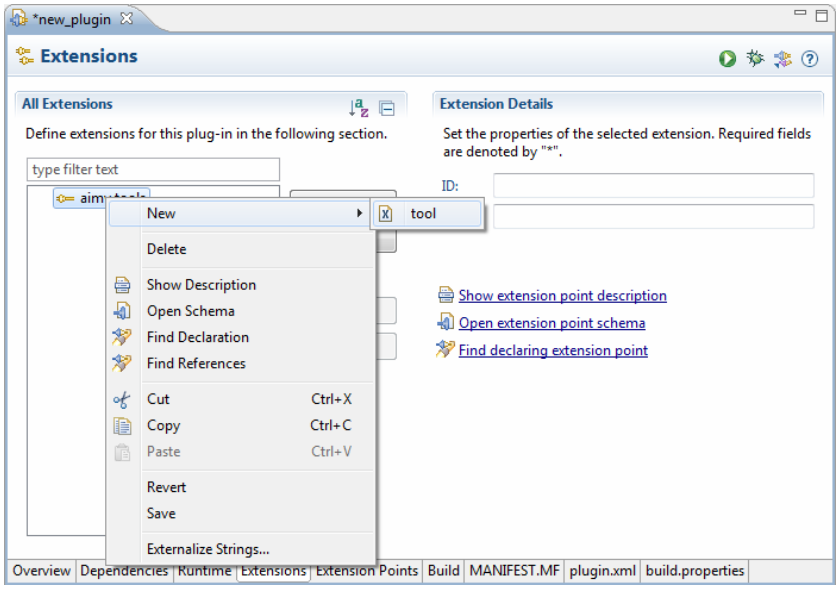

Figura 65: Criando Nova Ferramenta

Após a criação da ferramenta é necessário inserir algumas informações para sua identificação. Existem três atributos que são utilizados para caracterizar as ferramentas, como mostra a Figura 66:

id usado para identificar e diferenciar a ferramenta das outras ferramentas

name usado para atribuir um nome para a ferramenta

class usado para selecionar a classe que irá representar a ferramenta

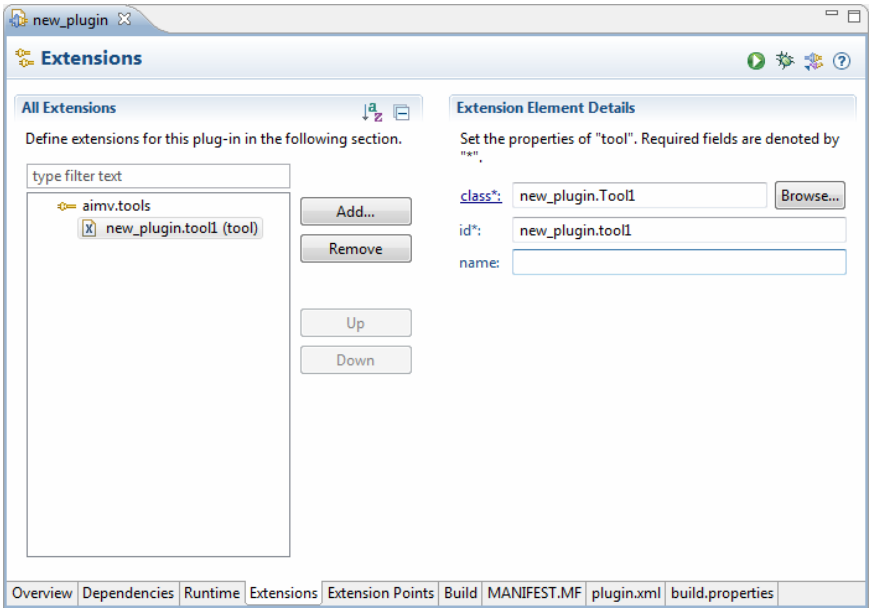

Figura 66: Inserindo Informações da Nova Ferramenta

O ponto de extensão de ferramentas utiliza como base para criação de novas ferramentas a classe **aimv.tools.Tool**. Esta classe fornece os métodos necessários para implementação das funcionalidades de cada ferramenta. Desta forma, ao criar uma nova ferramenta é necessário criar uma classe que faça herança com a classe **aimv.tools.Tool**. Esta classe será utilizada para representar a ferramenta. Se a classe já estiver criada então utilize a opção **Browse...** e selecione a classe, caso contrário clique no atributo **class**. Irá aparecer uma tela para criação da nova classe e o usuário pode preencher as informações necessárias da classe (Figura 67).

