

(c) Copyright Eclipse contributors and others, 2000, 2008. All rights reserved. Java and all Java-related trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S., other countries, or both. Eclipse is a trademark of the Eclipse Foundation, Inc.



Développement sous IDE

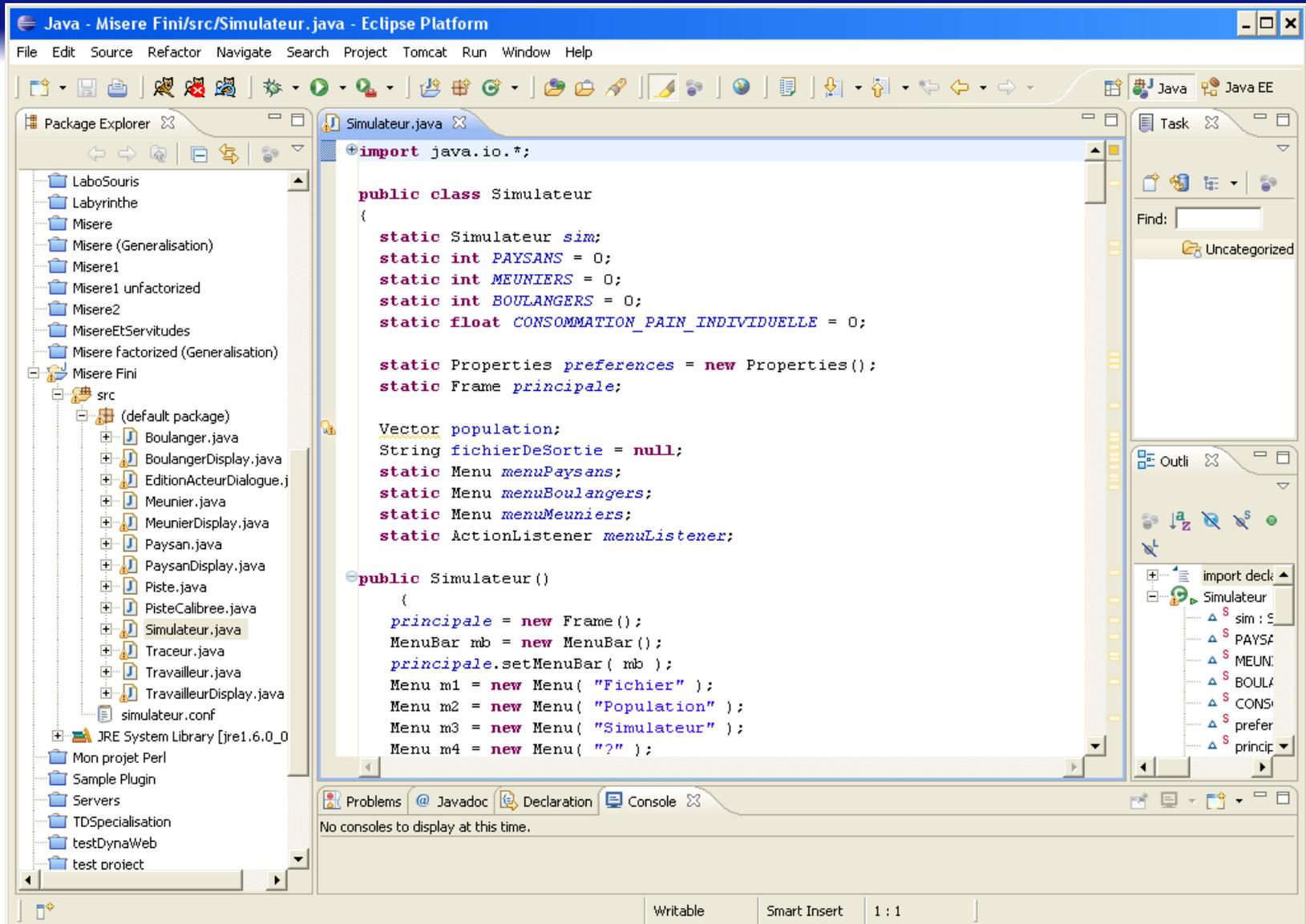
- Les perspectives :
 - Chaque type de projet peut proposer des nouvelles perspectives.
 - Elles organisent des phases du travail : pour Java
 - La construction des squelettes } Perspective "Java"
 - L'écriture de code
 - Le test et le débogage } Perspective "Debug"
 - L'exploration de code } Perspective "Java Browsing"



Perspective Java

- Elle favorise l'écriture et l'édition de code :
 - Browser dans les différents sources du projet
 - Ecrire du code dans un éditeur
 - Visualiser la structure de la classe sur laquelle on travaille.
 - Observer les résultats immédiats du compilateur

Perspective Java



The screenshot shows the Eclipse IDE in the Java perspective. The main editor displays the source code for `Simulateur.java`. The code includes imports, class declarations, static variables, and a constructor.

```

import java.io.*;

public class Simulateur
{
    static Simulateur sim;
    static int PAYSANS = 0;
    static int MEUNIER = 0;
    static int BOULANGERS = 0;
    static float CONSOMMATION_PAIN_INDIVIDUELLE = 0;

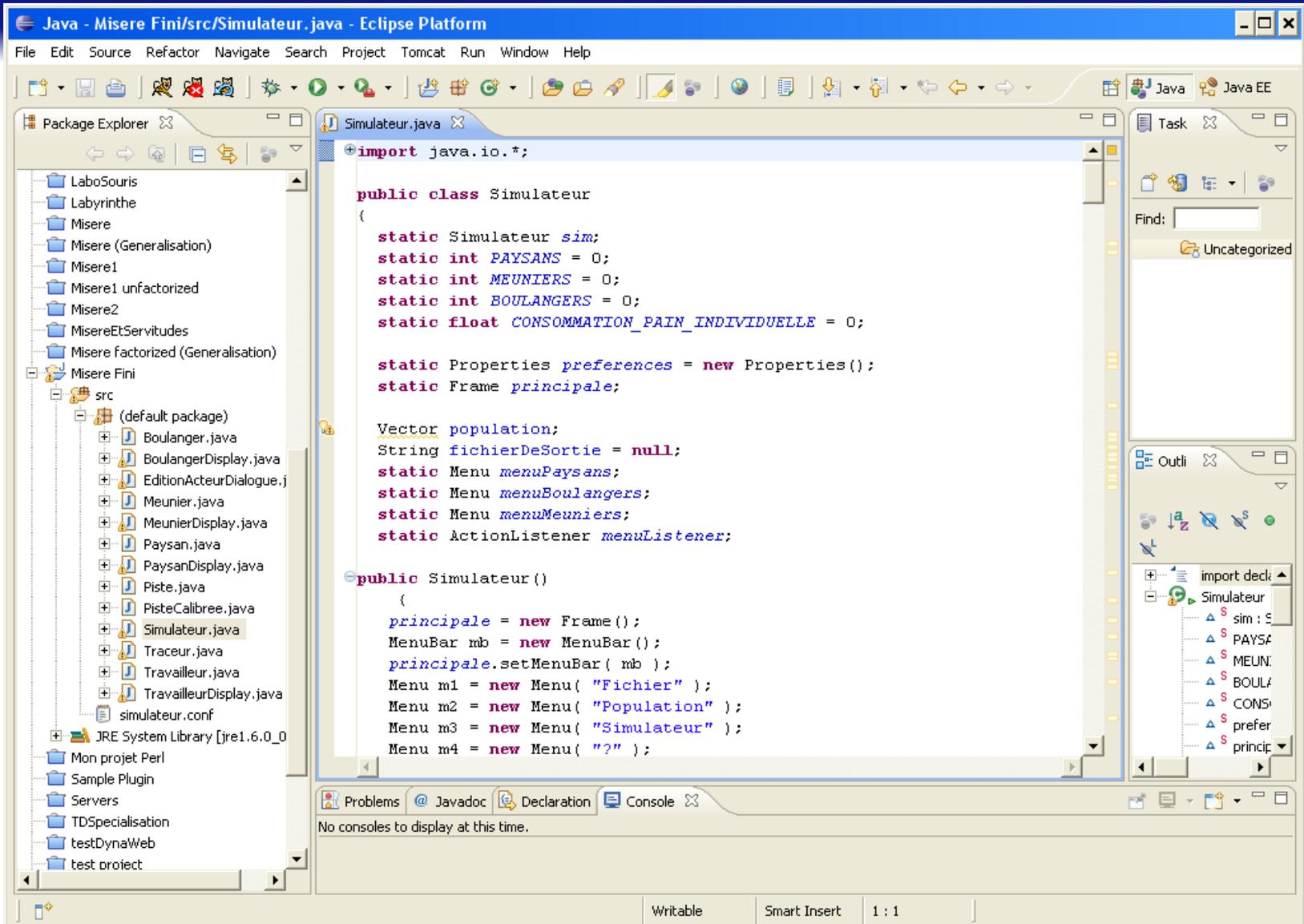
    static Properties preferences = new Properties();
    static Frame principale;

    Vector population;
    String fichierDeSortie = null;
    static Menu menuPaysans;
    static Menu menuBoulangers;
    static Menu menuMeuniers;
    static ActionListener menuListener;

    public Simulateur()
    {
        principale = new Frame();
        MenuBar mb = new MenuBar();
        principale.setMenuBar( mb );
        Menu m1 = new Menu( "Fichier" );
        Menu m2 = new Menu( "Population" );
        Menu m3 = new Menu( "Simulateur" );
        Menu m4 = new Menu( "?" );
    }
}
    
```

