



# Java Integrated Development Environments: ECLIPSE



# Part1

## Installation



# Eclipse Installation

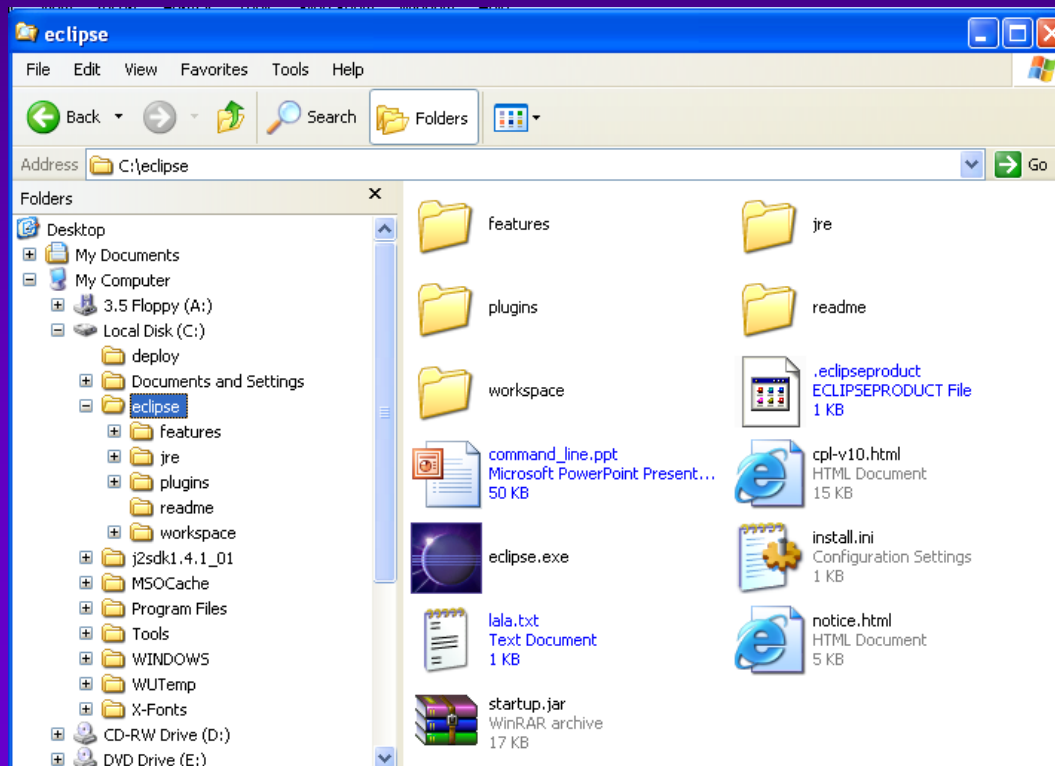
---

- Detailed Instructions  
[http://www.3plus4software.de/eclipse/installation\\_en.html](http://www.3plus4software.de/eclipse/installation_en.html)
- Basic Ingredients:
- Windows, Linux, Solaris, QNX or Mac OS/X operating system.
- 256 MB RAM or more
- A Java 2 runtime environment (JRE) or Java 2 Software Development Kit (J2SDK). Eclipse needs version 1.3 or higher .
- Eclipse SDK version 2.1.3 (the last stable version)



# Eclipse Installation

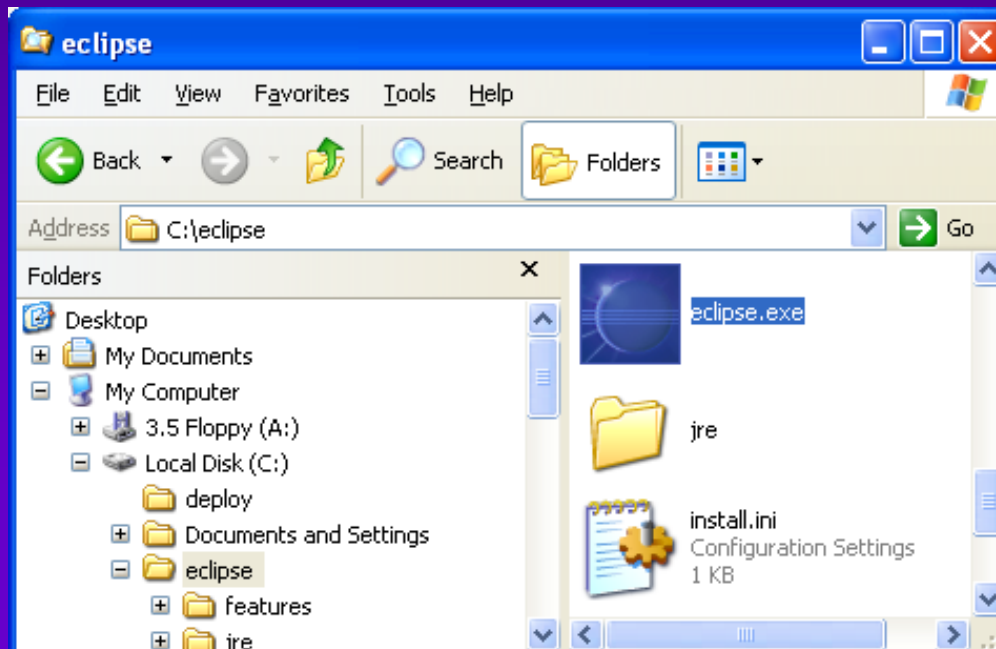
- Extract eclipse-SDK-*\*-\**.zip in the path that will now more be the %eclipse\_home%. An eclipse directory will be created.
  - *E.g. C:\eclipse*





# Eclipse Installation

- Eclipse is a Java program. You have to specify the Java path. There are two ways:
  - 1<sup>st</sup>: Specify the Java path in the PATH environment variable
  - 2<sup>nd</sup>: Copy the jre folder from the JAVA\_HOME directory to the ECLIPSE\_HOME directory
- Run eclipse; Click on the eclipse icon





# Eclipse Installation

---

- Alternative. Run eclipse from a command line window specifying the Java path (-vm)
- Gives flexibility in specifying various parameters when starting eclipse

```
C:\WINDOWS\System32\cmd.exe
C:\eclipse>eclipse.exe -vm C:\j2sdk1.4.1_01\jre\bin\javaw.exe
C:\eclipse>
```

A screenshot of a Windows command prompt window. The title bar reads "C:\WINDOWS\System32\cmd.exe". The command prompt shows the user navigating to the "C:\eclipse" directory and running the command "eclipse.exe -vm C:\j2sdk1.4.1\_01\jre\bin\javaw.exe". The prompt then returns to "C:\eclipse>".



# Eclipse Installation

---

- For more details on running eclipse visit this:  
[http://help.eclipse.org/help21/index.jsp?topic=/org.eclipse.platform.doc.user/tasks/running\\_eclipse.htm](http://help.eclipse.org/help21/index.jsp?topic=/org.eclipse.platform.doc.user/tasks/running_eclipse.htm)
- The eclipse site <http://www.eclipse.org>

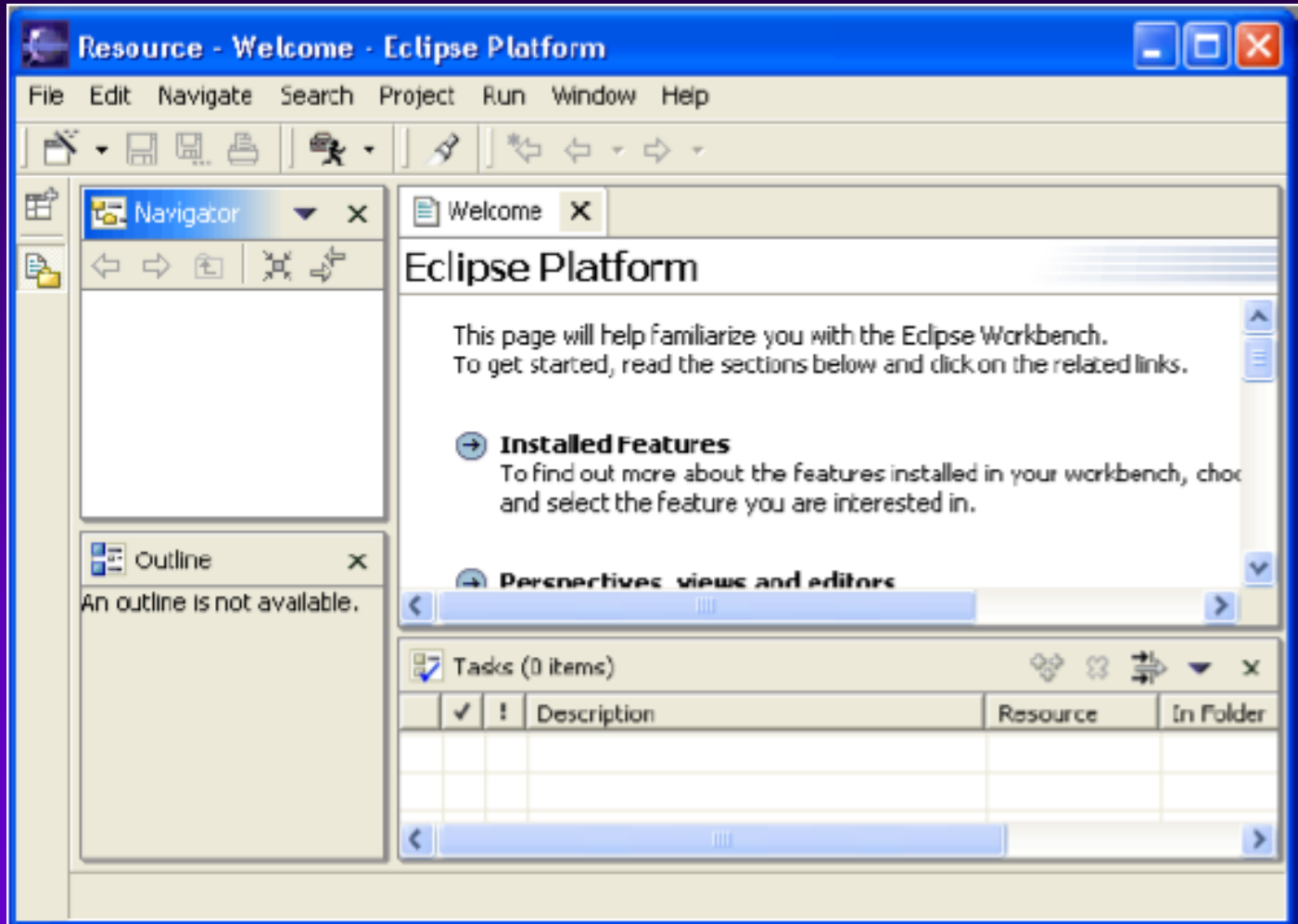


# Part2

Workbench



# Workbench





# Workbench

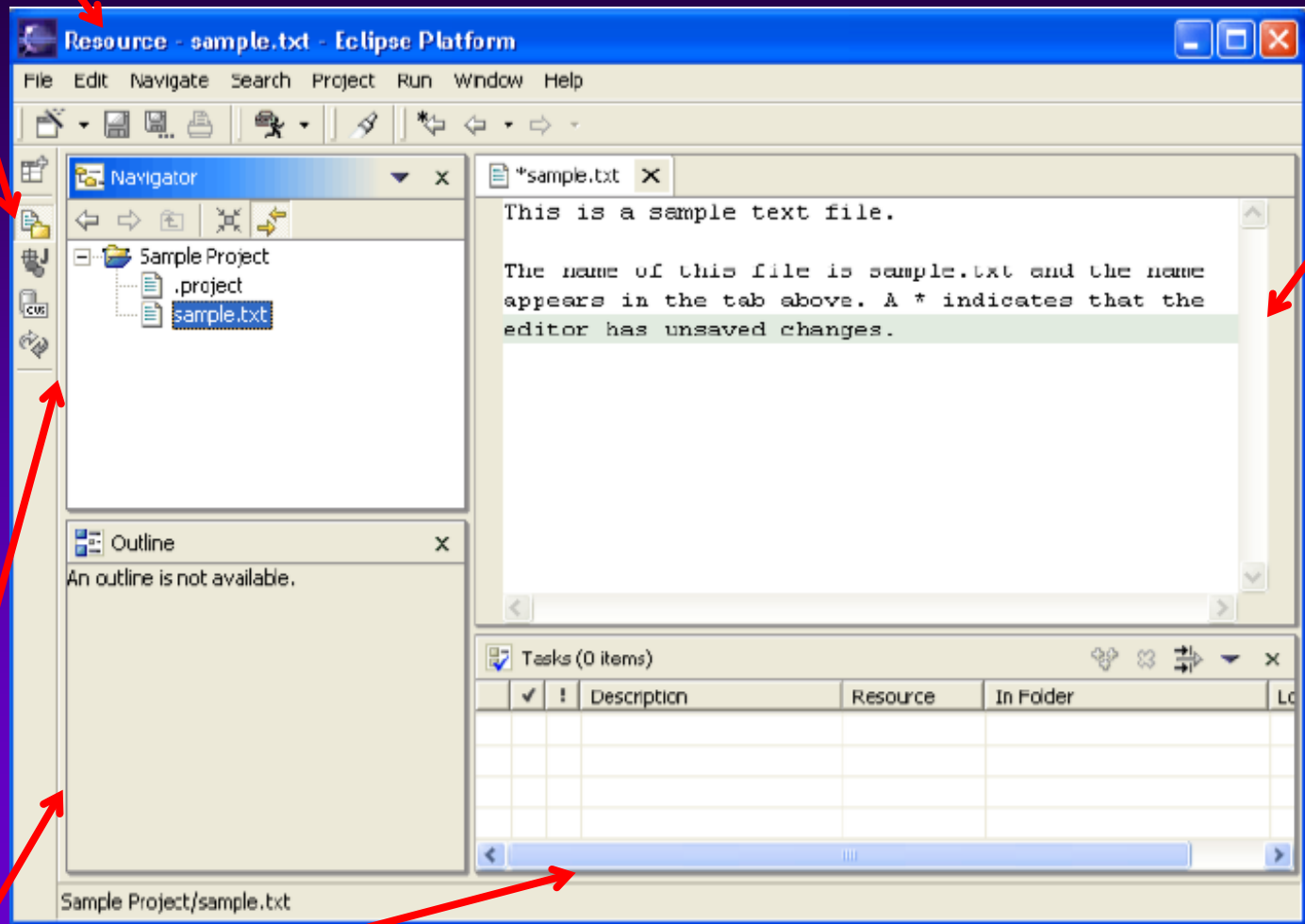
---

- Editors
  - A visual component within the workbench
  - Used to edit or browse a resource
- Views
  - A visual component within the workbench
  - Used to navigate a hierarchy of information, open an editor or display properties of an active editor
- Perspectives
  - Groups of views and editors within the workbench

# A simple example...

perspective

editor



views



# Creating resources

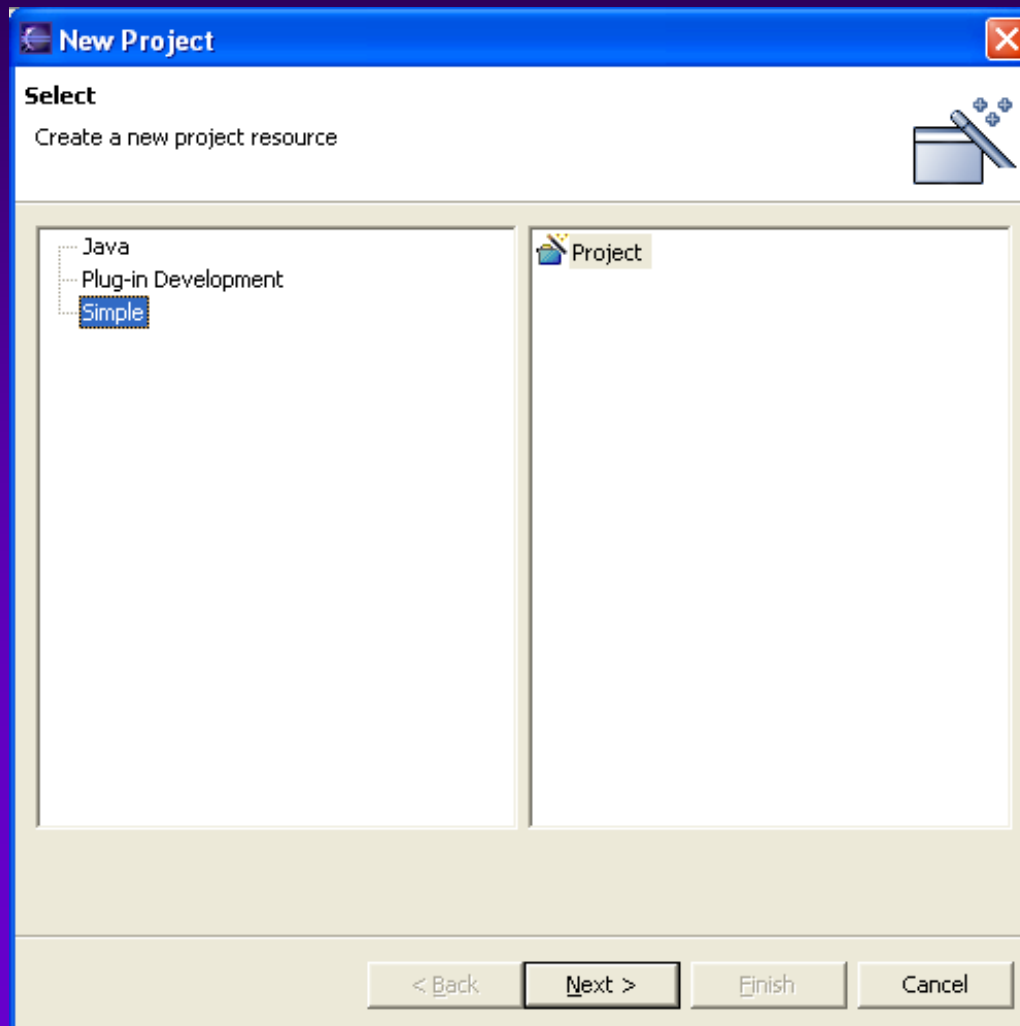
---

- Three alternatives
  1. File menu
  2. Navigator context menu (right-click on selected resource)
  3. New Wizard button