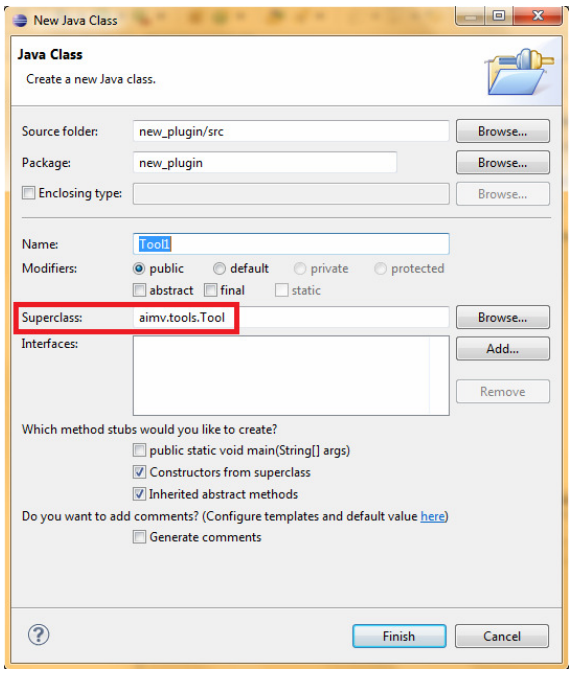

Figura 67: Criando Classe para Representar Ferramenta

A classe será criada com os métodos a serem implementados, como mostra a Figura 68. Os métodos fornecidos pela classe **aimv.tools.Tool** são apresentados a seguir.

activeTool(...): método utilizado para ativar a ferramenta. Este método recebe uma lista de argumentos que são utilizados para realizar as ações da ferramenta.

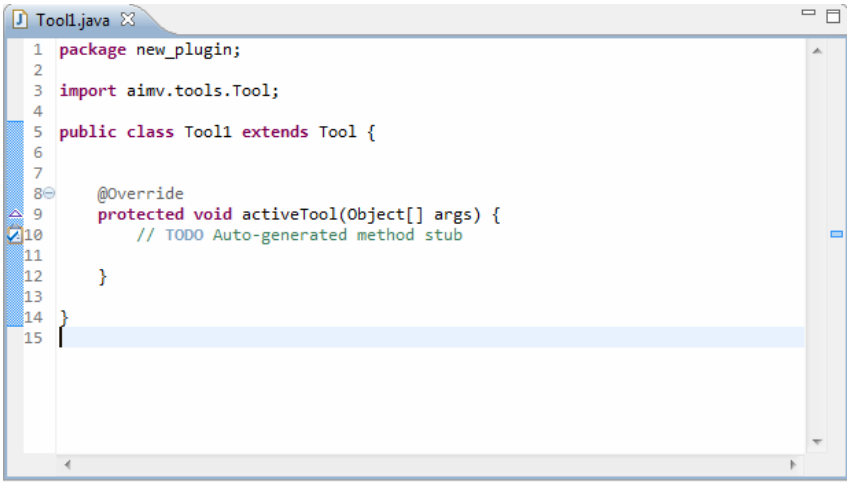

Figura 68: Exemplo de Classe para Representação de uma Ferramenta

3. Criação de Atalhos para Ferramentas

As ferramentas são aplicadas através de **Actions** (ações). As **Actions**[2] são recursos disponibilizados pelo Eclipse que podem ser anexados a componentes visuais, tais como, botões e menus. Estes recursos possuem métodos que são chamados sempre que os usuários finais, por exemplo, clicam sobre um item de menu ou botão. Desta forma, estes recursos podem ser utilizados para ativar as ferramentas. As **Actions** podem ser incluídas em diversos contextos da aplicação, desde a barra de menus e barra de ferramentas da aplicação e até mesmo na barra de ferramentas de cada visão ou janela da aplicação.

[2] Exemplos de uso de Actions podem ser encontrados em http://www.eclipsepluginsite.com/actions.html

Roteiro 4 – Criação de Paradigmas

1. Requisitos Básicos

Para utilizar os recursos para criação de paradigmas do toolkit AIMV é necessário antes a sua configuração. Se o toolkit já está totalmente instalado e configurado no seu Eclipse é só seguir os próximos passos, caso contrário, utilize o **Manual de Instalação** para instalação e o **Manual de Configuração** para configurar o toolkit. Em seguida baixe o plug-in **draw2d**. Este plug-in é um kit de ferramentas para layout que trabalha em cima do SWT. Após baixar, copie o plug-in para pasta de plug-ins do eclipse.