The Package Explorer on the left shows the project structure, including the `Misere Fini` project and its `src` directory. The right-hand side of the IDE contains the Task, Outli, and Import Declaration views.

Perspective Java



The screenshot shows the Eclipse IDE in the Java perspective. The main window displays the source code for `Simulateur.java`. The Package Explorer on the left shows the project structure, including the `Misere Fini` package and its `src` directory. The Task and Outlines views are visible on the right. The Console view at the bottom shows no output.

```

import java.io.*;

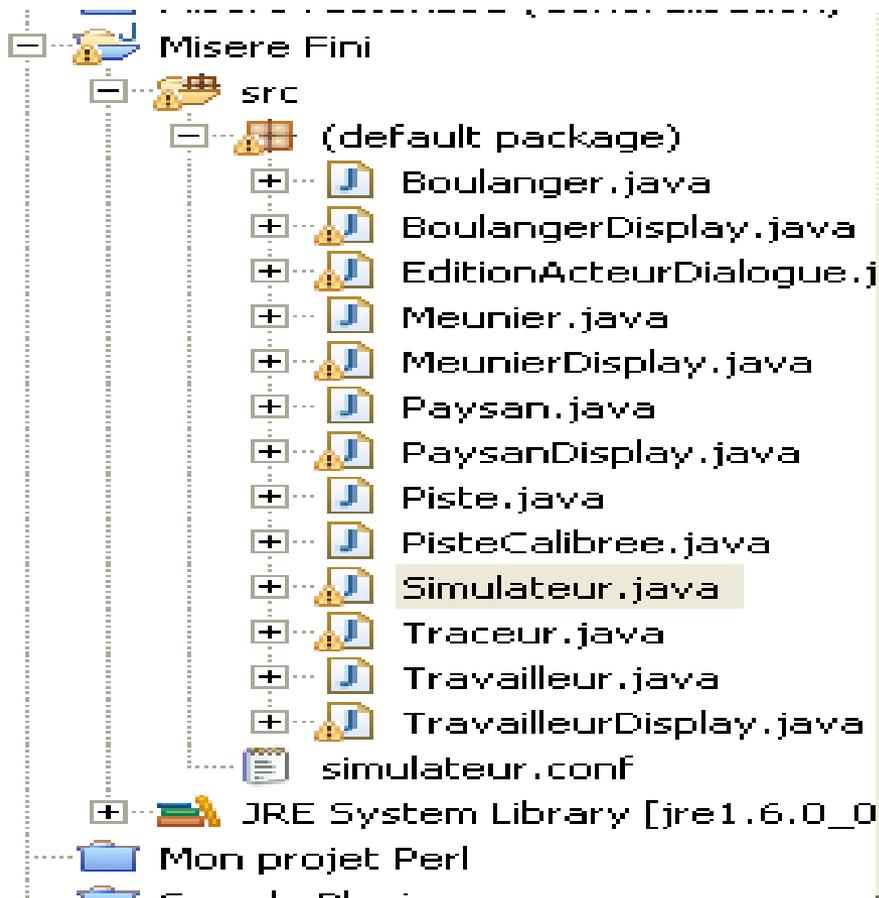
public class Simulateur
{
    static Simulateur sim;
    static int PAYSANS = 0;
    static int MEUNIER = 0;
    static int BOULANGERS = 0;
    static float CONSOMMATION_PAIN_INDIVIDUELLE = 0;

    static Properties preferences = new Properties();
    static Frame principale;

    Vector population;
    String fichierDeSortie = null;
    static Menu menuPaysans;
    static Menu menuBoulangers;
    static Menu menuMeuniers;
    static ActionListener menuListener;

    public Simulateur()
    {
        principale = new Frame();
        MenuBar mb = new MenuBar();
        principale.setMenuBar( mb );
        Menu m1 = new Menu( "Fichier" );
        Menu m2 = new Menu( "Population" );
        Menu m3 = new Menu( "Simulateur" );
        Menu m4 = new Menu( "?" );
    }
}
    
```

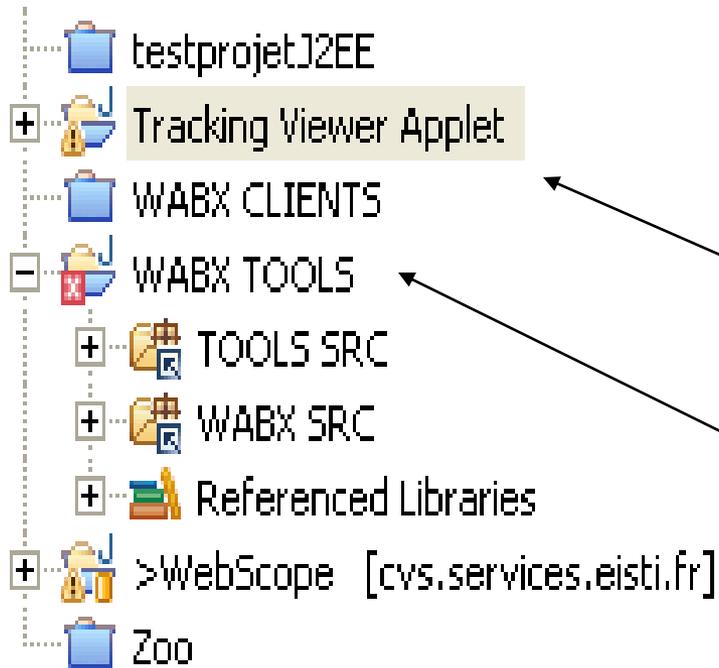
Le browser de projet



- Le browser de projet permet un accès rapide aux sources et le chargement dans l'éditeur.