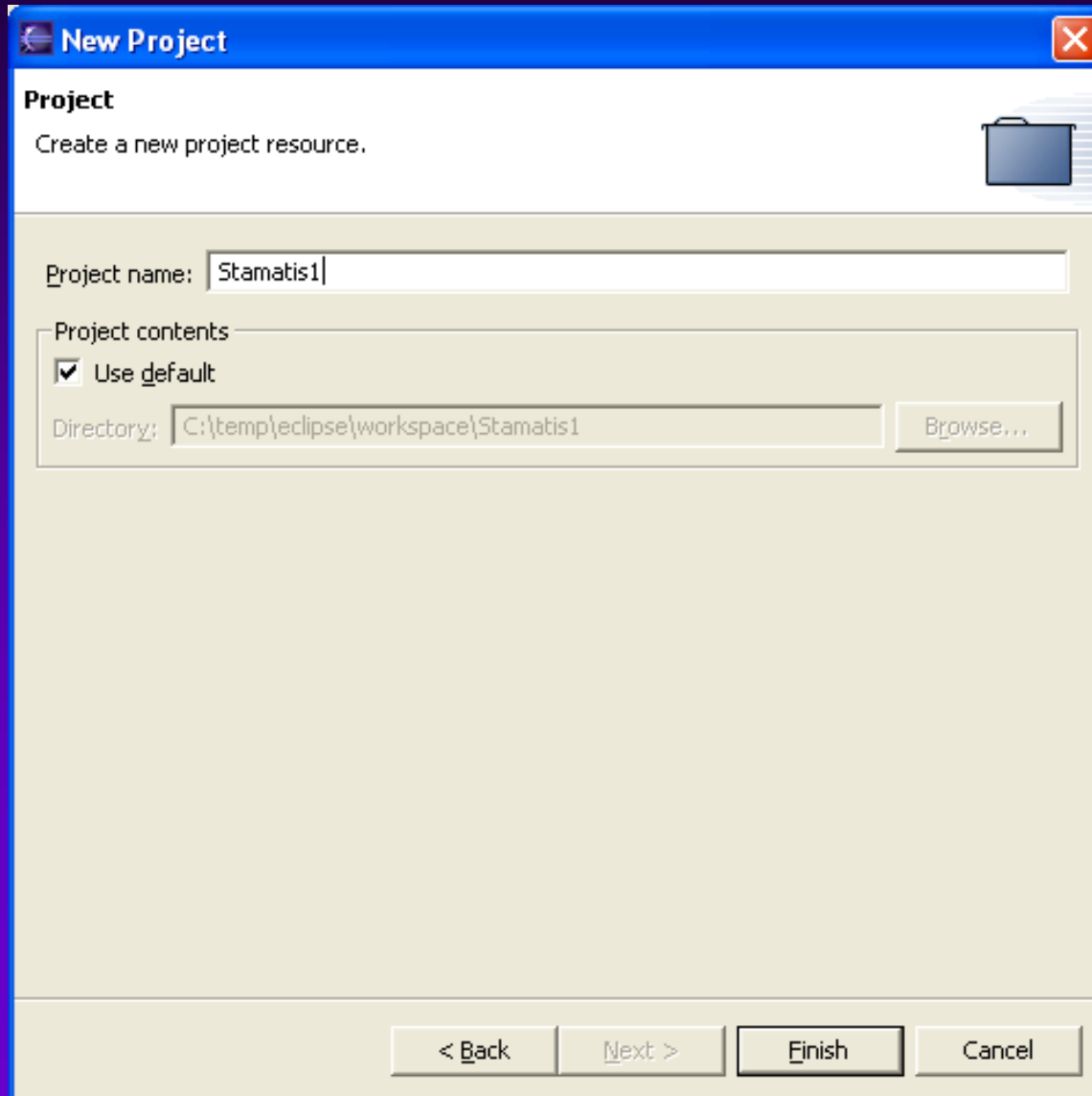
# Creating a new project

- From the menubar, select File > New > Project...