2. Criação de Paradigmas

Os paradigmas são criados a partir do ponto de extensão **org.eclipse.ui.views**. Para adicionar este ponto de extensão, utilize a aba **Extensions** do arquivo **plugin.xml** do seu plug-in e selecione a opção **Add..,** em seguida, adicione o ponto de extensão **org.eclipse.ui.views** e clique em **Finish**, como mostra a Figura 69.

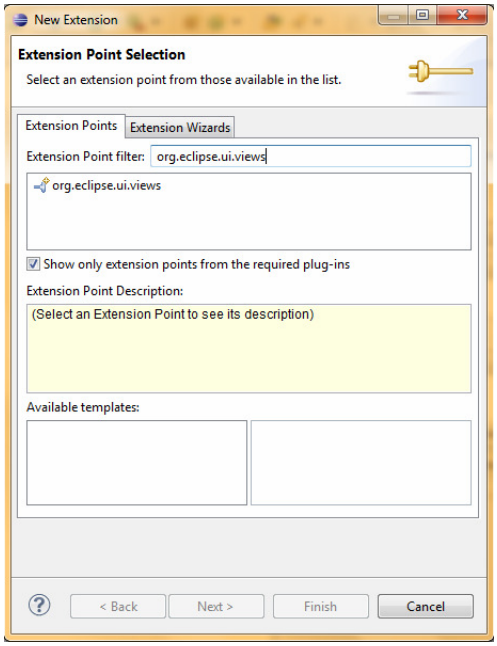

Figura 69: Adicionando Ponto de Extensão para Criação de Paradigmas

Depois de adicionar o ponto de extensão já é possível a criação de paradigmas. Para criar um novo paradigma, clique com o botão esquerdo no ponto de extensão adicionado e selecione a opção **New > view** (Figura 70).

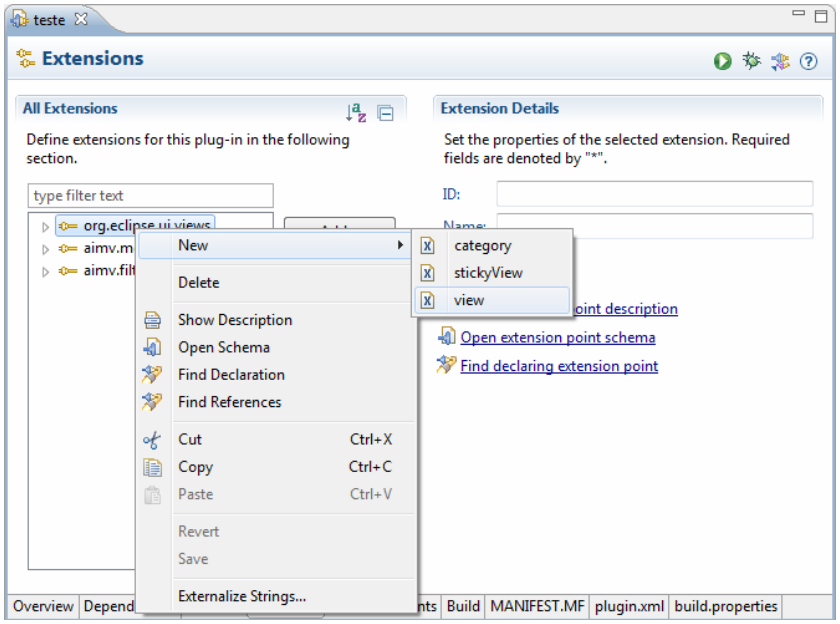

Figura 70: Criando Paradigma

Após a criação da visão é necessário inserir algumas informações para sua identificação. Os atributos que são utilizados para caracterizar os paradigmas são ilustrados na Figura 71.