Synchronise l'éditeur



- Le browser donne un aperçu rapide de la situation :

Ce projet compile mais à encore des Warnings

Ce projet ne peut être construit (build) il y a des erreurs

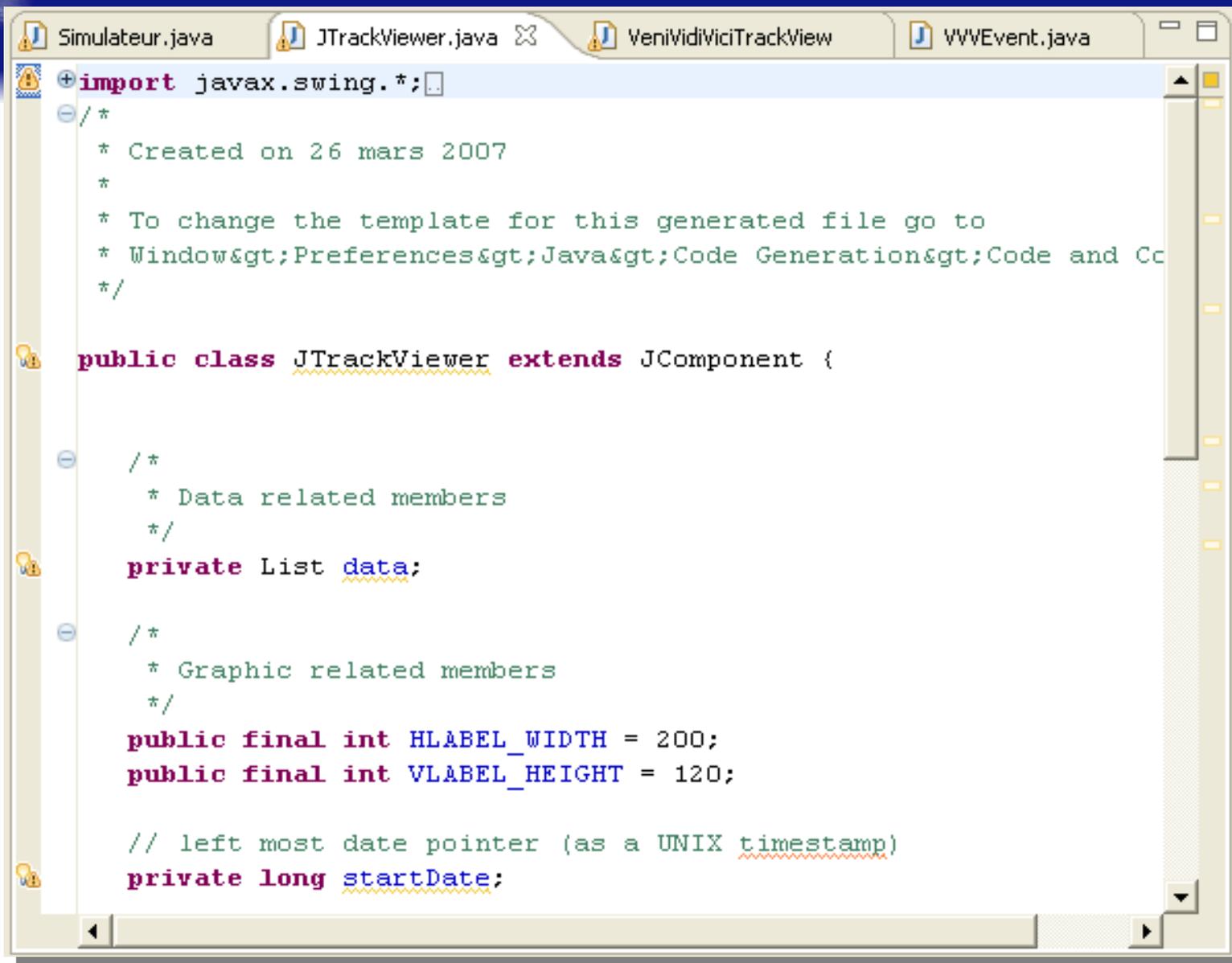
Ce projet est fermé

Ce projet est synchronisé



Les fonctions d'éditeur

- Les fonctions importantes à l'usage :
 - Autocomplétion : décharge la mémoire de travail du développeur
 - Coloration syntaxique : essentielle pour le repérage et les premières corrections
 - Signalement d'erreurs et compilation instantanée : possible en Java, pas nécessairement dans d'autres langages.
 - Suggestions dynamiques de correction.



The screenshot shows the Eclipse IDE interface. The top toolbar contains icons for File, Edit, and Run. The title bar shows four open files: Simulateur.java, JTrackViewer.java, VeniVidiViciTrackView, and WVEEvent.java. The main editor window displays the following Java code:

```
import javax.swing.*;

/*
 * Created on 26 mars 2007
 *
 * To change the template for this generated file go to
 * Window>Preferences>Java>Code Generation>Code and Co
 */

public class JTrackViewer extends JComponent {

    /*
     * Data related members
     */
    private List data;

    /*
     * Graphic related members
     */
    public final int HLABEL_WIDTH = 200;
    public final int VLABEL_HEIGHT = 120;

    // left most date pointer (as a UNIX timestamp)
    private long startDate;
```



La zone interactive sur la gauche indique des événements dans le code :

- Warnings
- Erreurs de syntaxe
- Erreurs de compilation et de liaison
- Surcharge



```
static Menu menuBoullanger  
static Menu menuMeuniers;  
... ..
```

Ces zones sont interactives et indiquent le problème et une solution si elle existe.