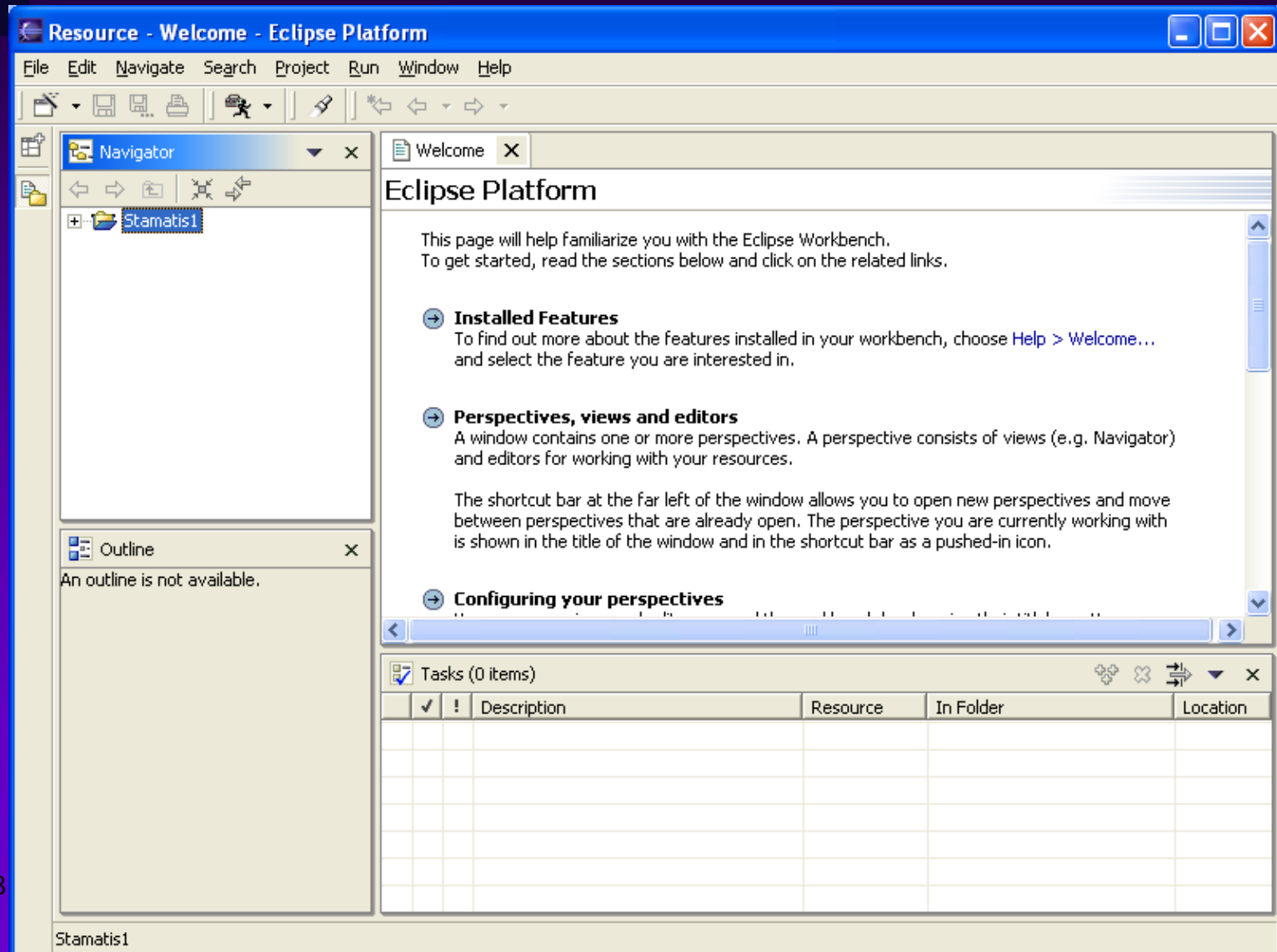




# Creating a new project



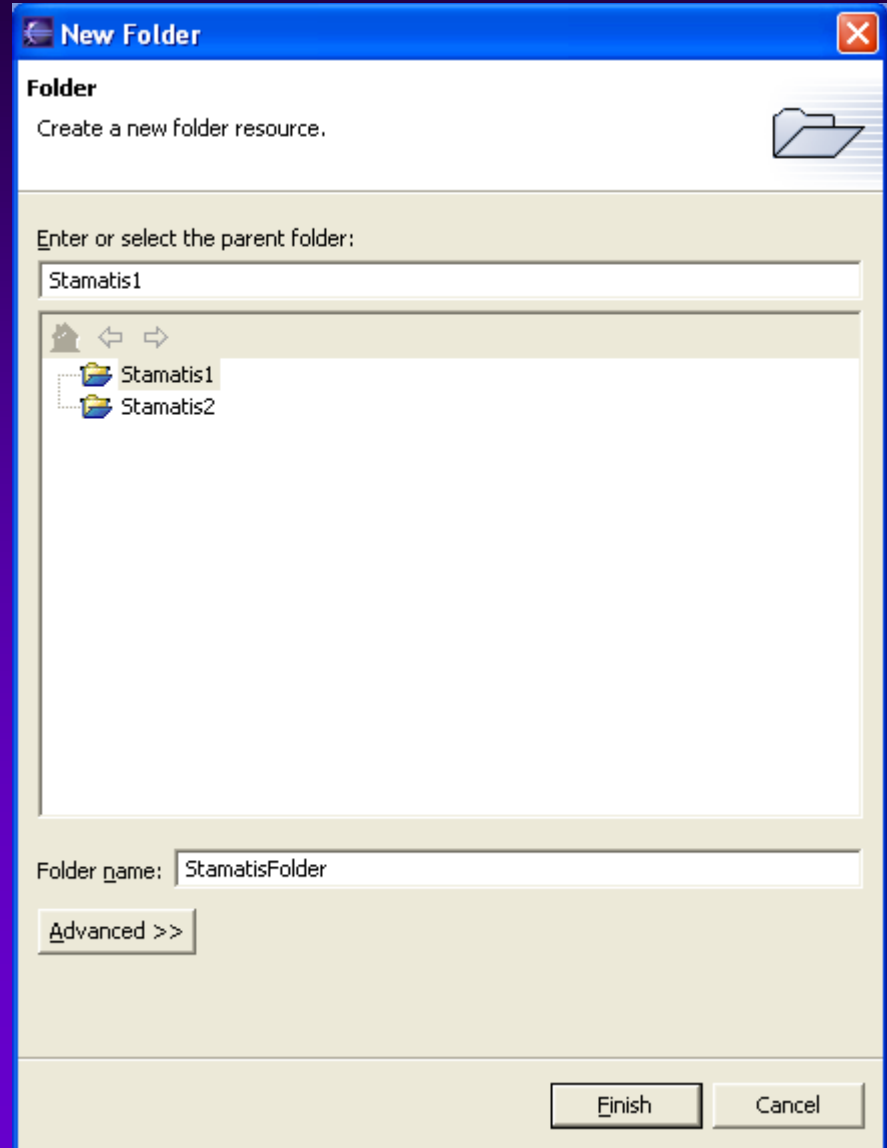
# Creating a new project



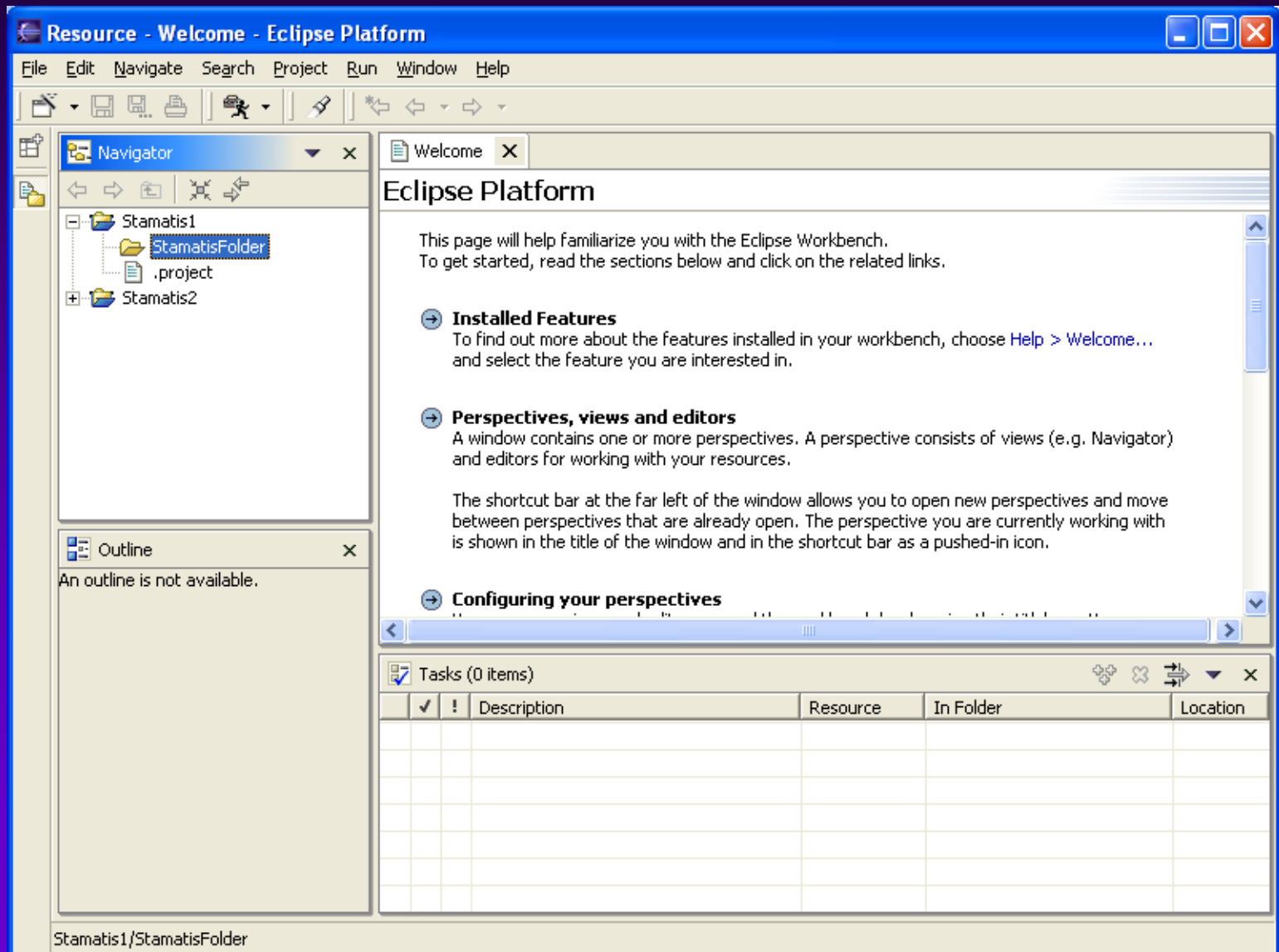


# Creating a new folder

- From the popup menu of a project in the navigation view, select New > Folder



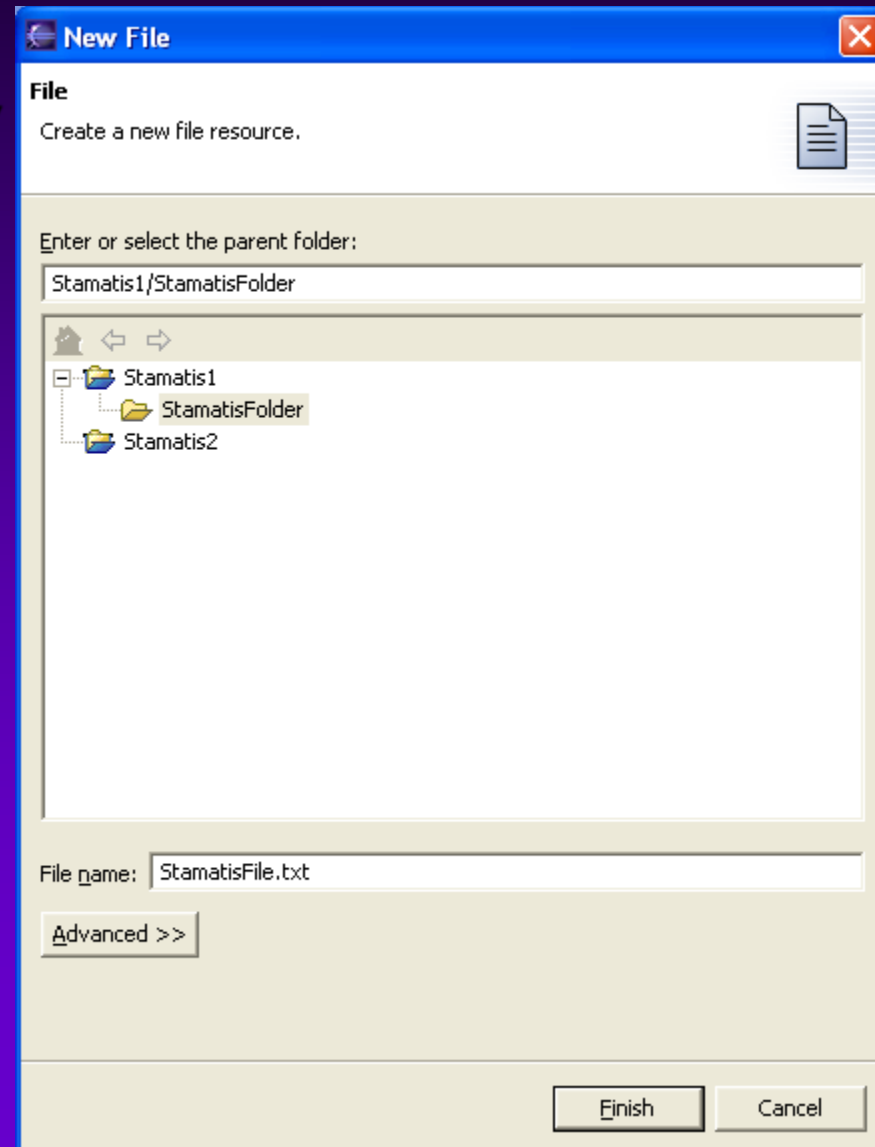
# Creating a new folder



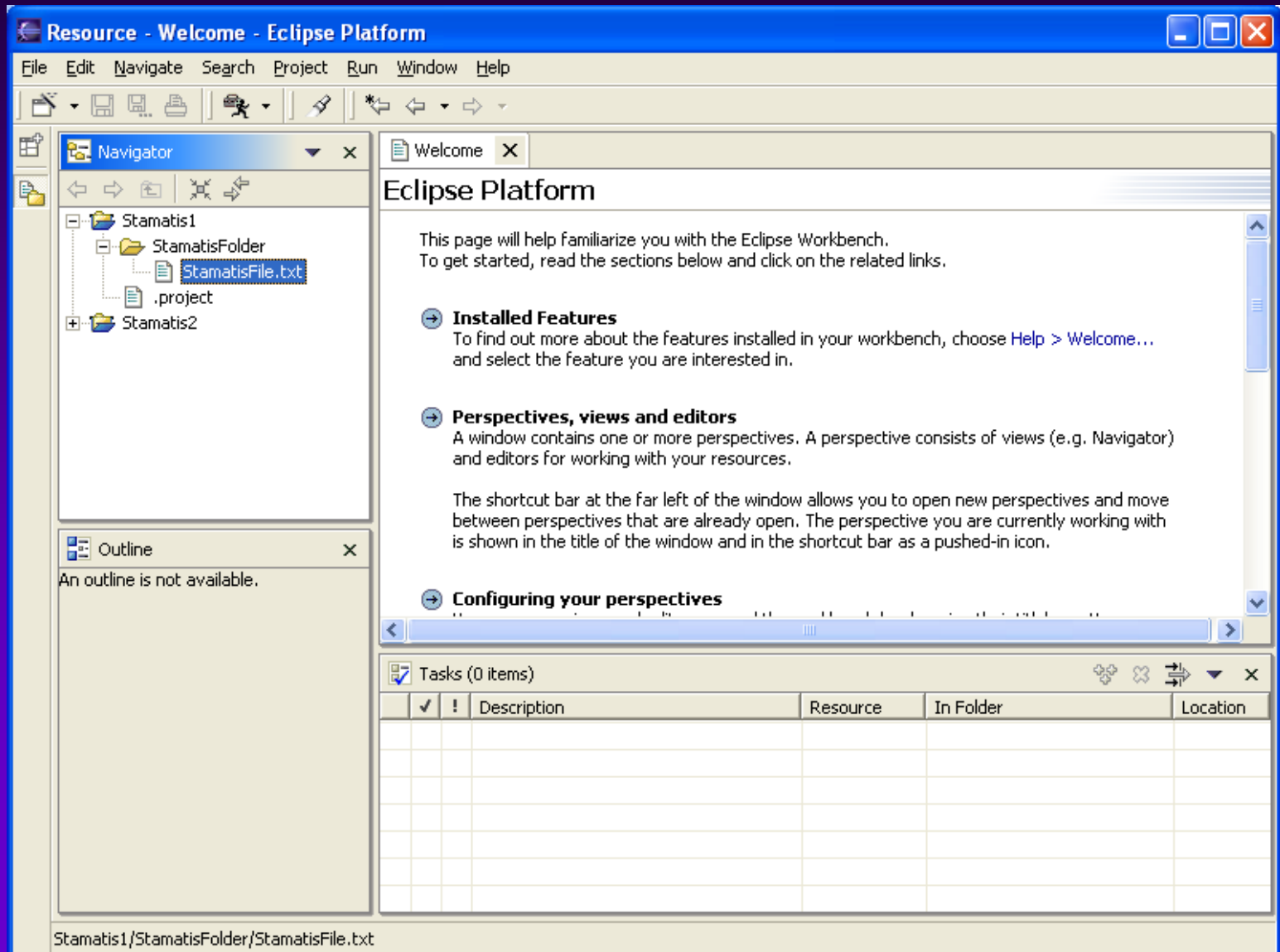


# Creating a new file

- From the toolbar use the "New" Wizard button and select "File"



# Creating a new file



The screenshot shows the Eclipse IDE interface. The title bar reads "Resource - Welcome - Eclipse Platform". The menu bar includes File, Edit, Navigate, Search, Project, Run, Window, and Help. The toolbar contains icons for file operations like New, Open, Save, and Print.

The left sidebar contains two views:
 

- Navigator:** Shows a project tree with "Stamatis1" containing a "StamatisFolder" which has a new file "StamatisFile.txt" being created. Below it is a ".project" file and another folder "Stamatis2".
- Outline:** Displays the message "An outline is not available."

The main editor area shows the "Welcome" page for the "Eclipse Platform". It contains the following text:
 

This page will help familiarize you with the Eclipse Workbench. To get started, read the sections below and click on the related links.

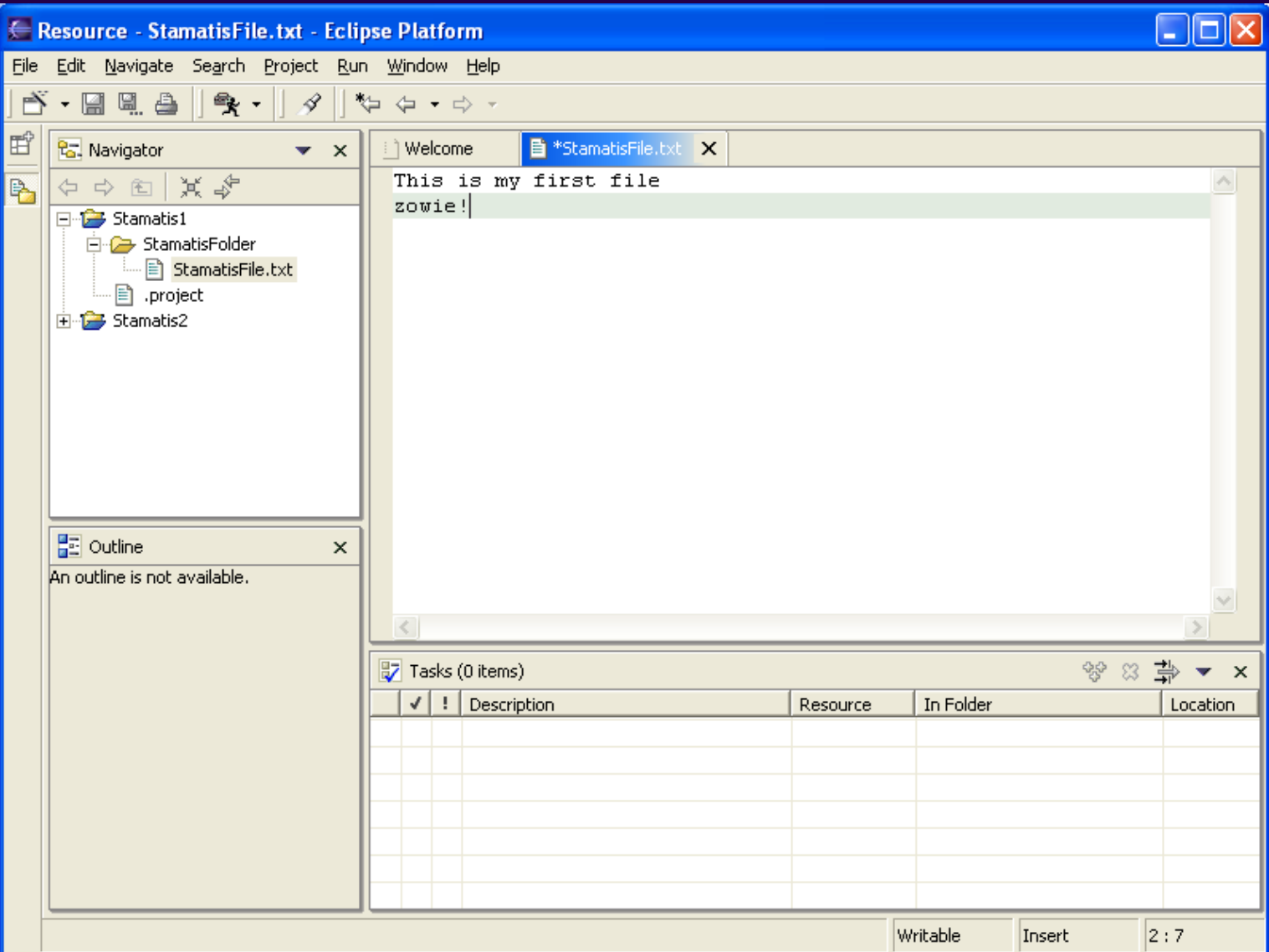
- Installed Features**  
To find out more about the features installed in your workbench, choose [Help > Welcome...](#) and select the feature you are interested in.
- Perspectives, views and editors**  
A window contains one or more perspectives. A perspective consists of views (e.g. Navigator) and editors for working with your resources.  
  
The shortcut bar at the far left of the window allows you to open new perspectives and move between perspectives that are already open. The perspective you are currently working with is shown in the title of the window and in the shortcut bar as a pushed-in icon.
- Configuring your perspectives**

At the bottom, the "Tasks" view is empty, showing a table with columns: Description, Resource, In Folder, and Location.

The status bar at the bottom of the window displays the path: "Stamatis1/StamatisFolder/StamatisFile.txt".



# Creating a new file

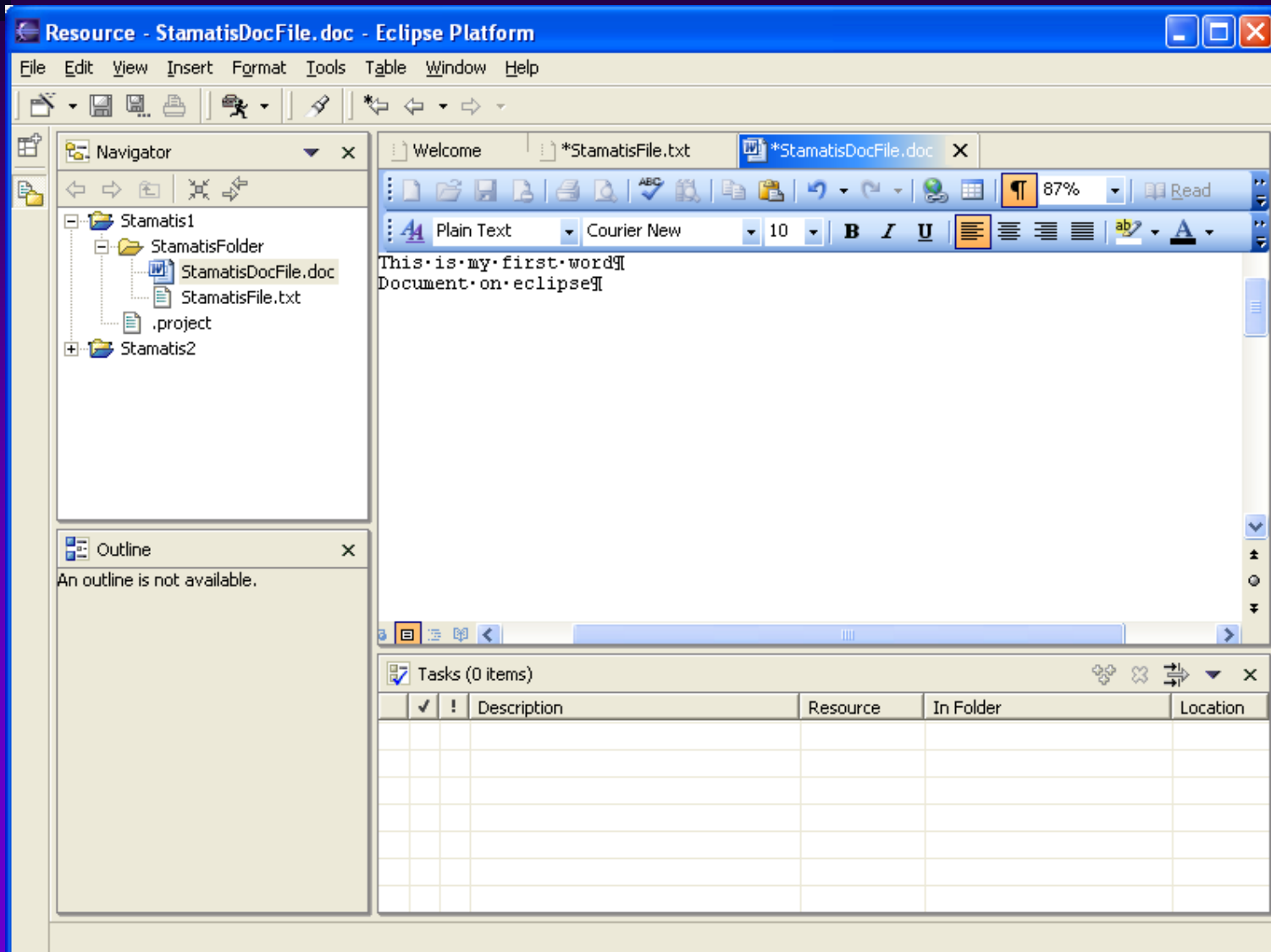


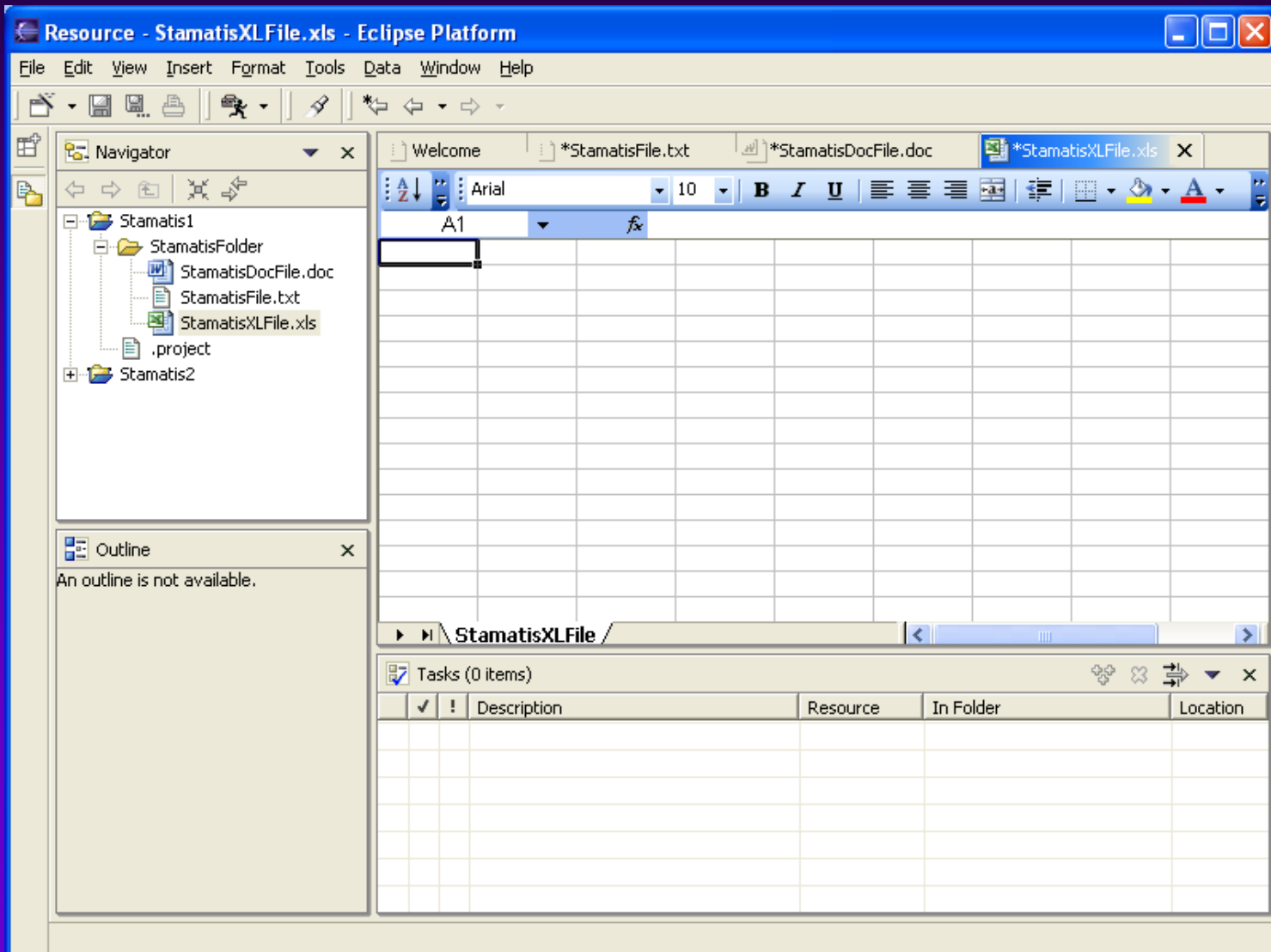


# Editors

---

- Editors are tiled using tabs (if there is more than one file opened)
- The appropriate editor is used according to the type of file