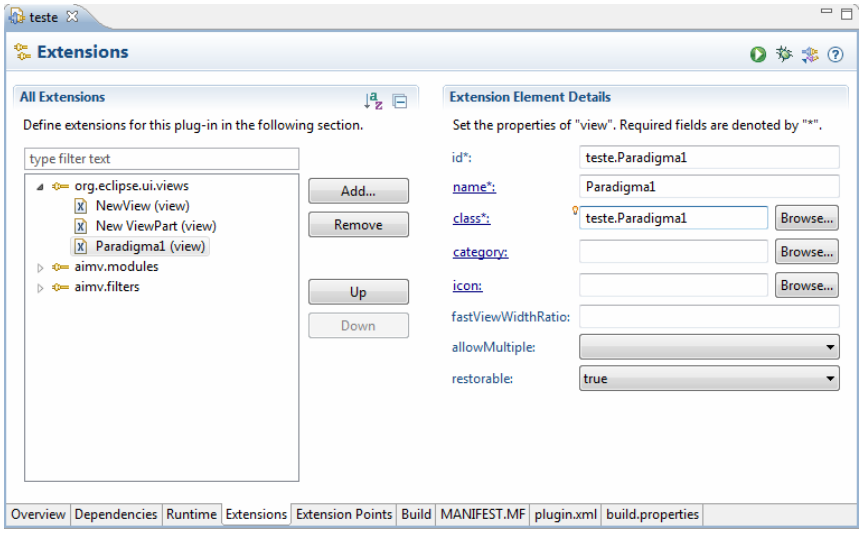

Figura 71: Inserindo Informações do Paradigma

A criação de paradigmas tem como base a classe **aimv.views.Paradigm**. Esta classe fornece os métodos necessários para implementação das funcionalidades de cada paradigma. Desta forma, ao criar um novo paradgima é necessário criar uma classe que faça herança com a classe **aimv.views.Paradigm**. Esta classe será utilizada para representar o paradigma. Se a classe já estiver criada então utilize a opção **Browse...** e selecione a classe, caso contrário clique no atributo **class**. Irá aparecer uma tela para criação da nova classe e o usuário pode preencher as informações necessárias da classe, como mostra a Figura 72.

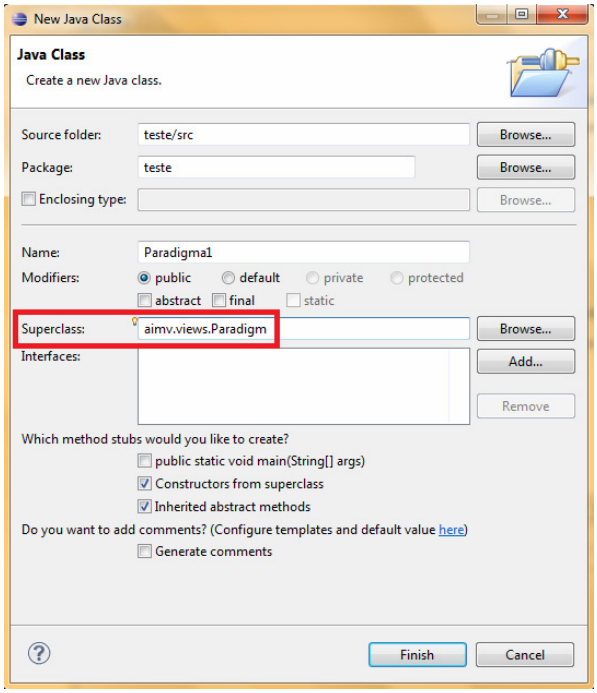

Figura 72: Criando Classe para Representar o Paradigma

A classe será criada com os métodos a serem implementados, como mostra a Figura 73. Os métodos fornecidos pela classe **aimv.views.Paradigm** são apresentados a seguir.

createLayout(): método utilizado para criação dos componentes visuais da visão de filtragem.

open(): método utilizado para definir as ações a serem tomadas quando a visão de filtragem for aberta.

closed(): método utilizado para definir as ações a serem tomadas quando a visão de filtragem for fechada.