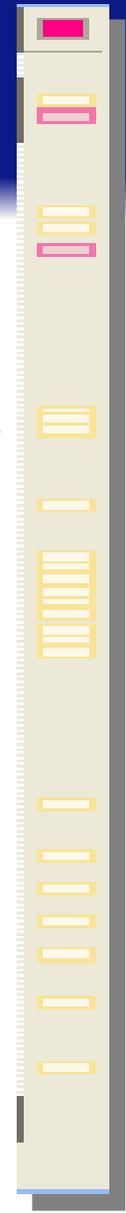
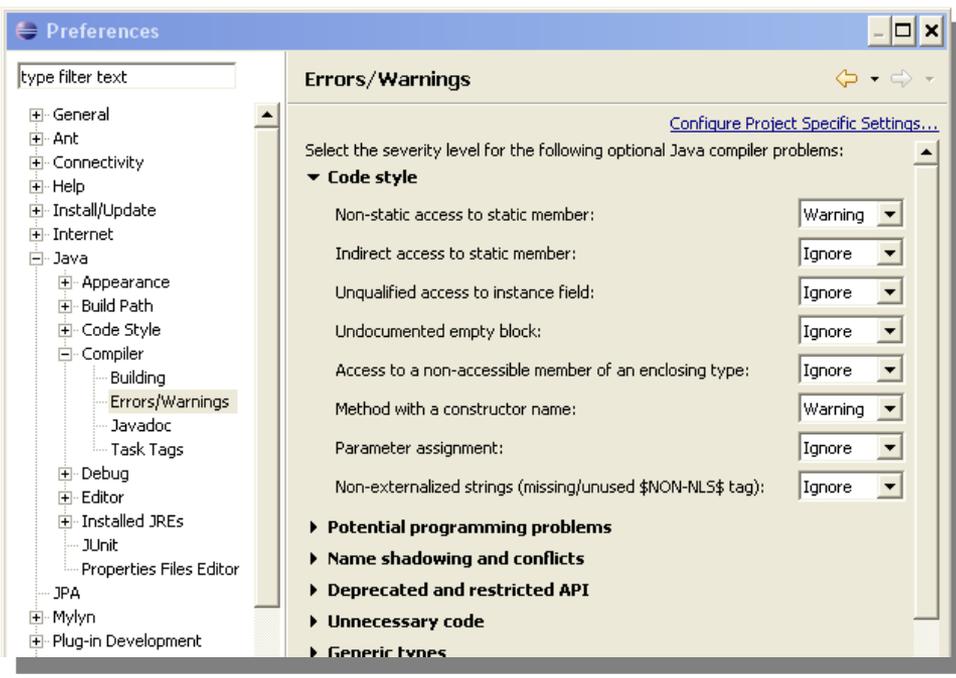
L'éditeur : zone de localisation

A droite, la zone de localisation permet de savoir où il reste des erreurs.

Erreurs

Warnings

Le nombre d'erreurs et warnings dépend du paramétrage du compilateur



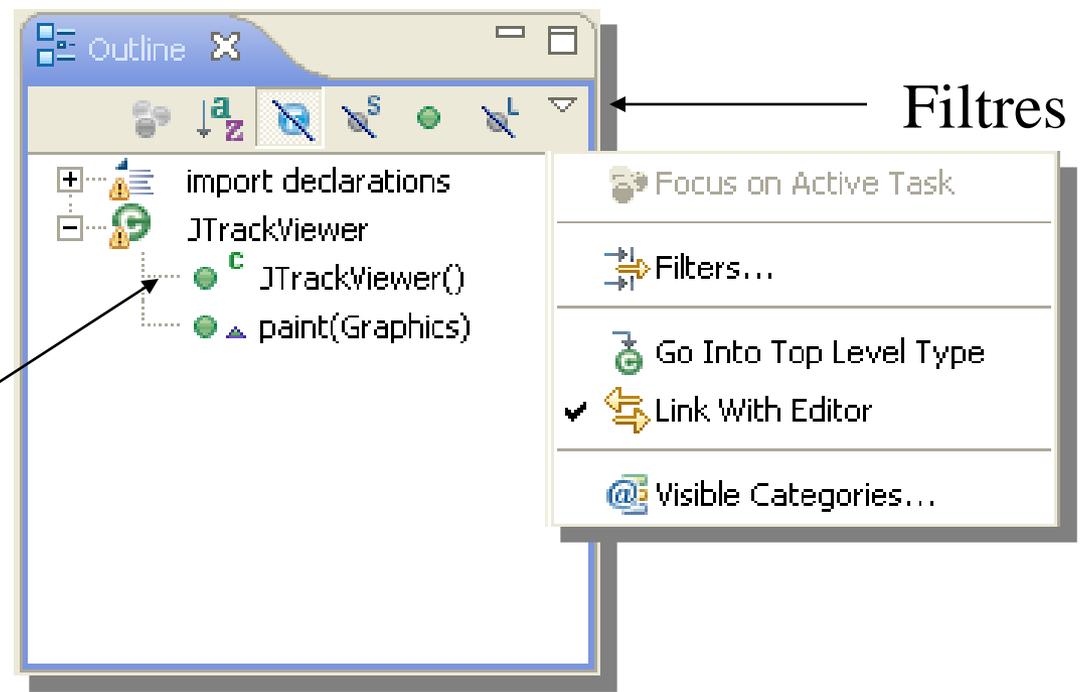
La hauteur symbolise la longueur du source



L'Outline

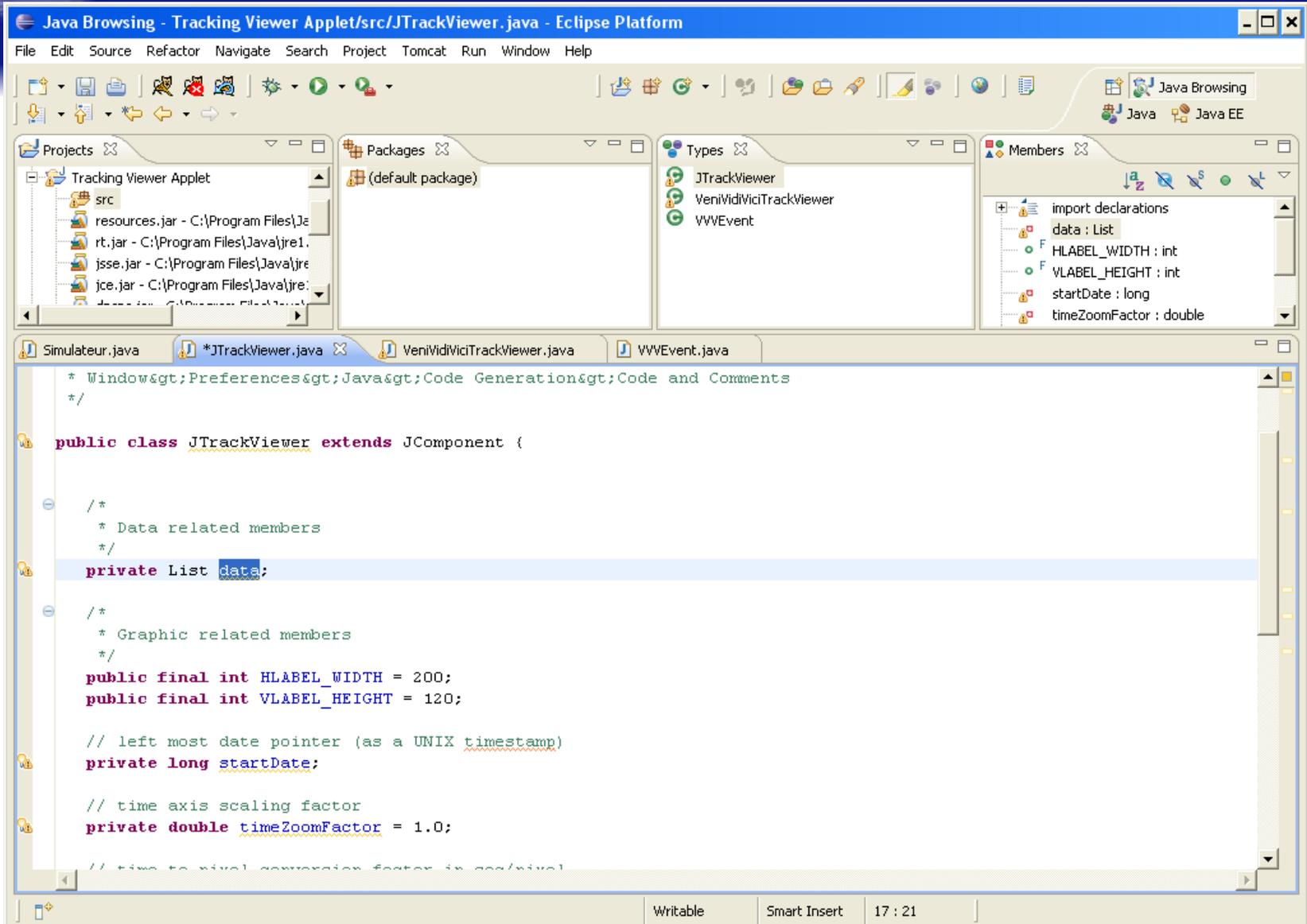
Si l'option de synchronisation est active, les liens pilotent l'affichage de l'éditeur