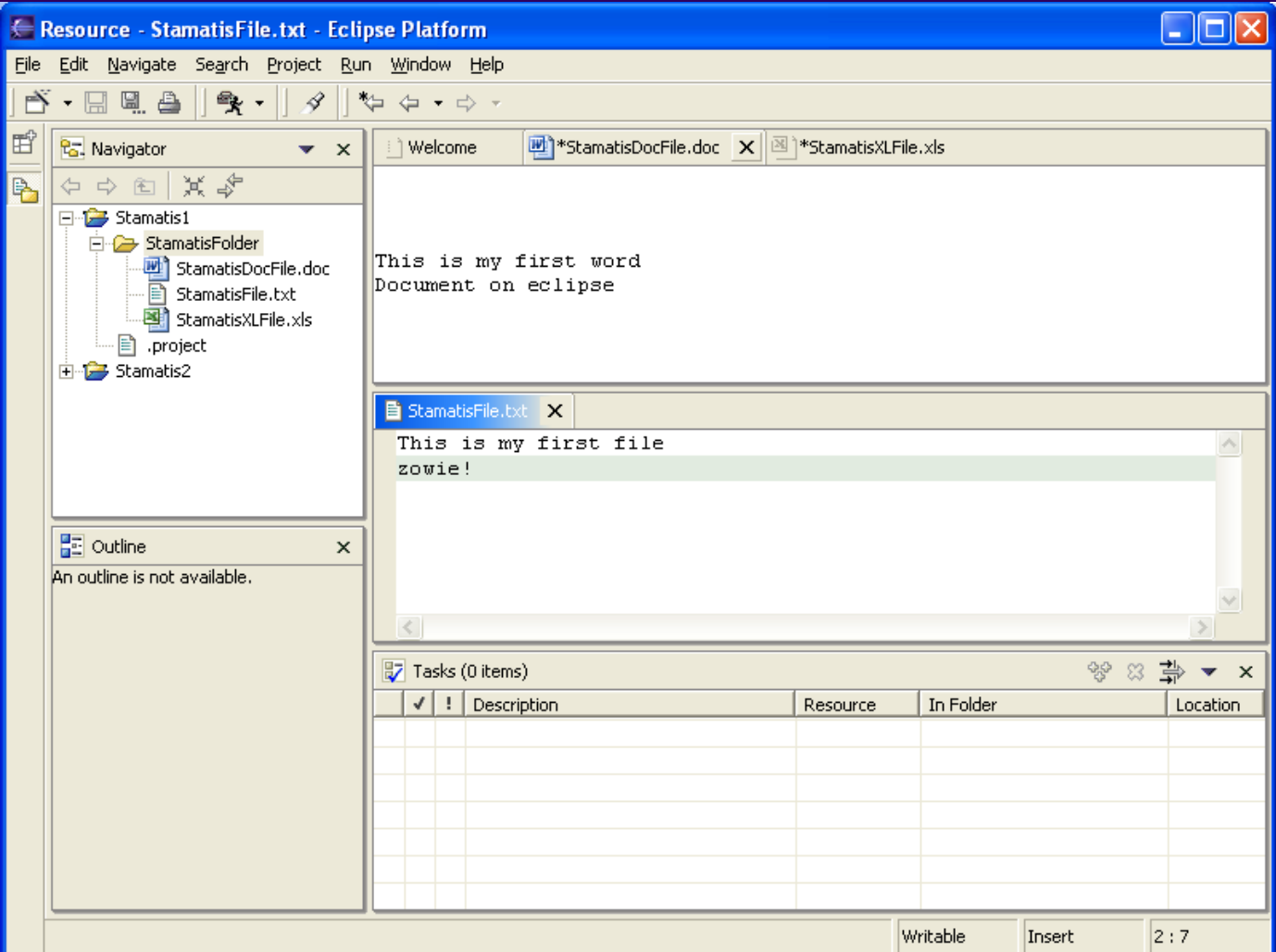






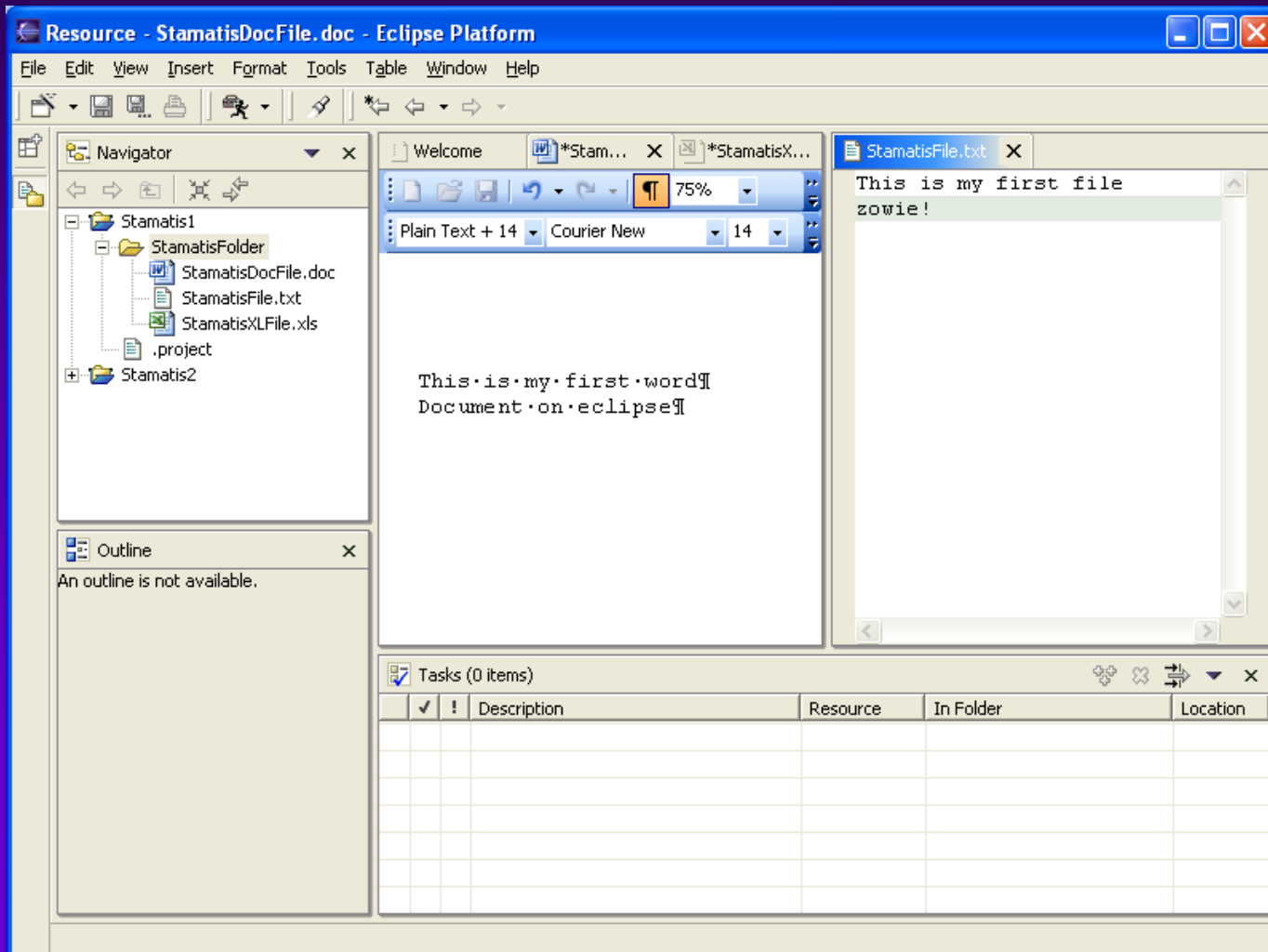
# Stack editors

- Possible both side by side and one above the other



# Stack editors

- Possible both side by side and one above the other





# Views

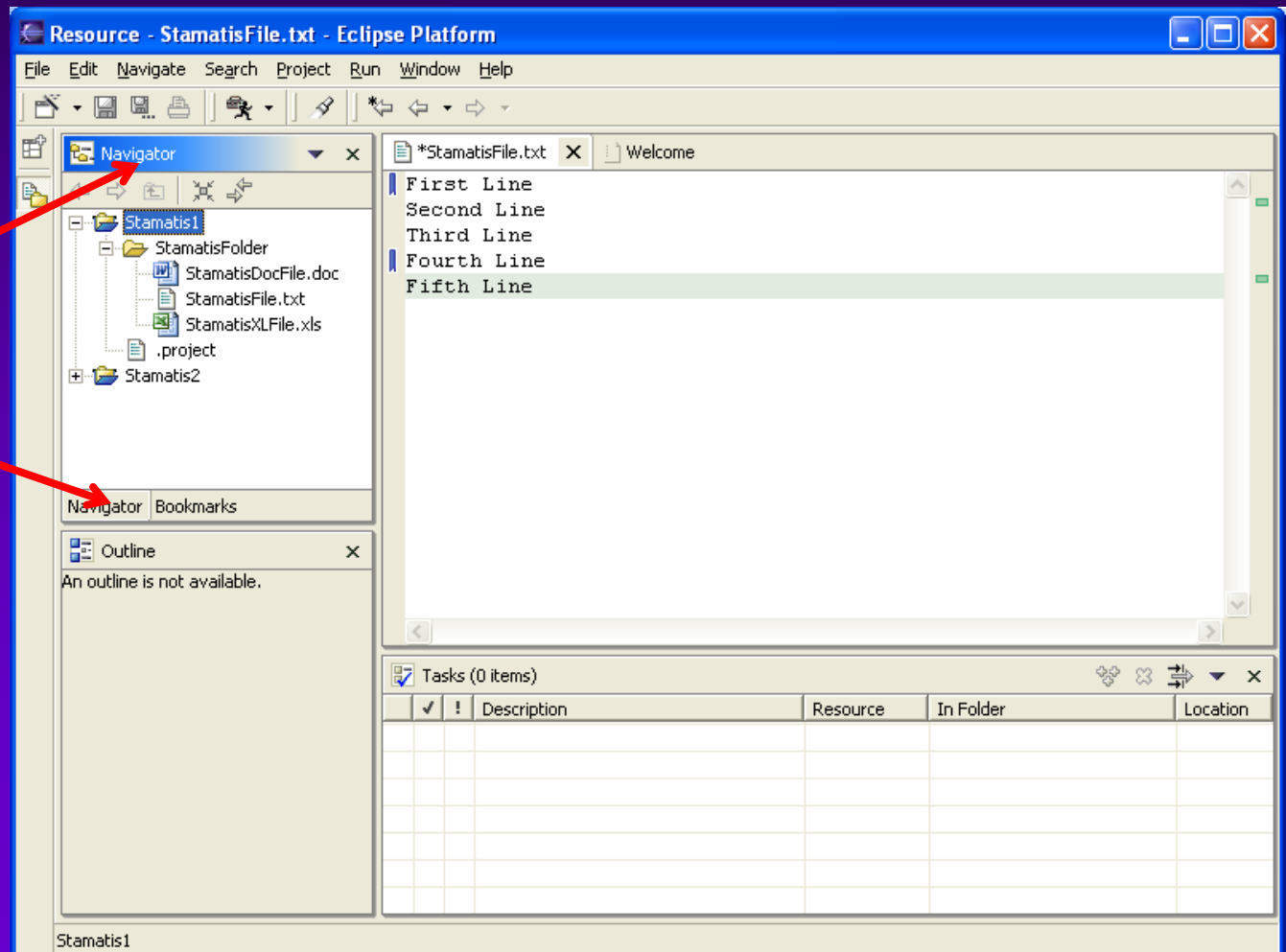
---

- Views help you work with resources in addition to editors



# Views (example)

- The Navigator view displays the projects and other resources that you are working with

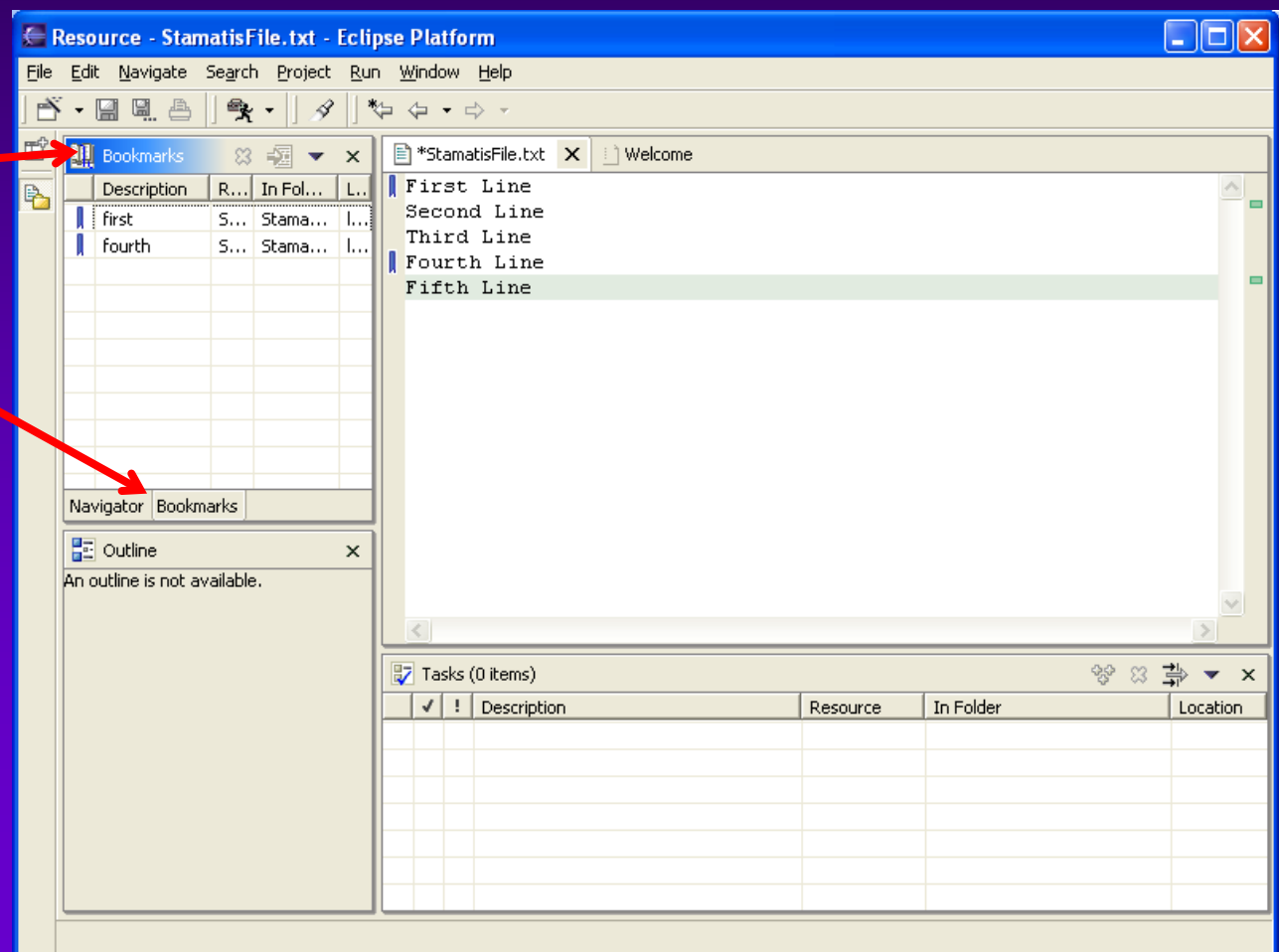




# Views (example)

- The bookmark view displays all bookmarks in the workbench along with the name of the file the bookmark is associated