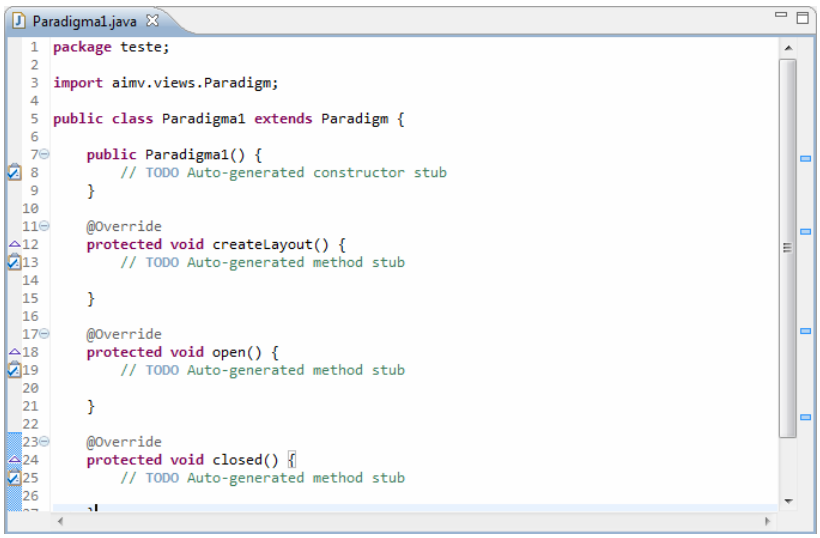

Figura 73: Exemplo de classe para Representação de um Paradigma

3. Criando os componentes Visuais

A classe **aimv.paradigms.Paradigm** disponibiliza o método layout. Esse método é responsável pela criação dos componentes visuais do novo paradigma. Os componentes visuais podem ser criados diretamente através dos componentes SWT do eclipse ou dos componentes disponíveis no plug-in **draw2D**. O método **getFigure** da classe **Paradigm** fornece o container do tipo **Figure** onde serão adicionados os componentes criados. A seguir será apresentado um exemplo de criação de componentes gráficos. Para isso é necessário a utilização da classe **org.eclipse.draw2d.Figure** disponível no plug-in **draw2D**. O primeiro passo é criar uma classe que herde da classe **Figure**, como mostra a Figura 74. Em seguida sobrescrever o método **paintFigure**. Esse método é responsável por desenhar o componente.

Figura 74: Criando um Componente Gráfico

Depois de criar o componente é necessário adicioná-lo ao container do paradigma, como mostra a Figura 75.

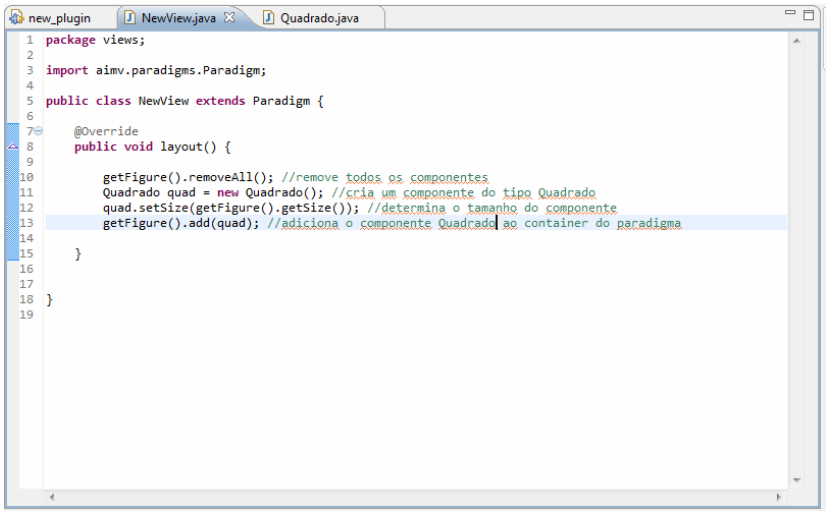

Figura 75: Adicionando Componente Gráfico no Paradigma

Neste exemplo foi criado um componente do tipo **Quadrado** para representar uma figura. Primeiramente foram removidos todos os componentes que estavam no container **getFigure** do paradigma. Depois foi criado o componente **quad** e foi determinado o seu tamanho. Em seguida o novo componente foi adicionado no container. A Figura 76 mostra a interface resultante do paradigma criado.