L'outline est situé à droite et décrit le fichier source



Liens

Perspective Java Browsing



The screenshot shows the Eclipse IDE in the 'Java Browsing' perspective. The main window title is 'Java Browsing - Tracking Viewer Applet/src/JTrackViewer.java - Eclipse Platform'. The menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Tomcat, Run, Window, and Help. The toolbar contains various icons for file operations and development tools.

The interface is divided into several panes:

- Projects:** Shows a tree view of the 'Tracking Viewer Applet' project, including subfolders like 'src' and 'resources.jar'.
- Packages:** Shows the 'default package'.
- Types:** Lists the classes 'JTrackViewer', 'VeniVidiViciTrackViewer', and 'WVEvent'.
- Members:** Displays the members of the selected class, including 'import declarations', 'data : List', 'HLABEL_WIDTH : int', 'VLABEL_HEIGHT : int', 'startDate : long', and 'timeZoomFactor : double'.

The editor shows the source code for 'JTrackViewer.java'. The code is as follows:

```

* Window>>Preferences>>Java>>Code Generation>>Code and Comments
*/

public class JTrackViewer extends JComponent {

    /*
     * Data related members
     */
    private List data;

    /*
     * Graphic related members
     */
    public final int HLABEL_WIDTH = 200;
    public final int VLABEL_HEIGHT = 120;

    // left most date pointer (as a UNIX timestamp)
    private long startDate;

    // time axis scaling factor
    private double timeZoomFactor = 1.0;

    // time to pixel conversion factor in sec/pixel

```

The status bar at the bottom indicates 'Writable', 'Smart Insert', and '17 : 21'.



L'exécution sous Eclipse

- On peut exécuter des programmes directement à partir d'Eclipse.
 - L'exécution est lancée à travers Eclipse
 - Eclipse garde le contrôle de ce qui est exécuté
 - Eclipse capture les entrées sorties standard du programme exécuté.
 - Eclipse mémorise de très nombreuses "configurations de lancement"

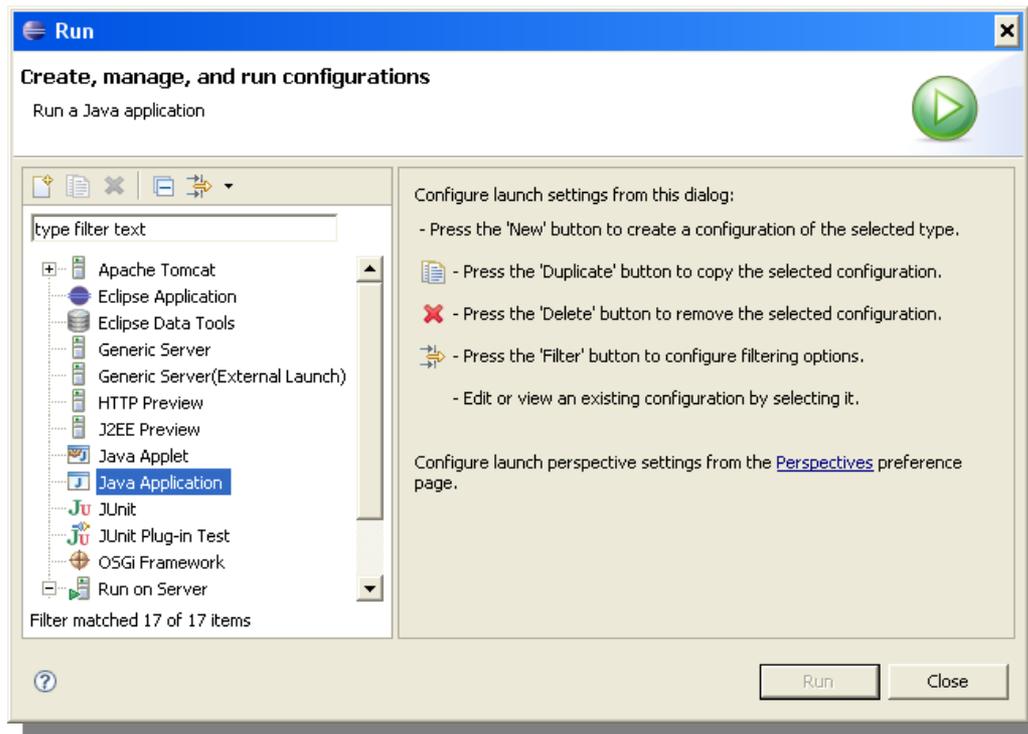
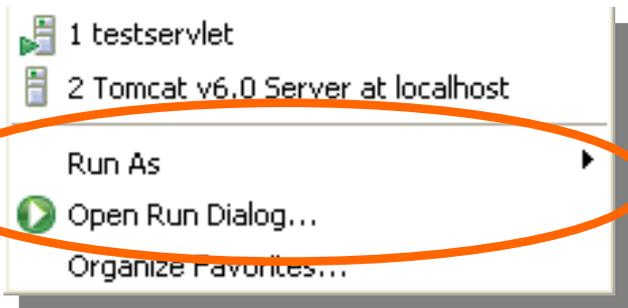
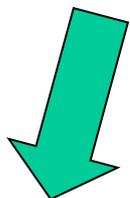
L'exécution sous Eclipse