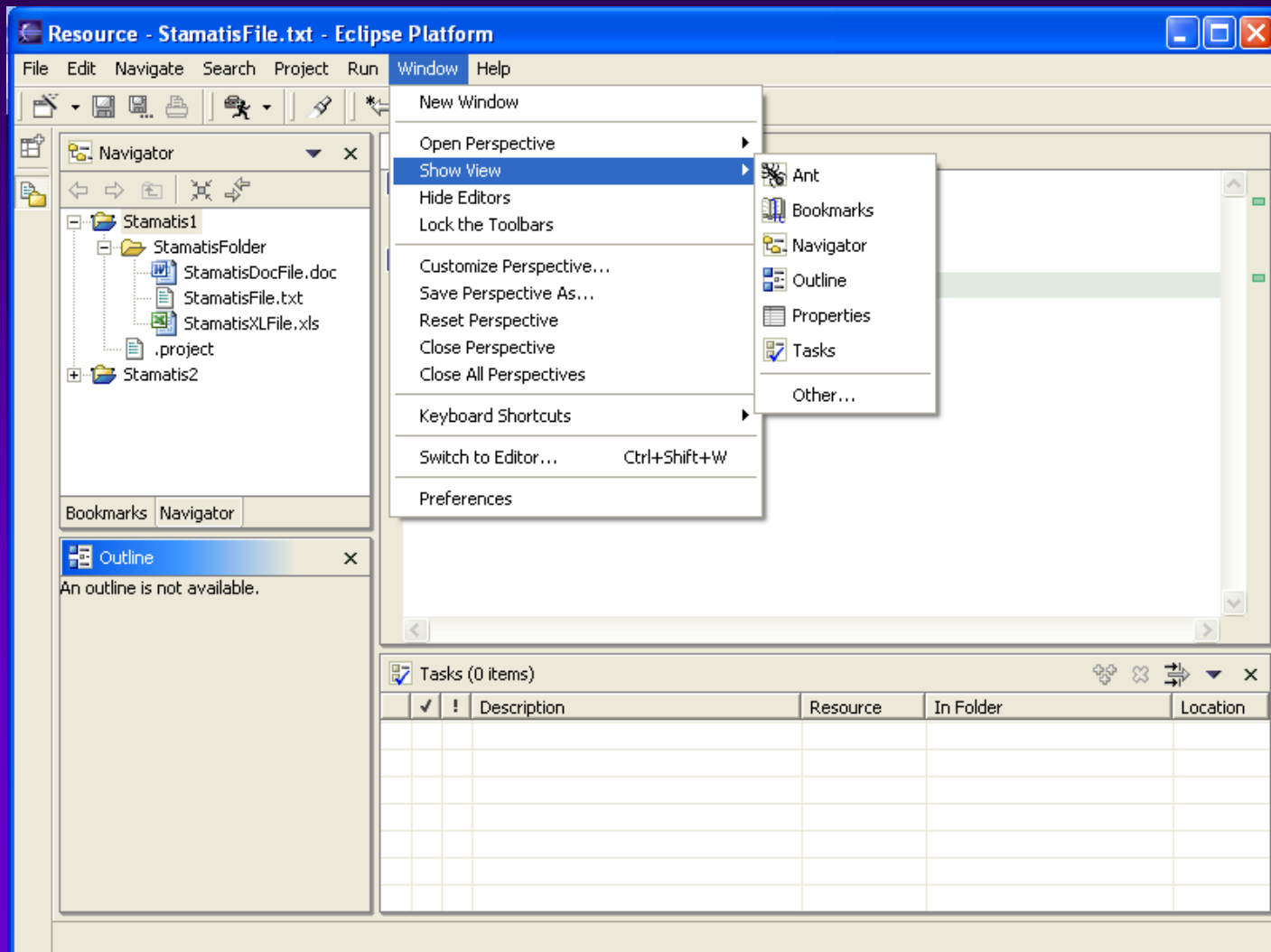
Bookmark view





# Views

- Display views from Window > Show View menu





# Views

---

- Experiment as much as you want with the views
- Rearrange the Workbench using the Window > Reset Perspective menu



# Show me the files...

---

- Q: Where are my files actually stored?
- A:
  1. Open your file system explorer
  2. Navigate to the location where you installed eclipse (%ECLIPSE\_HOME%). You will find a sub-directory called "workspace"
  3. You will find the projects, folders and files in that sub-directory

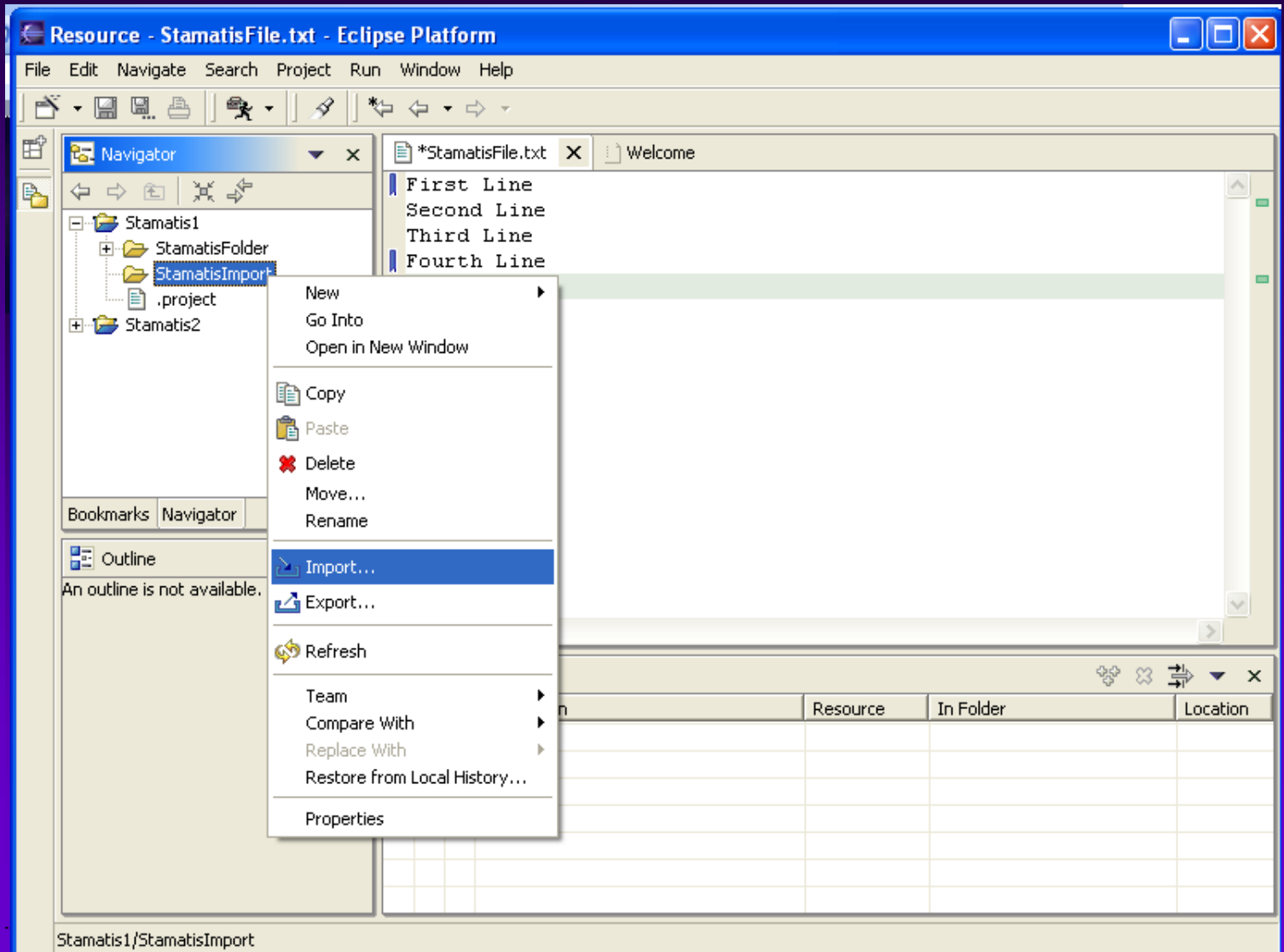


# Import files

---

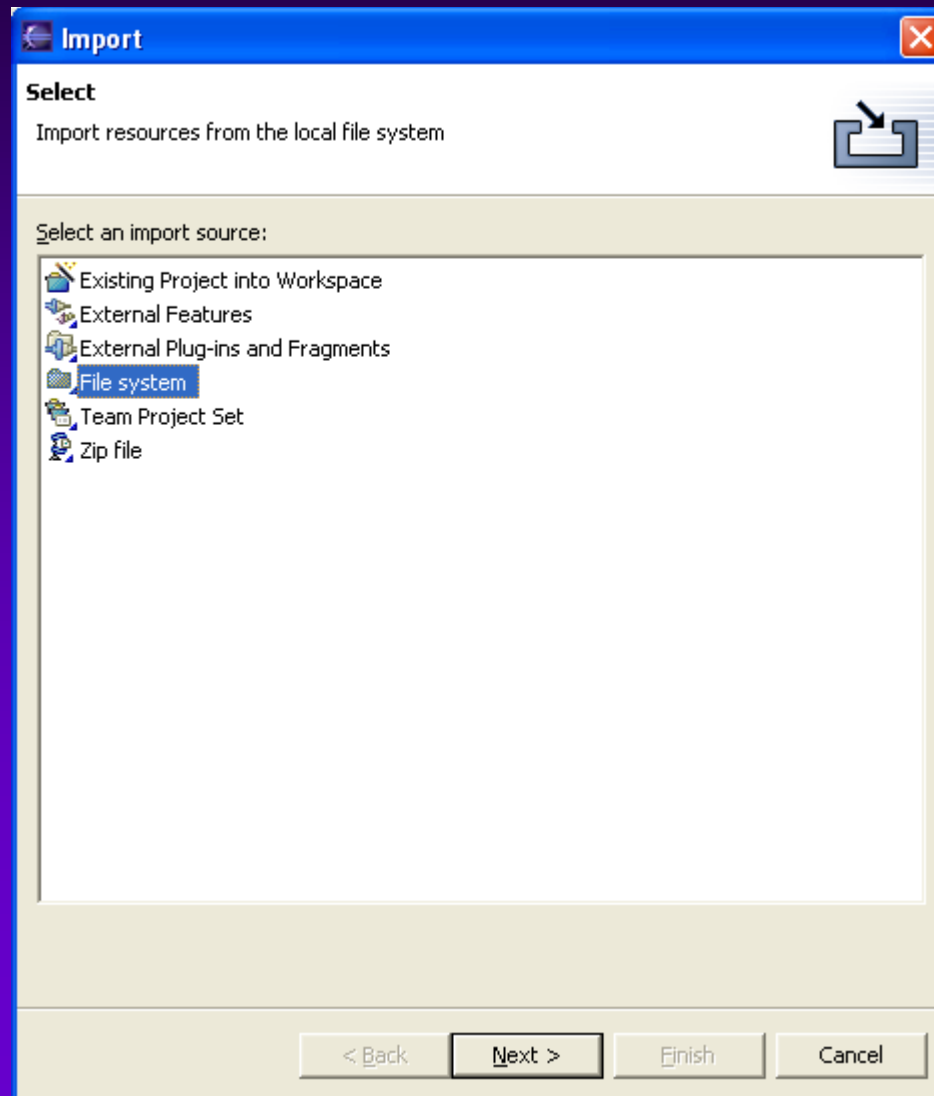
1. Drag and Drop files from the file system
2. Copy and Paste files from the file system
3. Use the import wizard either from the popup menu of the navigator or from the File menu

# Import files

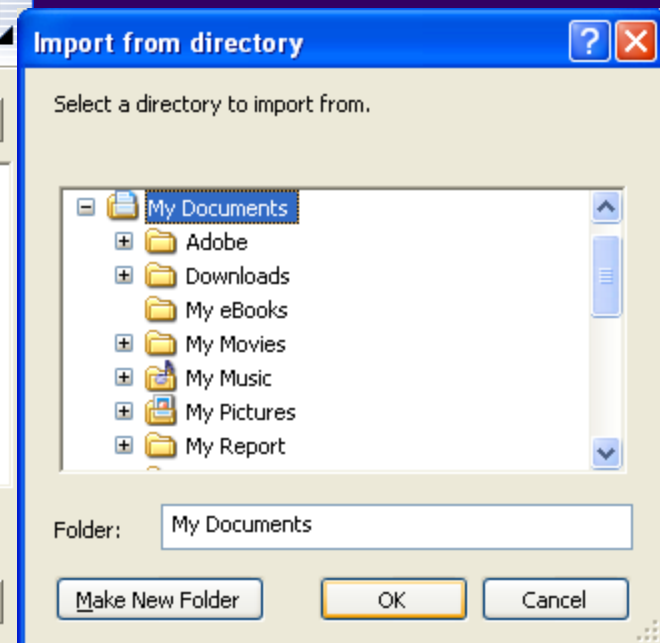
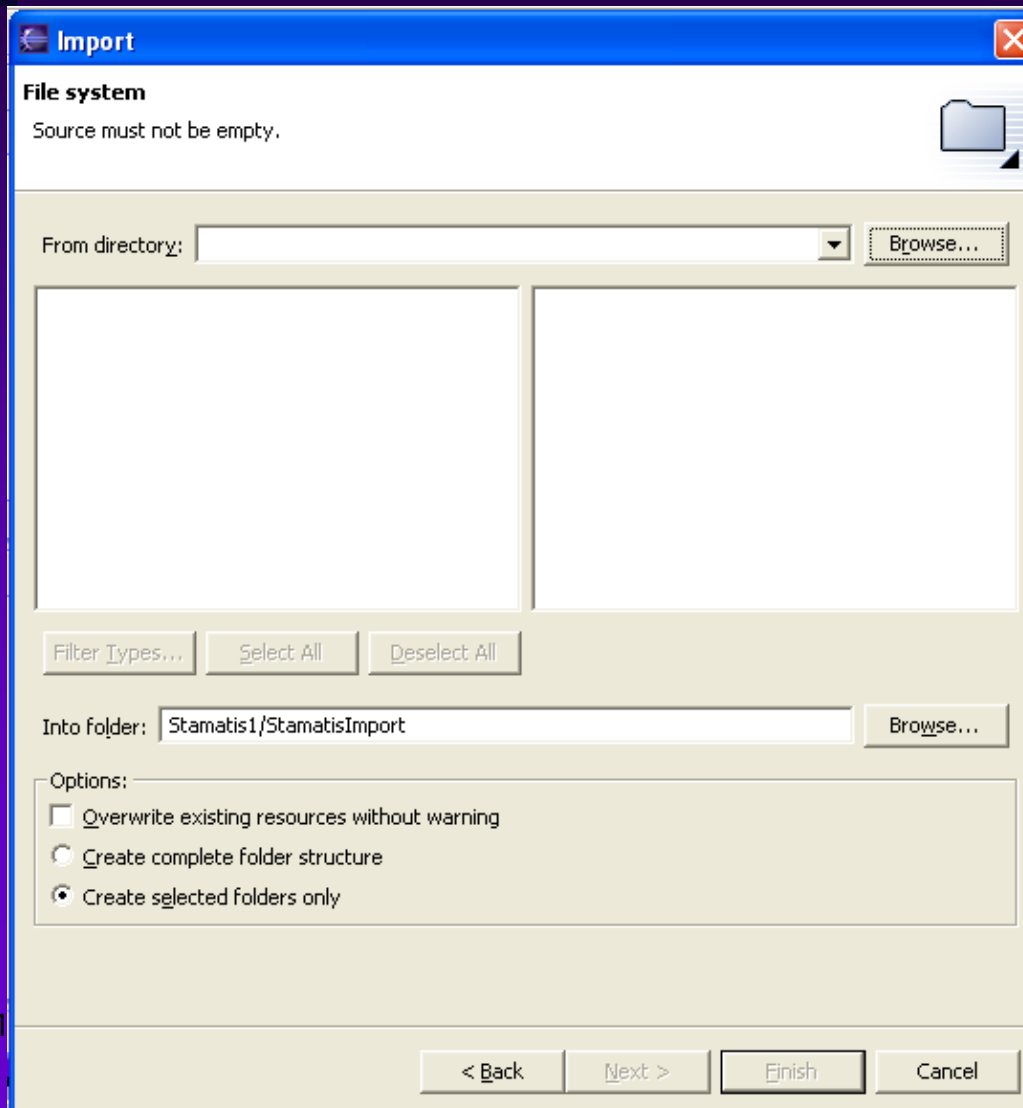