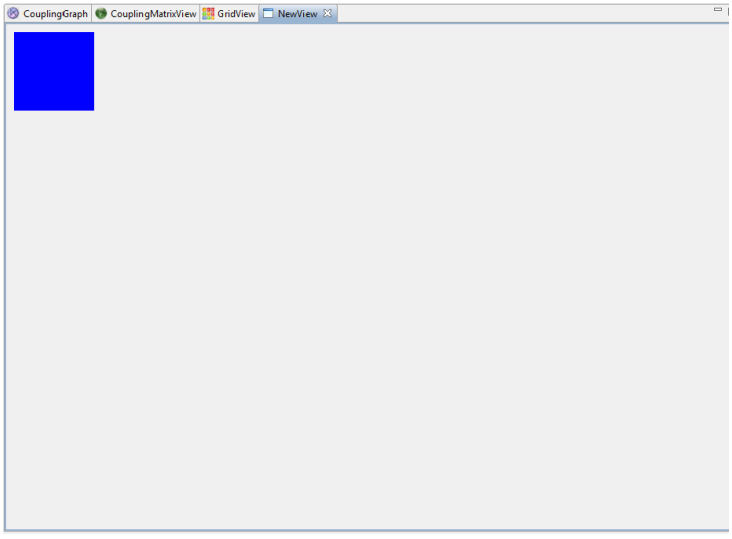

Figura 76: Interface do Paradigma Criado

Roteiro 5 – Tratamento e Utilização de Eventos

1. Requisitos Básicos

Para utilizar os recursos de eventos disponibilizados pelo toolkit AIMV é necessário antes a sua configuração. Se o toolkit já está totalmente instalado e configurado no seu Eclipse é só seguir os próximos passos, caso contrário, utilize o **Manual de Instalação** para instalação e o **Configuração do Ambiente** para configurar o toolkit e em seguida siga os passos abaixo.

2. Tipos de Eventos

Cada componente do toolkit é responsável por uma função específica dentro da sua arquitetura. Além disto, cada componente possui um conjunto de métodos específicos criados de acordo com o seu objetivo. Assim, cada componente possui um tipo de evento. Foram criadas então interfaces para criação de ouvintes para cada componente do toolkit. Todos as interfaces de eventos do toolkit estão no pacote **aimv.events**. As interfaces fornecidas são **INodeListener**, **IRelationListener**, **IGroupListener**, **IFilterListener**, **IToolListener**, **IViewAIMVListener** e **IControllerListener**. Cada interface destas é utilizada para um tipo de componente.

3. Criar Ouvintes de Eventos

Para criar um ouvinte de eventos para um determinado componente é necessário implementar a interface de evento mais adequada para aquele tipo de componente. Esta seção irá mostrar um exemplo de criação de ouvinte para o componente **Node**. A interface de evento para o componente **Node** é a **INodeListener**. Esta interface fornece os métodos que devem se implementados para criação dos seus ouvintes. A Figura 77 mostra um exemplo de criação de ouvinte utilizando a interface **INodeListener**.

Figura 77: Criando Ouvinte de Eventos

4. Adicionar e Remover Ouvintes

Depois de criar os ouvintes é preciso adicioná-los aos componentes para que os eventos sejam transmitidos. Para adicionar um ouvinte a um componente é preciso utilizar o método para adicionar listener presente em cada um dos componentes. Para o componente **Node** o método utilizado é **addNodeListener**. Este método recebe como parâmetro o ouvinte. Depois de adicionar o ouvinte, ao utilizar os métodos do componente será enviada informações aos ouvintes da sua lista de ouvintes. Da mesma forma, é possível remover um ouvinte utilizando o método **removeNodeListener**. A Figura 78 mostra um exemplo de adição e remoção de ouvintes dos componentes **Node**.

Figura 78: Adicionando e Removendo Ouvintes

Obs: É recomendado que os ouvintes que não estão sendo mais utilizados sejam removidos da lista de ouvintes.

5. Obter Informações dos Eventos

Os eventos propagados são recebidos pelos seus ouvintes através das suas classes de eventos. A classe de evento do componente **Node** é a **NodeEvent**. Esta classe contém todas as informações do evento gerado pelo componente **Node**. Para acessar as informações contidas nos eventos é a utilização dos seus métodos. A Figura 79 mostra um exemplo de utilização da classe de evento **NodeEvent**.

NewView.java

```
1  package view;
2
3  import org.eclipse.swt.widgets.Composite;
10
11 public class NewView extends UtilityView implements INodeListener {
12
13     public NewView() {
14         // TODO Auto-generated constructor stub
15     }
16
17     @Override
18     protected void createLayout(Composite parent) {
19
20         for (Node node : Nodes.getGroup("all").getNodes()) {
21             node.removeNodeListener(this);
22             node.addNodeListener(this);
23         }
24
25     }
26
27     @Override
28     public void setPropertyNode(NodeEvent event) {
29
30         Node node = event.getNode();
31         System.out.println(node.getID());
32
33     }
34
35 }
36
```

Figura 79: Obtendo Informações dos Eventos