Lancement des outils externes

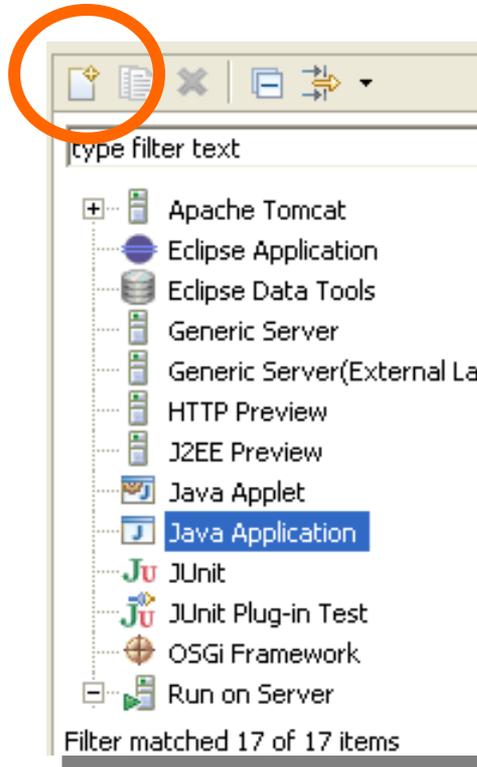
Lancement de l'exécution

Lancement du debug

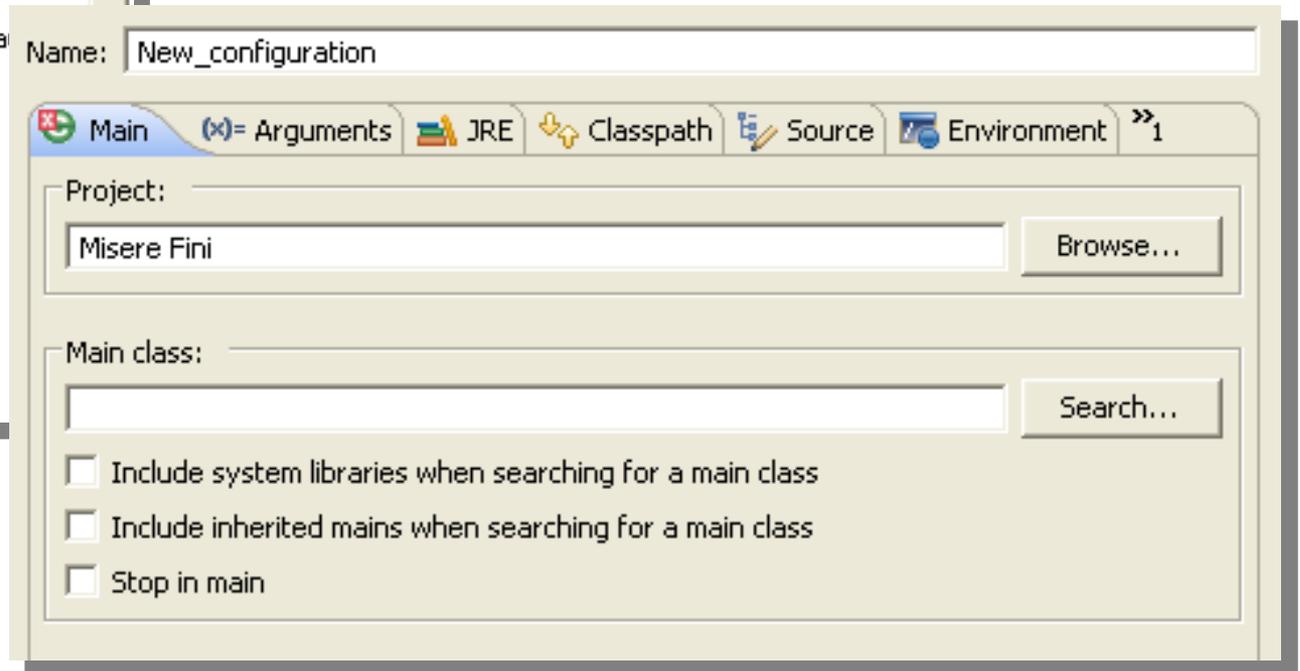




L'exécution sous Eclipse (2)

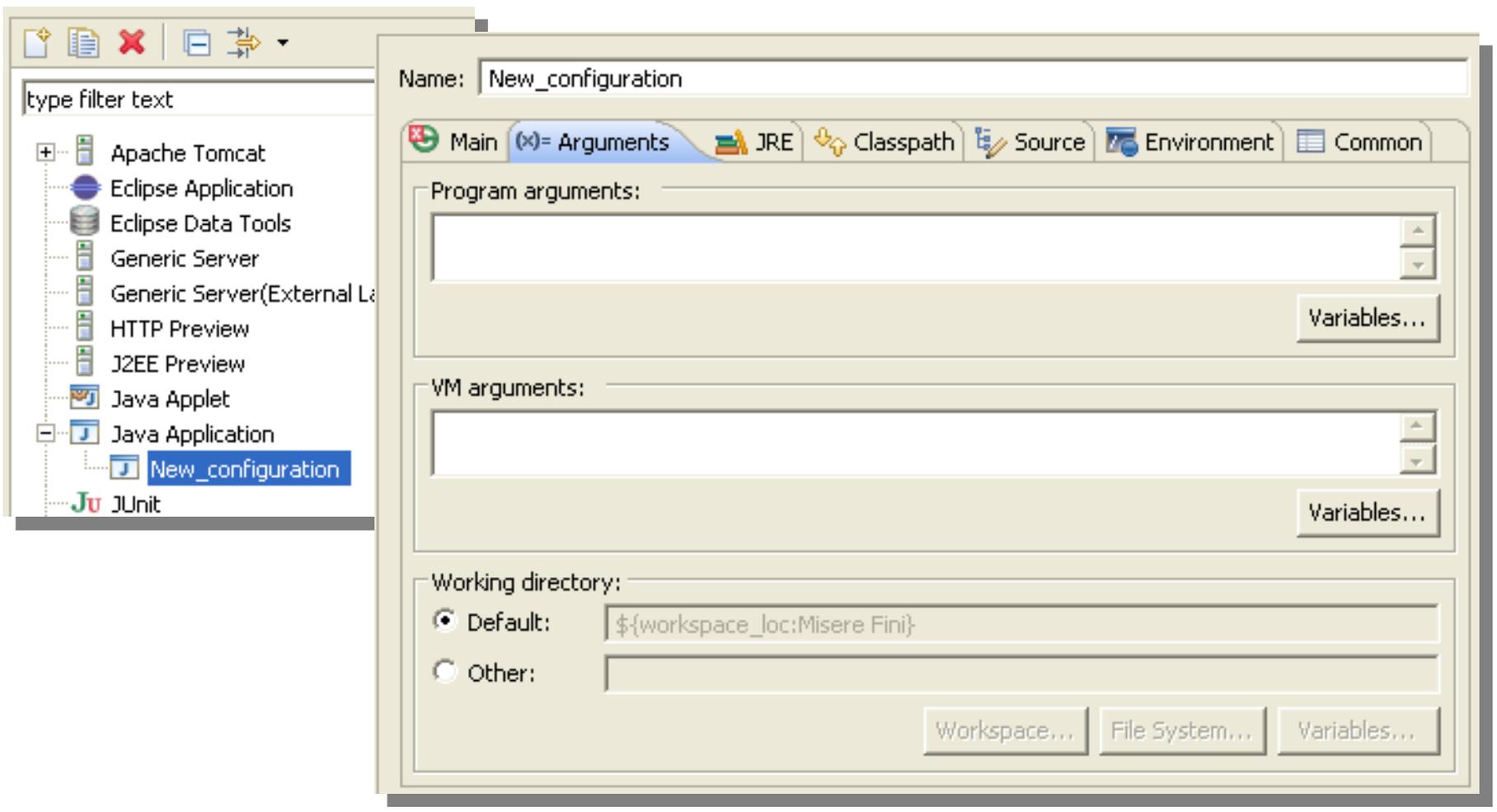


Eclipse stocke des "configurations d'exécution", rangés selon différents types.
Il faut créer une nouvelle exécution pour pouvoir la lancer.