# Import files

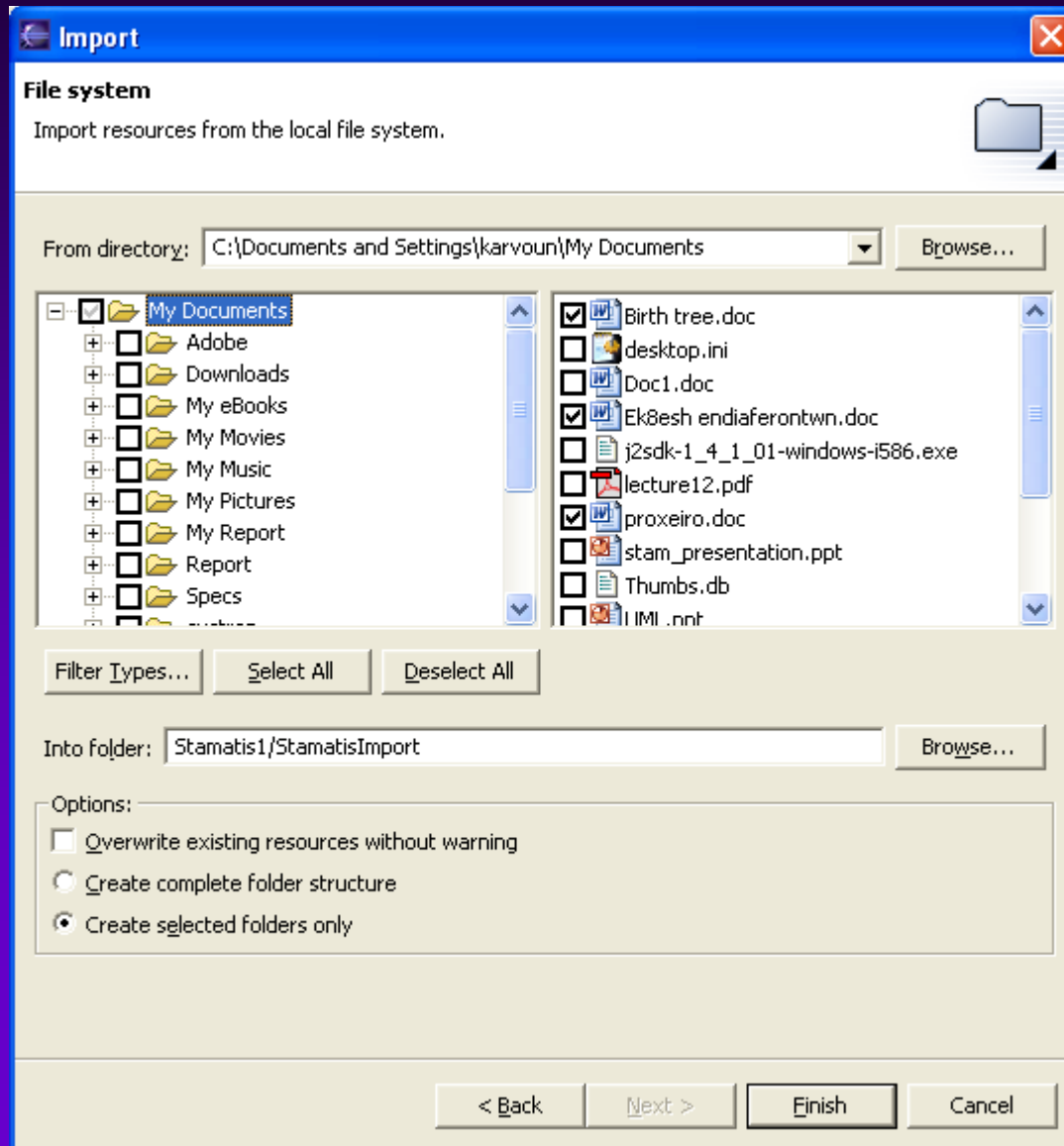


# Import files





# Import files





# Import files

The screenshot shows the Eclipse IDE interface. The title bar reads "Resource - StamatisFile.txt - Eclipse Platform". The menu bar includes "File", "Edit", "Navigate", "Search", "Project", "Run", "Window", and "Help". The Navigator on the left shows a project structure with "Stamatis1" containing "StamatisFolder", "StamatisImport", "Birth tree.doc", "Ek8esh endiaferontwn.d", "proxeiro.doc", and ".project". "Stamatis2" is also visible. The Editor window shows the contents of "StamatisFile.txt" with five lines of text: "First Line", "Second Line", "Third Line", "Fourth Line", and "Fifth Line". The Outline view shows "An outline is not available." The Tasks view shows "Tasks (0 items)" with a table structure.

	✓	!	Description	Resource	In Folder	Location



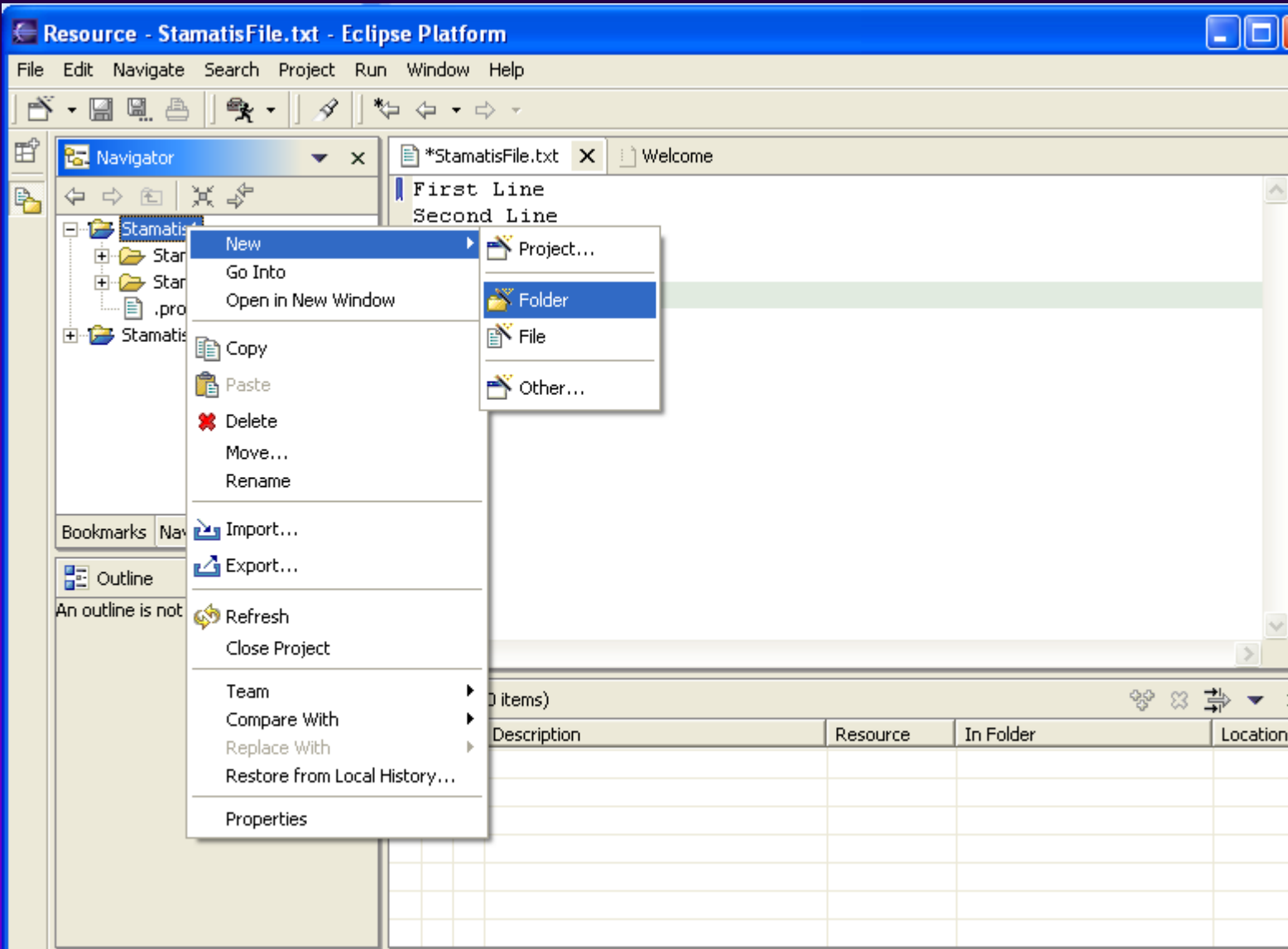
# Link folder to file system

---

- If you do not want the contents of the folder to be copied in the "workspace" you have to "link" the folder to a position in the file system upon its creation



# Link folder to file system





# Link folder to file system

---

- Click "Advanced"
- Check "Link to folder in file system"
- Browse to find the desired folder

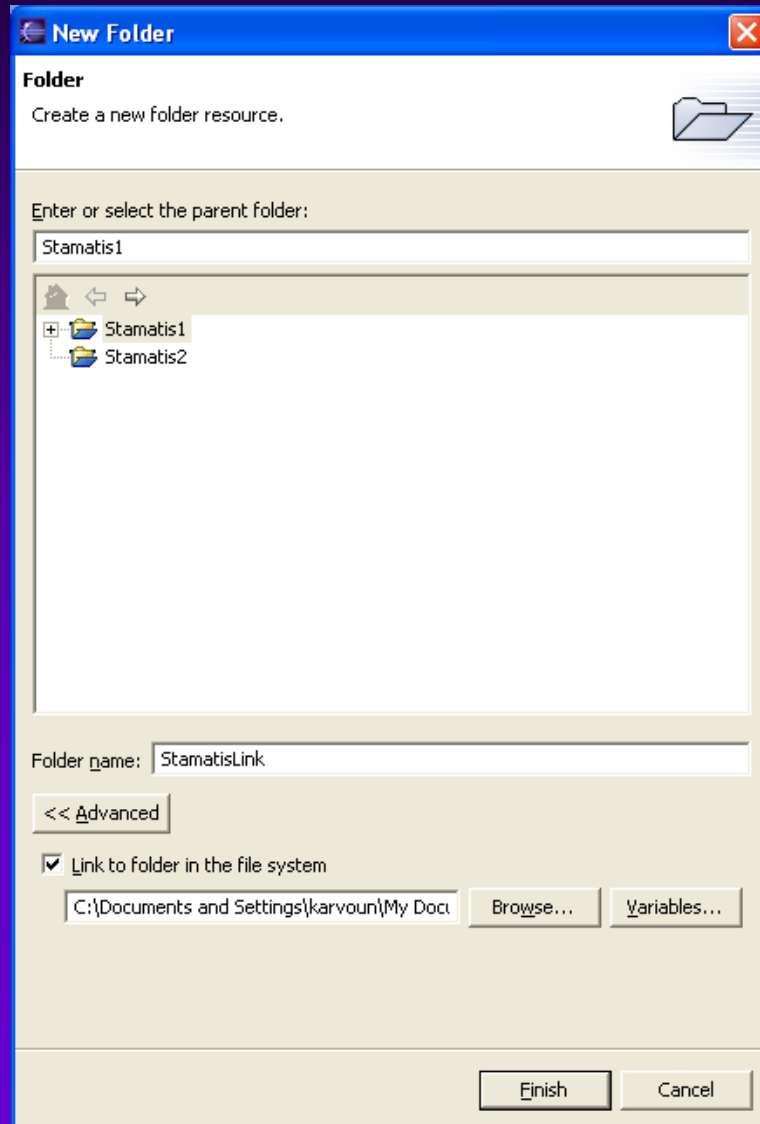


# Link folder to file system

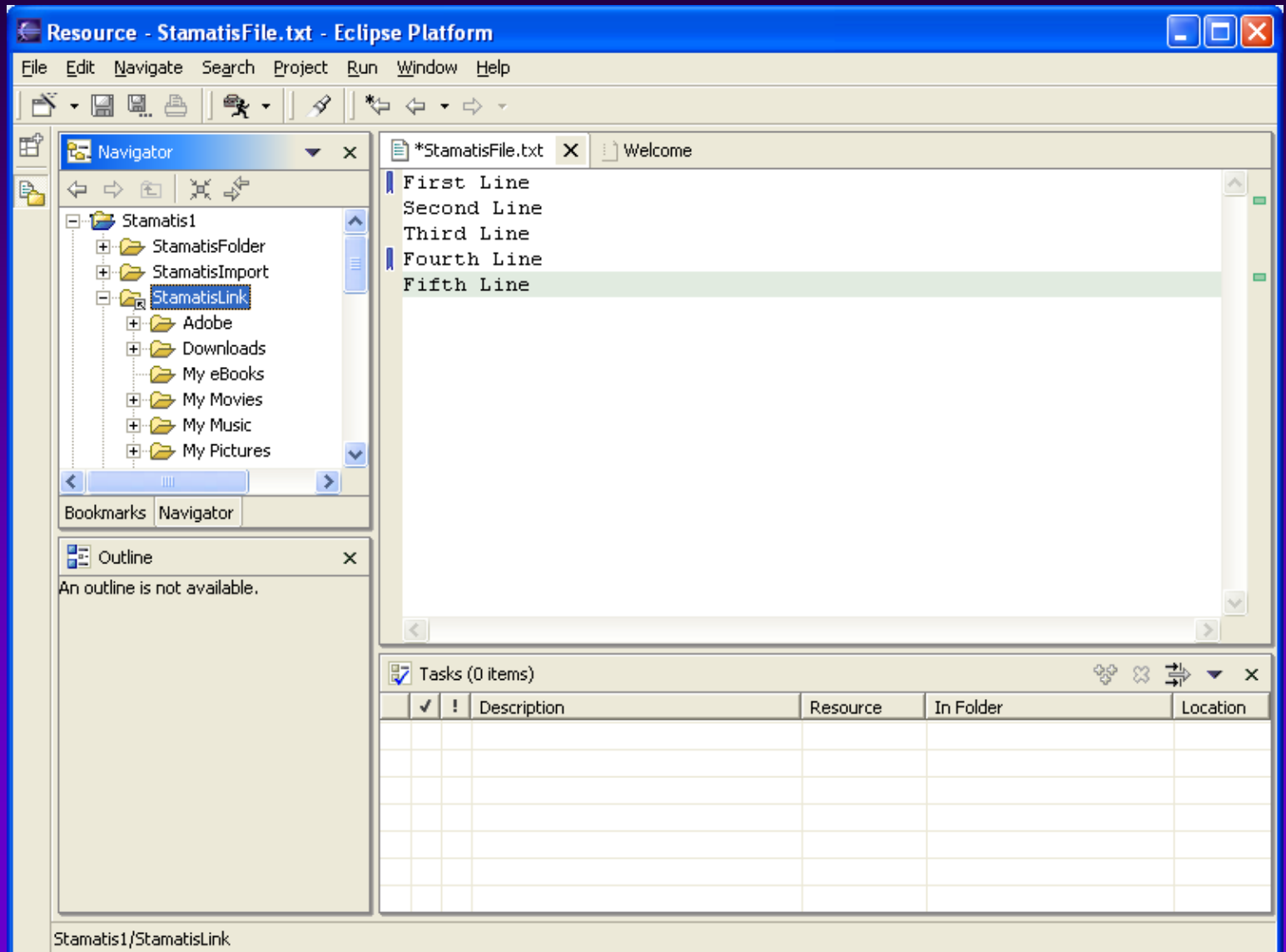




# Link folder to file system



# Link folder to file system



The screenshot shows the Eclipse IDE interface with a project named "Stamatis1". The Navigator view on the left shows a folder structure where "StamatisLink" is highlighted. This folder is linked to a file system location, as indicated by the "Stamatis1/StamatisLink" text at the bottom of the IDE. The main editor window displays the contents of the linked folder, which is a text file named "\*StamatisFile.txt" containing five lines of text:

```

First Line
Second Line
Third Line
Fourth Line
Fifth Line
    
```

The Tasks view at the bottom shows a table with the following columns: Description, Resource, In Folder, and Location. The table is currently empty.

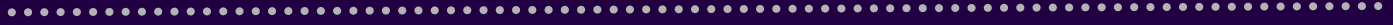
✓	!	Description	Resource	In Folder	Location



# Link folder to file system

---

- If you sneak in the file system explorer you will see that there is no folder "StamatisLink" in the "workspace" folder. Any change will be automatically made to the resources linked in the file system.



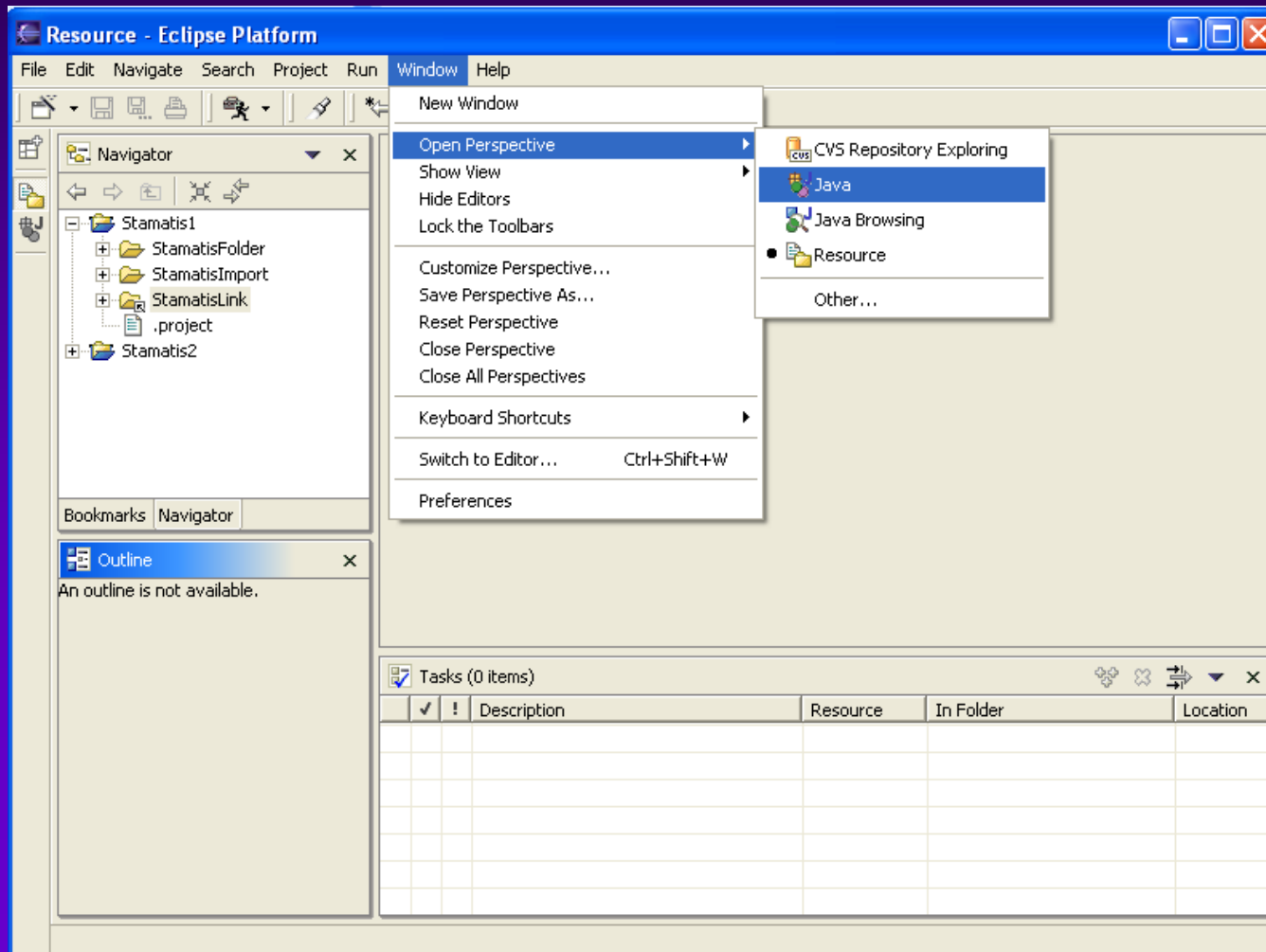
# Part3

Java Development

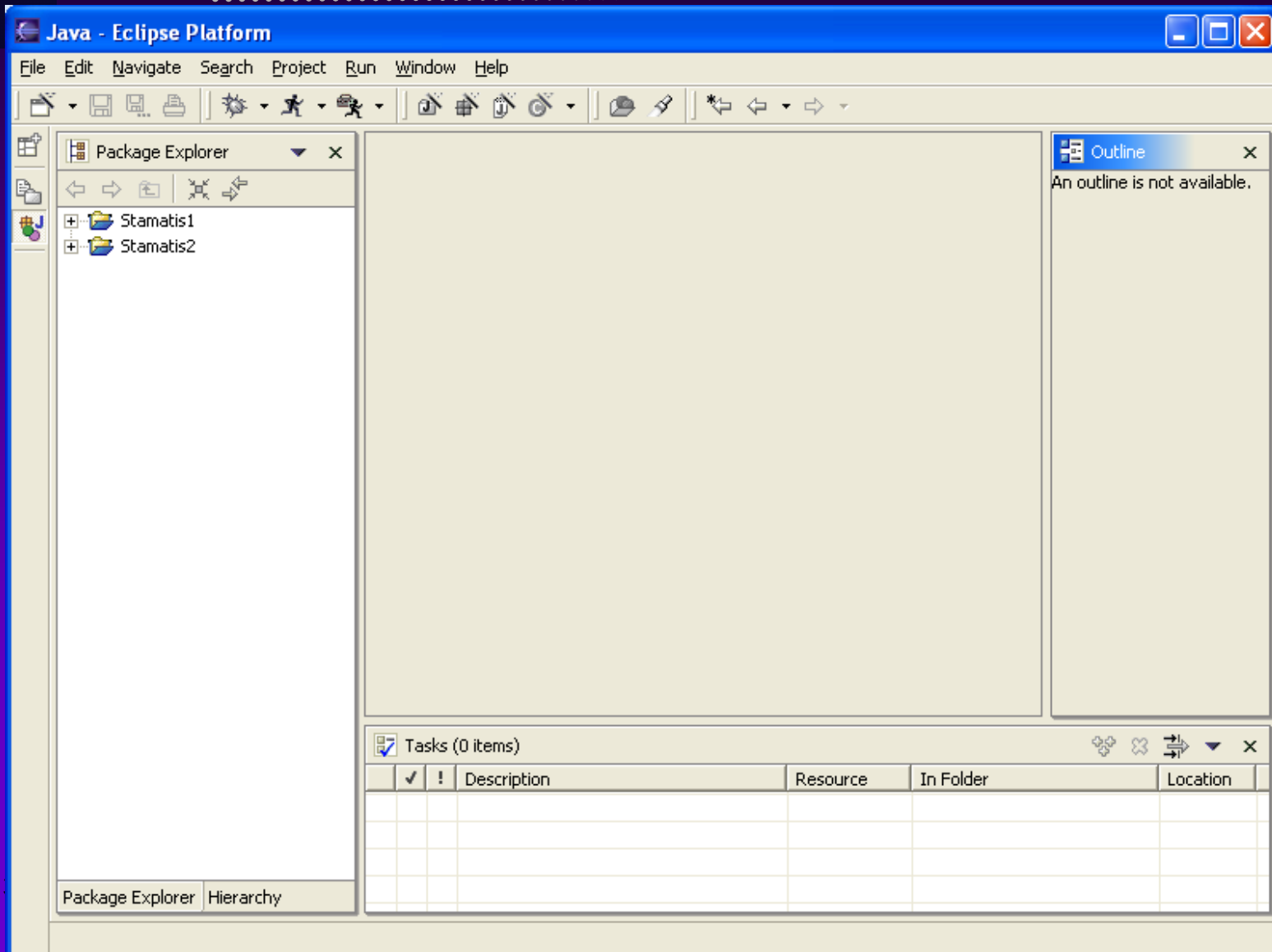


# Prepare Workbench

- Switch to Java Perspective using Window > Open Perspective > Java Menu



# Prepare Workbench





# Prepare Workbench

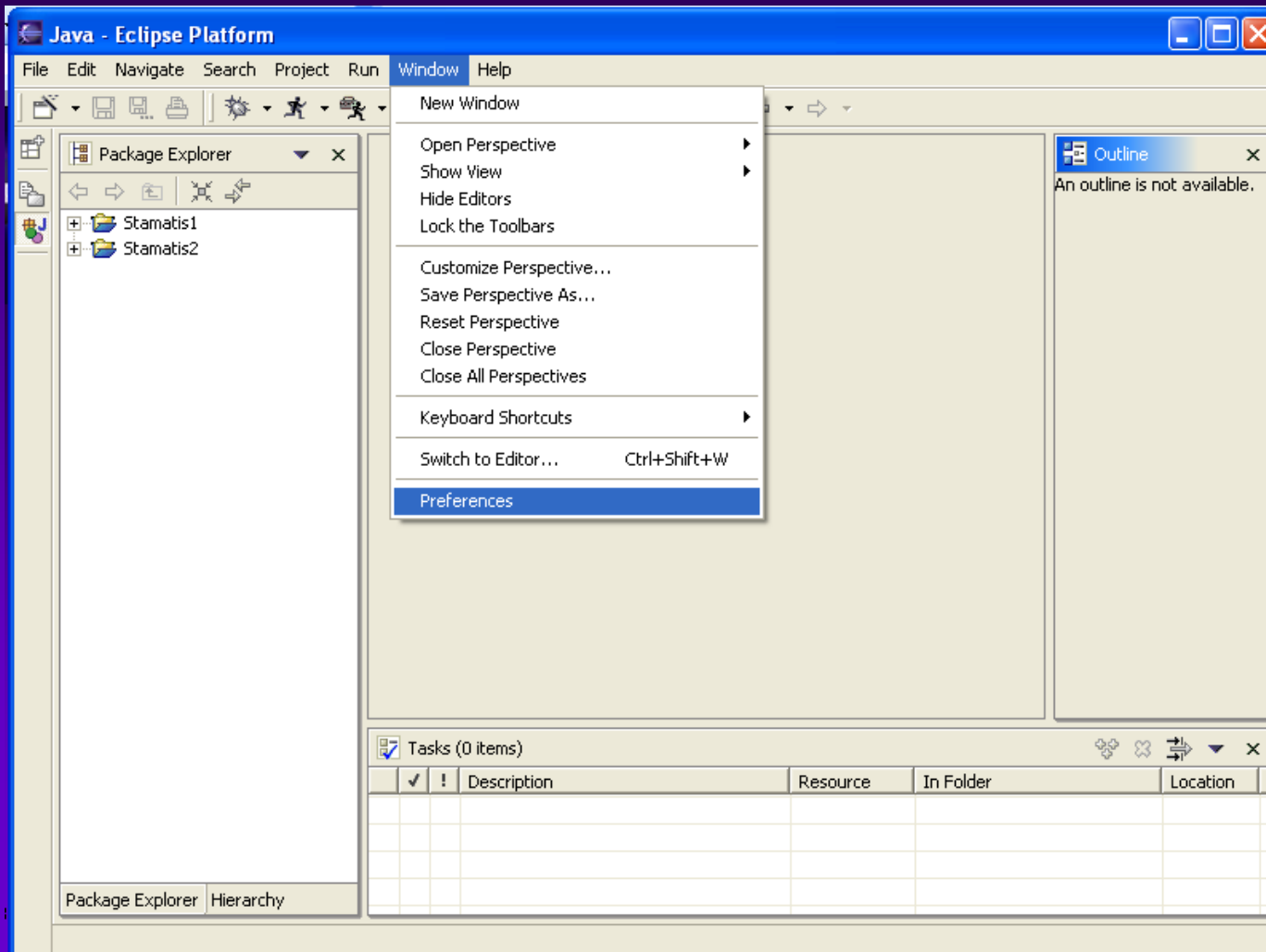
---

- Java perspective has the following views
  - Package Explorer
  - Hierarchy
  - Outline
  - Search
  - Console
  - Tasks



# Prepare Workbench

- Verify JRE installation and classpath variables
- Select menu Window > Preferences

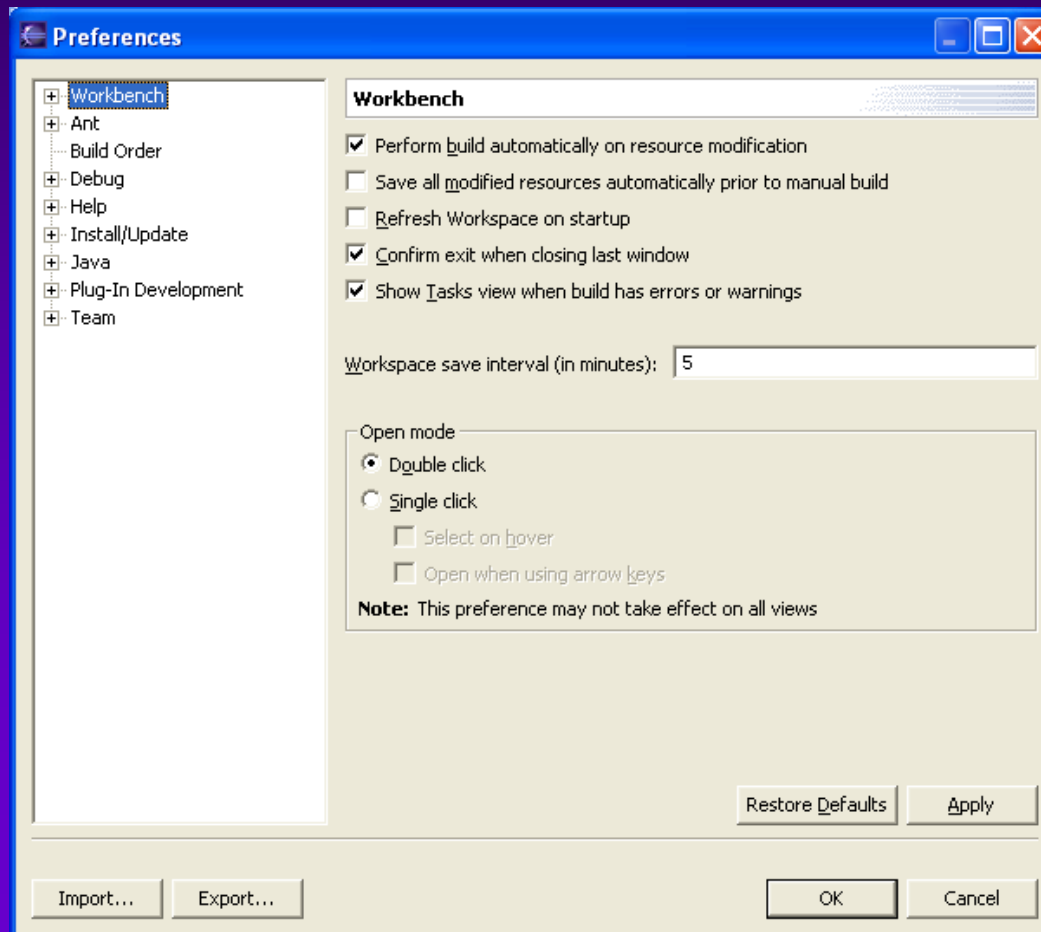






# Prepare Workbench

- Select "Workbench"
- Make sure "Perform build automatically on resource modification" is checked





# Code Organization

---

- Projects
- Packages
- Classes



# Create a new Java project

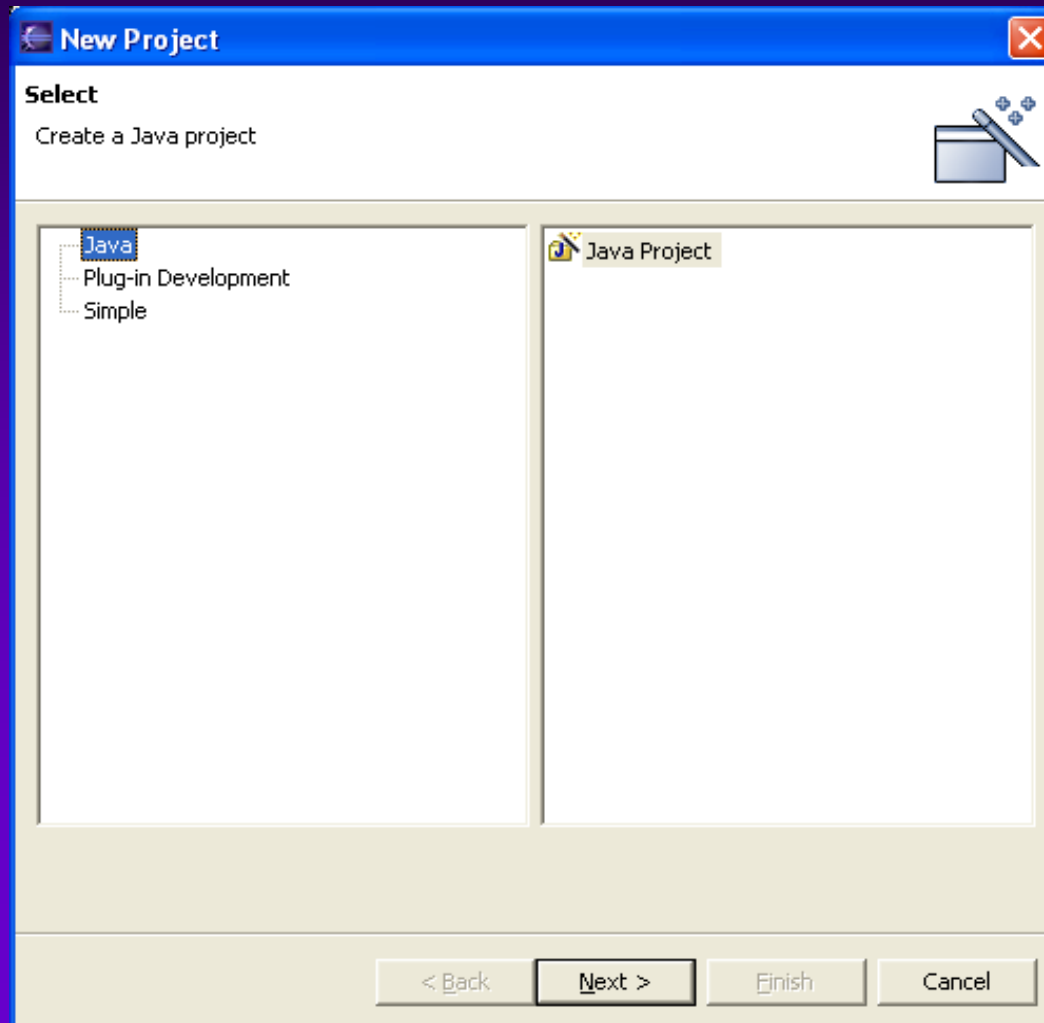
---

- Projects are the containers of packages
- All java projects are directly stored inside a project
- Generated CLASS files are stored along with the JAVA source files
- Two ways to create a new project
  1. From the File > New > Project menu
  2. From the "New Java Project" icon in the toolbar