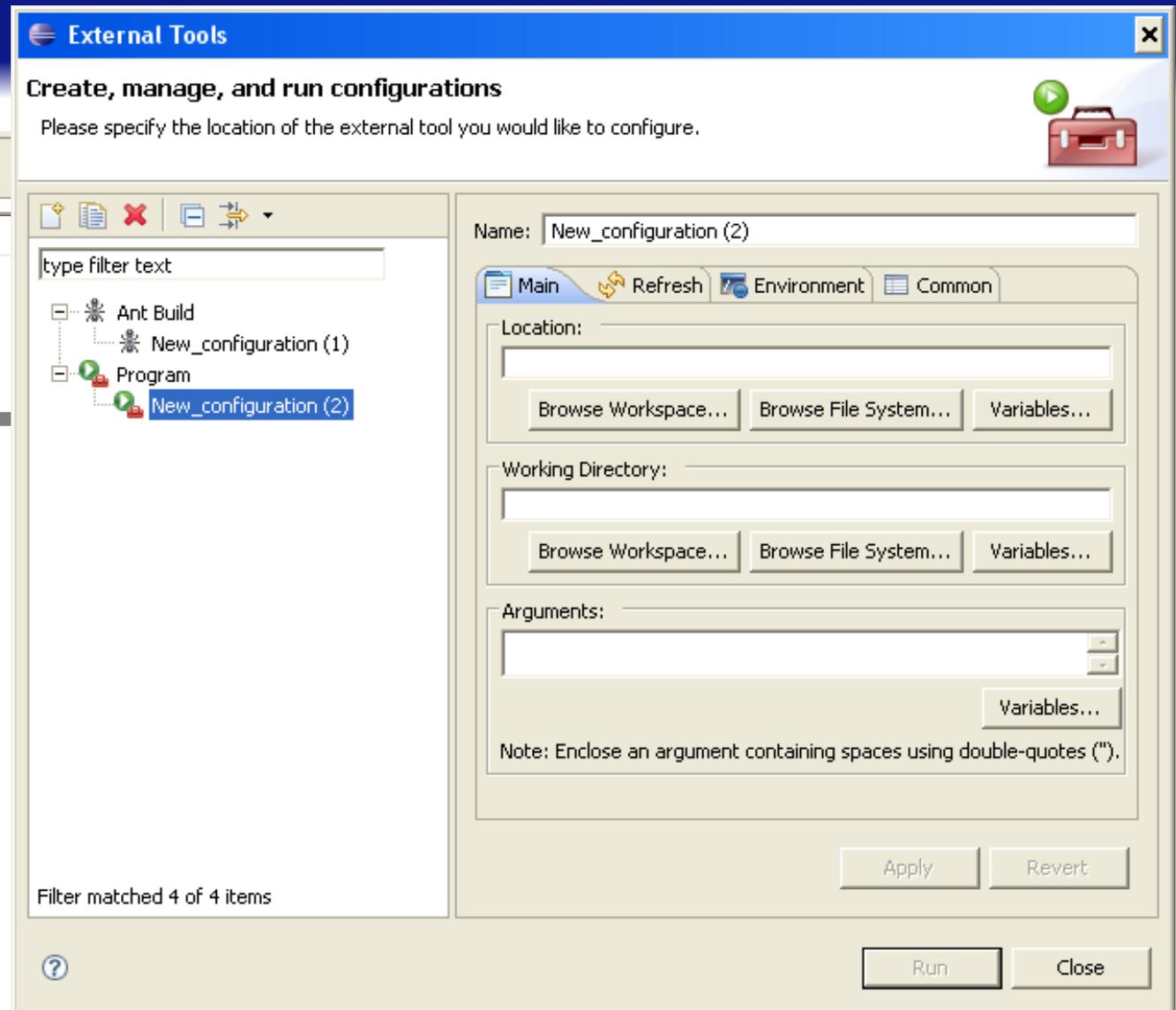




L'exécution sous Eclipse (3)



L'exécution de programmes externes



External Tools

Create, manage, and run configurations

Please specify the location of the external tool you would like to configure.

Name:

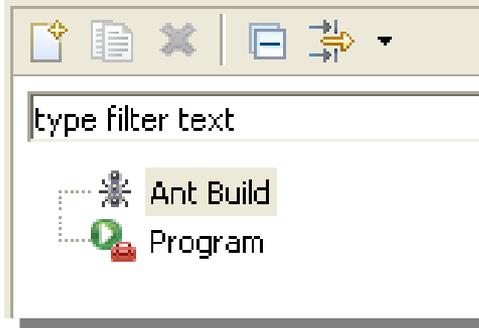
Location:

Working Directory:

Arguments:

Note: Enclose an argument containing spaces using double-quotes ("").

Filter matched 4 of 4 items

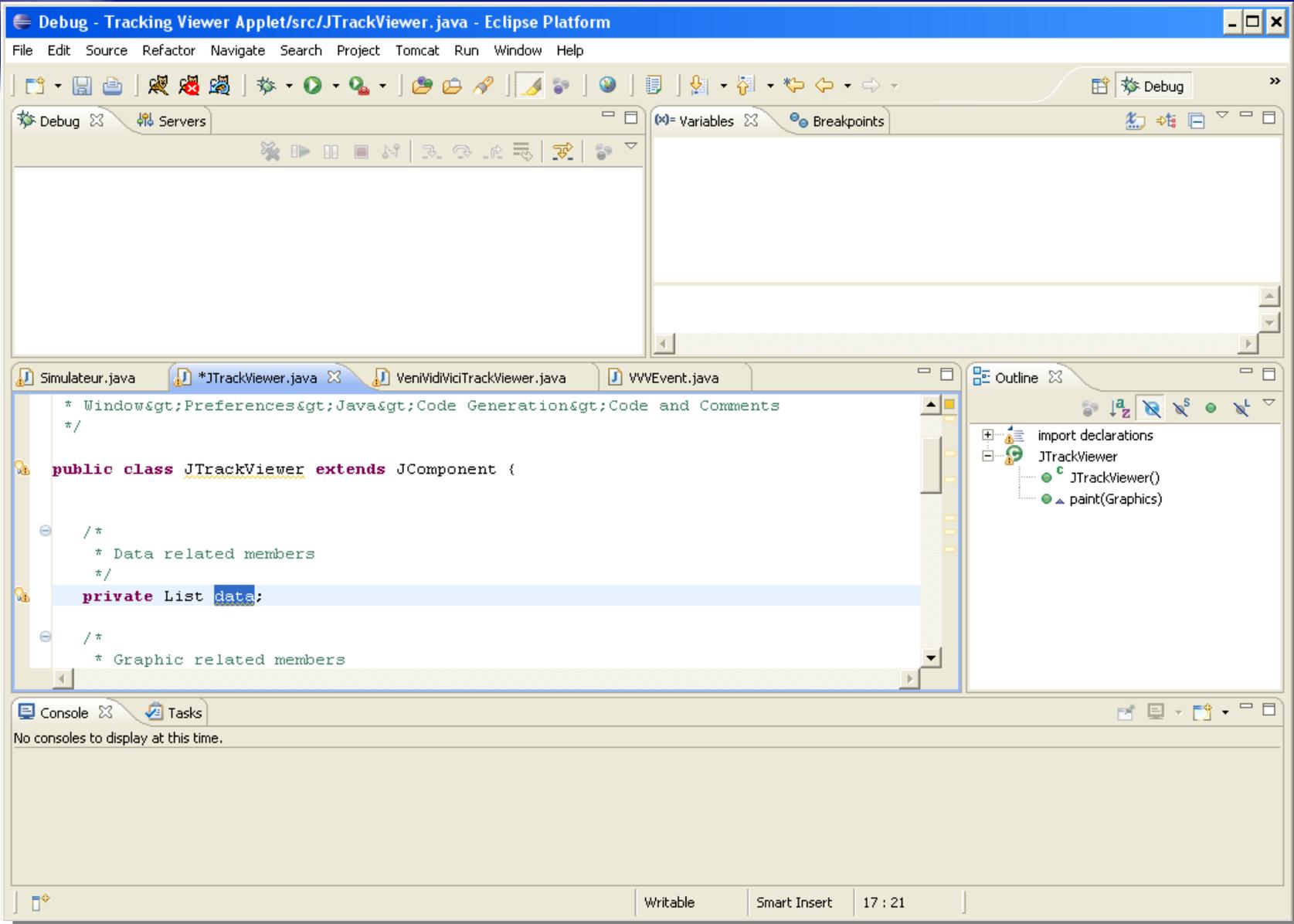


type filter text

Ant Build

Program

Perspective Debug



The screenshot shows the Eclipse IDE in the Debug perspective. The main editor displays the source code for `JTrackViewer.java`, which extends `JComponent`. The code includes comments for data and graphic related members, and a `private List data;` declaration is highlighted. The Outline view on the right shows the class structure with `JTrackViewer` and its methods `JTrackViewer()` and `paint(Graphics)`. The bottom console area is empty, displaying the message "No consoles to display at this time." The status bar at the bottom indicates "Writable", "Smart Insert", and the time "17 : 21".

```

/* Window>Preferences>Java>Code Generation>Code and Comments
*/
public class JTrackViewer extends JComponent {

    /*
     * Data related members
     */
    private List data;

    /*
     * Graphic related members
  
```



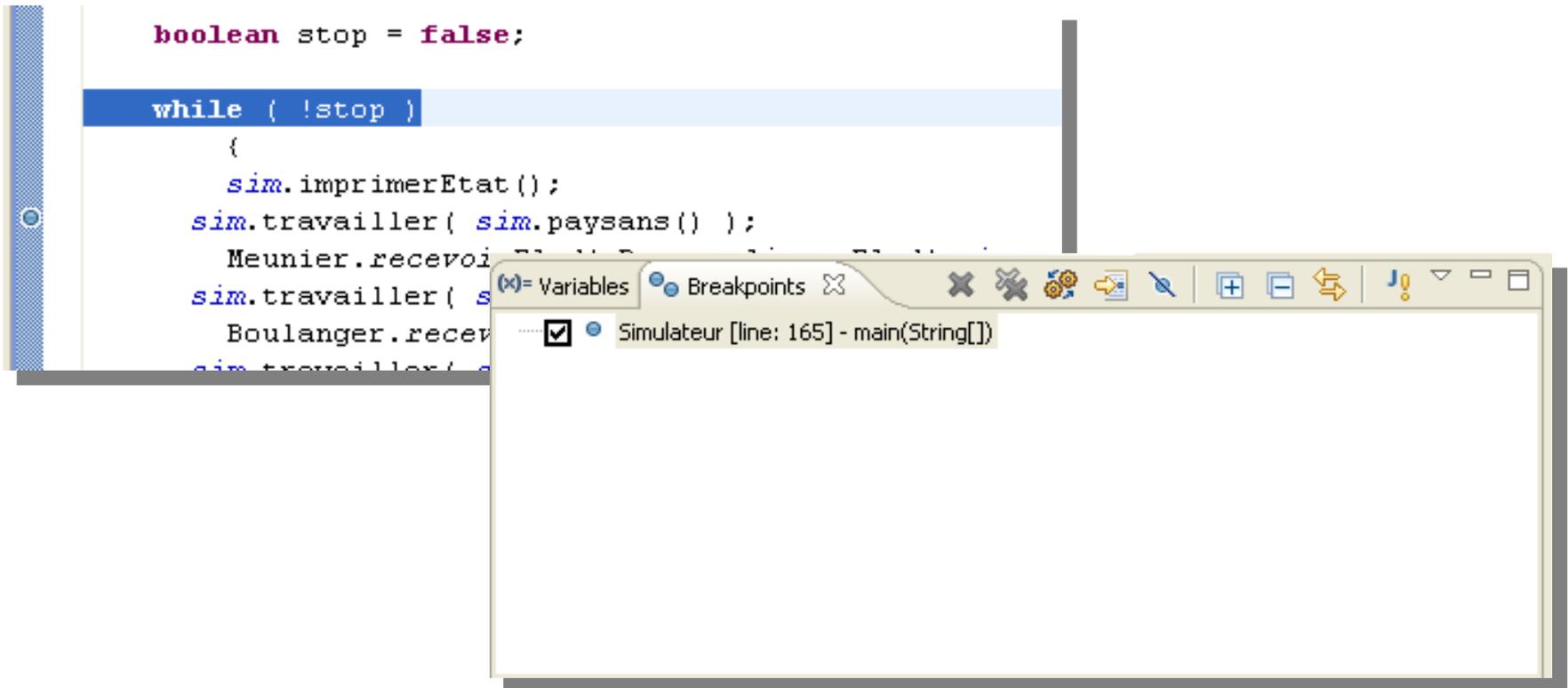
Le debug sous Eclipse

- De quoi a-t-on besoin ?
 - Du code et d'une possibilité d'y naviguer
 - Du contrôle du programme en exécution
 - Du contrôle de l'exécution du programme
 - D'observation du programme en cours "make it observable !!"
 - D'interagir avec les entrées sorties lorsque nécessaire ou demandé par le programme.



Les points d'arrêt et l'intrusion dans l'exécution

On met un point d'arrêt en double-cliquant dans le code





Les points d'arrêt et l'intrusion dans l'exécution

On met un point d'arrêt en double-cliquant dans le code

```
while ( !stop  
{  
    sim.imprimerEtat ();  
    sim.travailler( sim.paysans() );  
    Meunier.recevoirBled( Paysan.livrerBled( ... );  
    sim.travailler( sim.meuniers() );  
    Boulanger.recevoirFarine( Meunier.livrerF...
```

On lance le programme en mode debug.



Name	Value
argv	String[0] (id=25)
richier.references	File (id=27)
stop	false



L'observation du processus / programme