# Create a new Java project

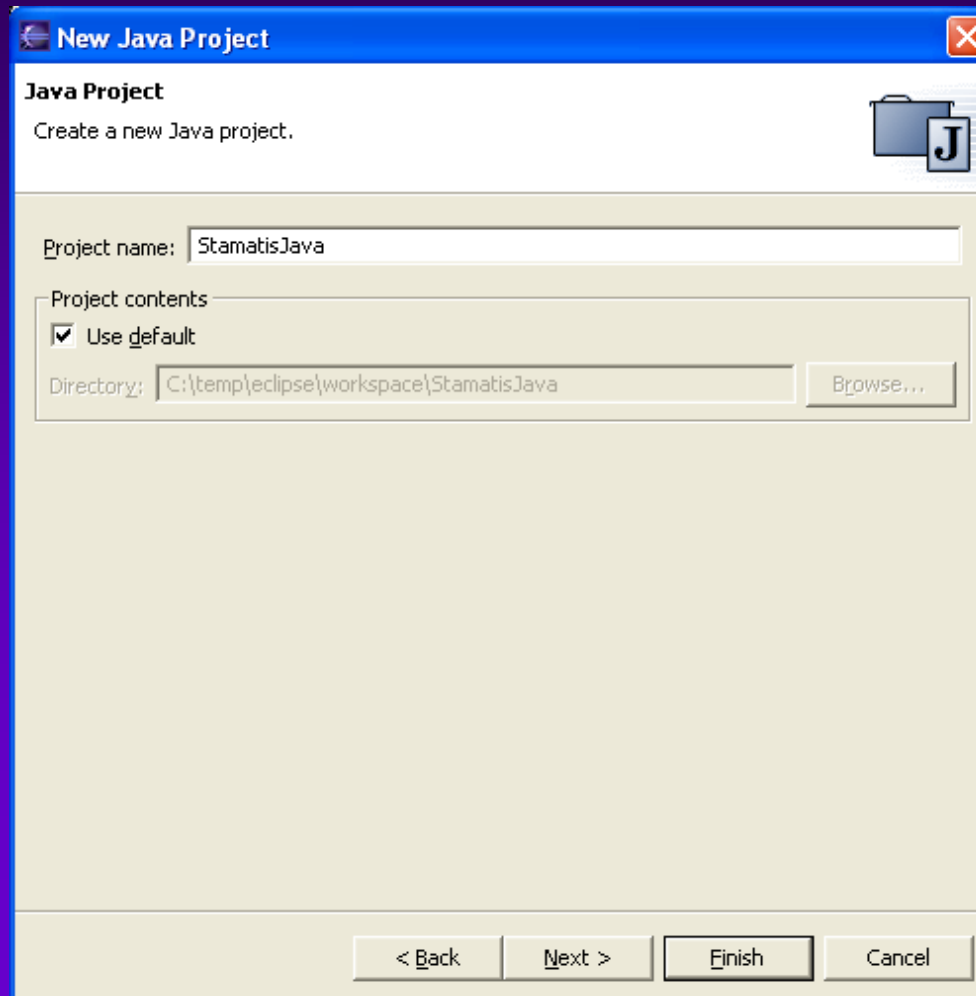
- Select "Java", then Next





# Create a new Java project

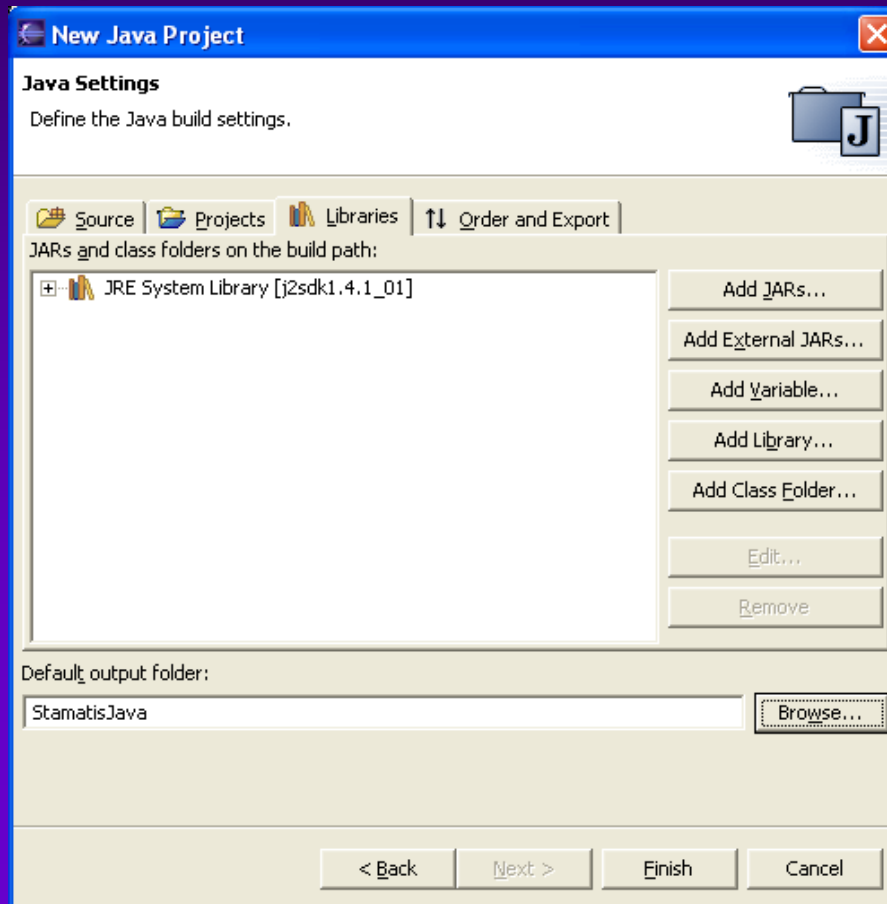
- Choose whether you will “import” content in the project. In this case the content is NOT copied in the workspace



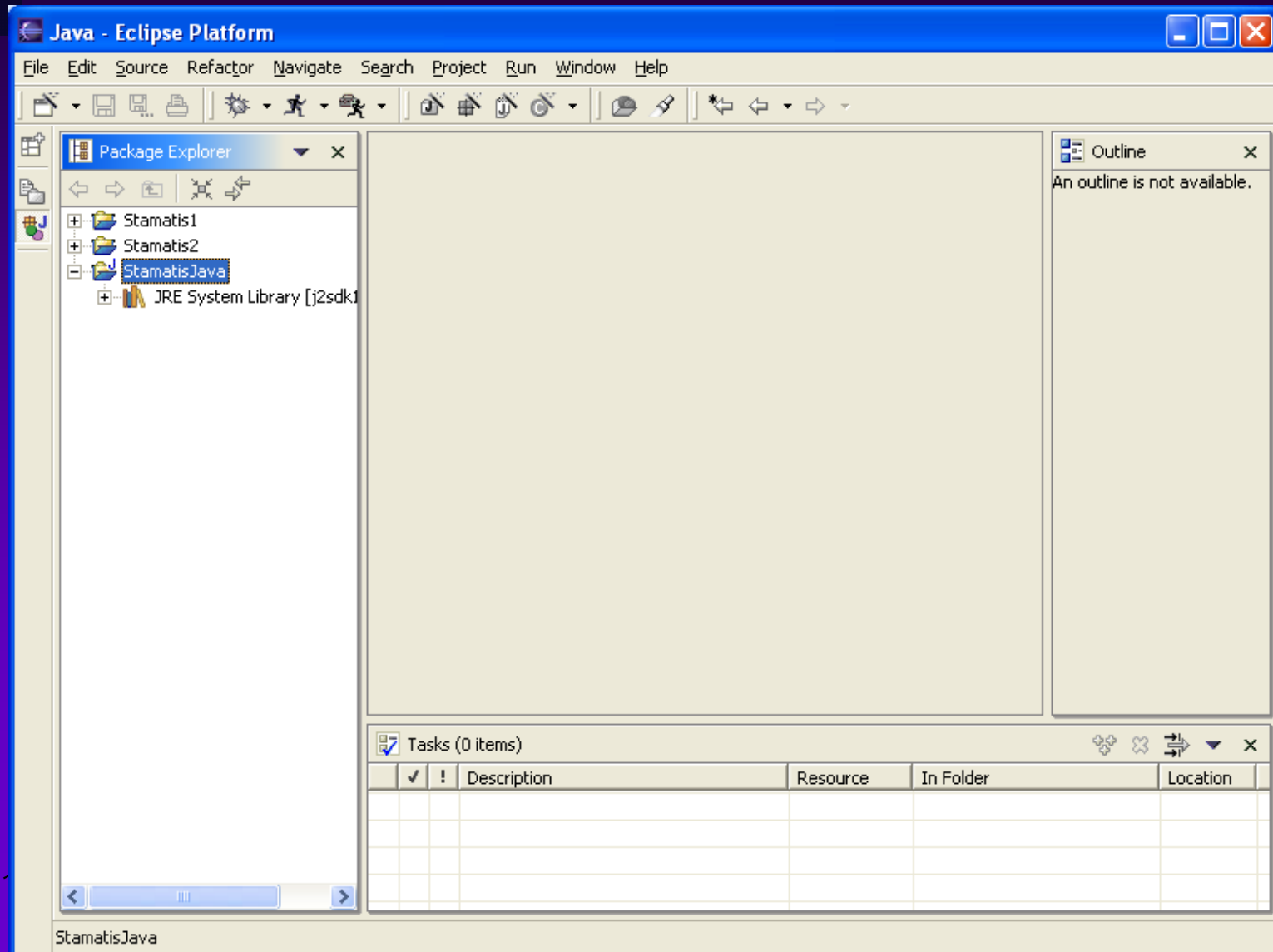


# Create a new Java project

- Add any external jars necessary to run your code (if any)
- Click "Finish"



# Create a new Java project





# Create a new Java package

---

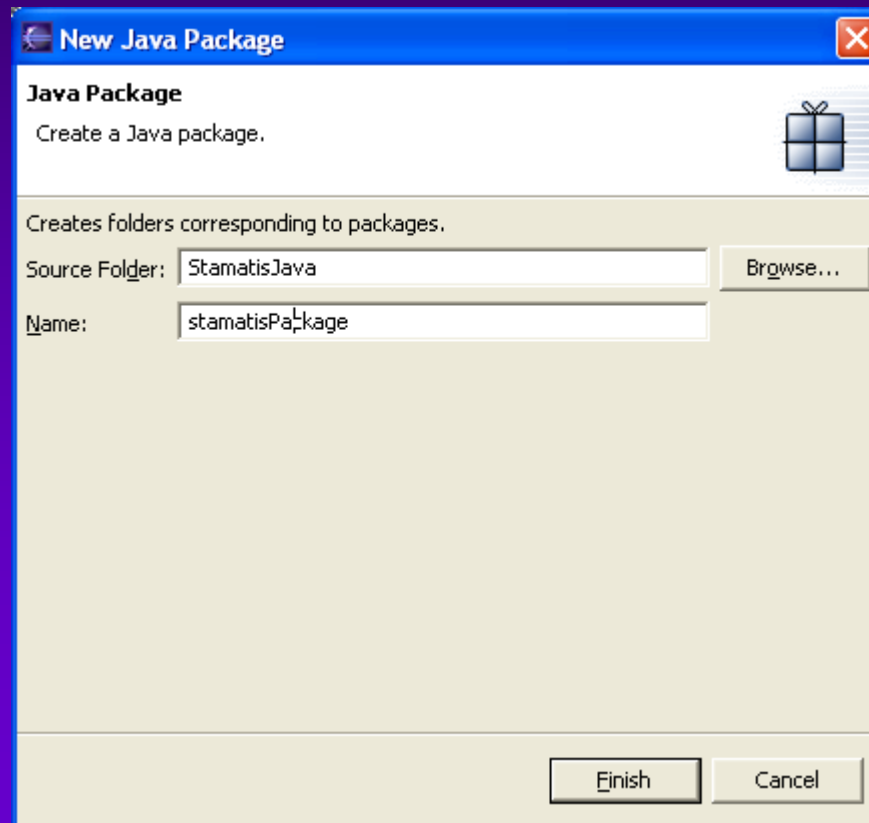
- Choose the project that will contain the package from the Package Explorer
- Three ways to create the new package
  1. From the pop-up menu (right-click on the project) select New > Package
  2. From the "File" menu select New > Package
  3. From the toolbar using the "New Package" icon



# Create a new Java package

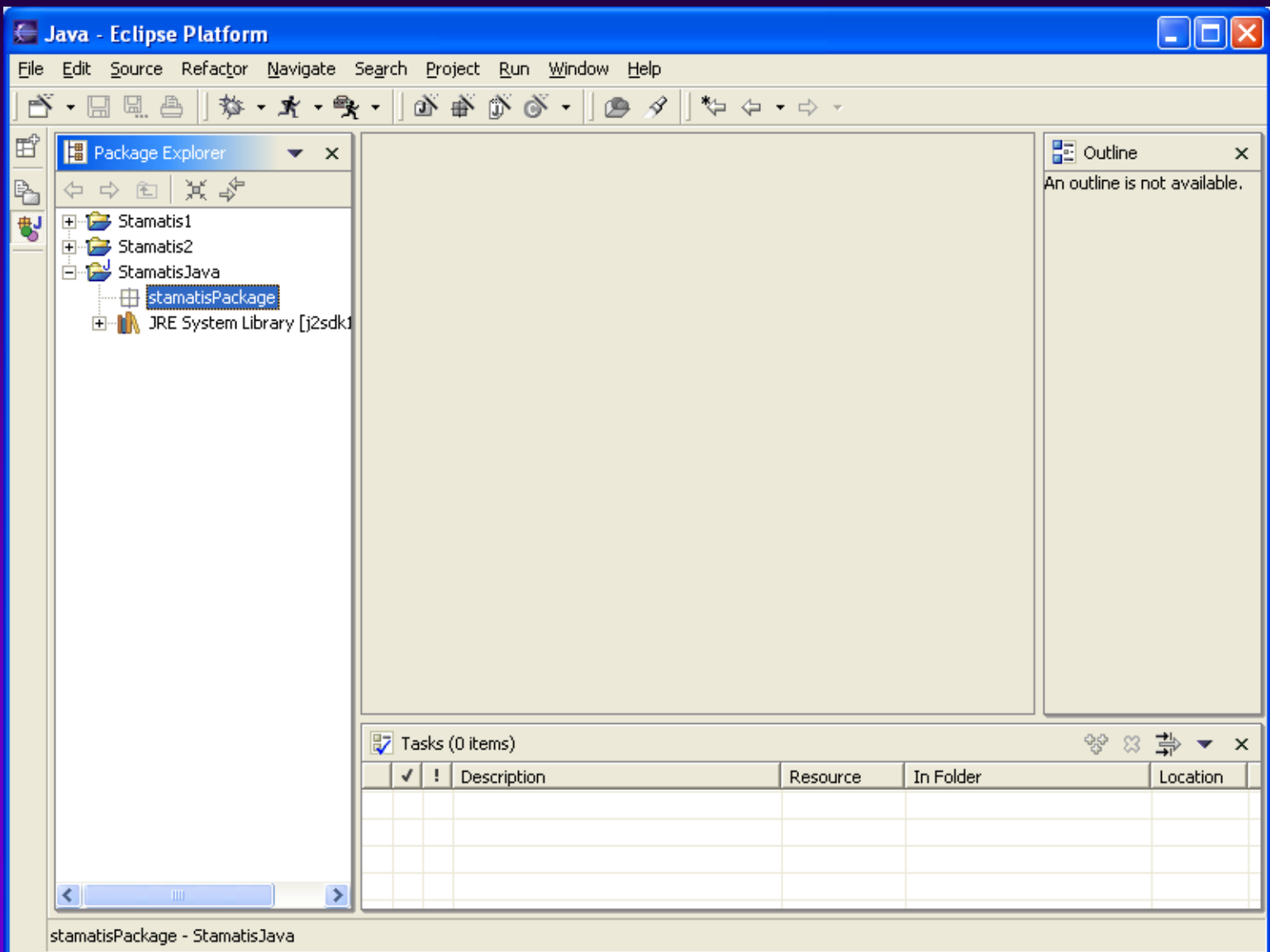
---

- Give a name to the package
- Click "Finish"





# Create a new Java package





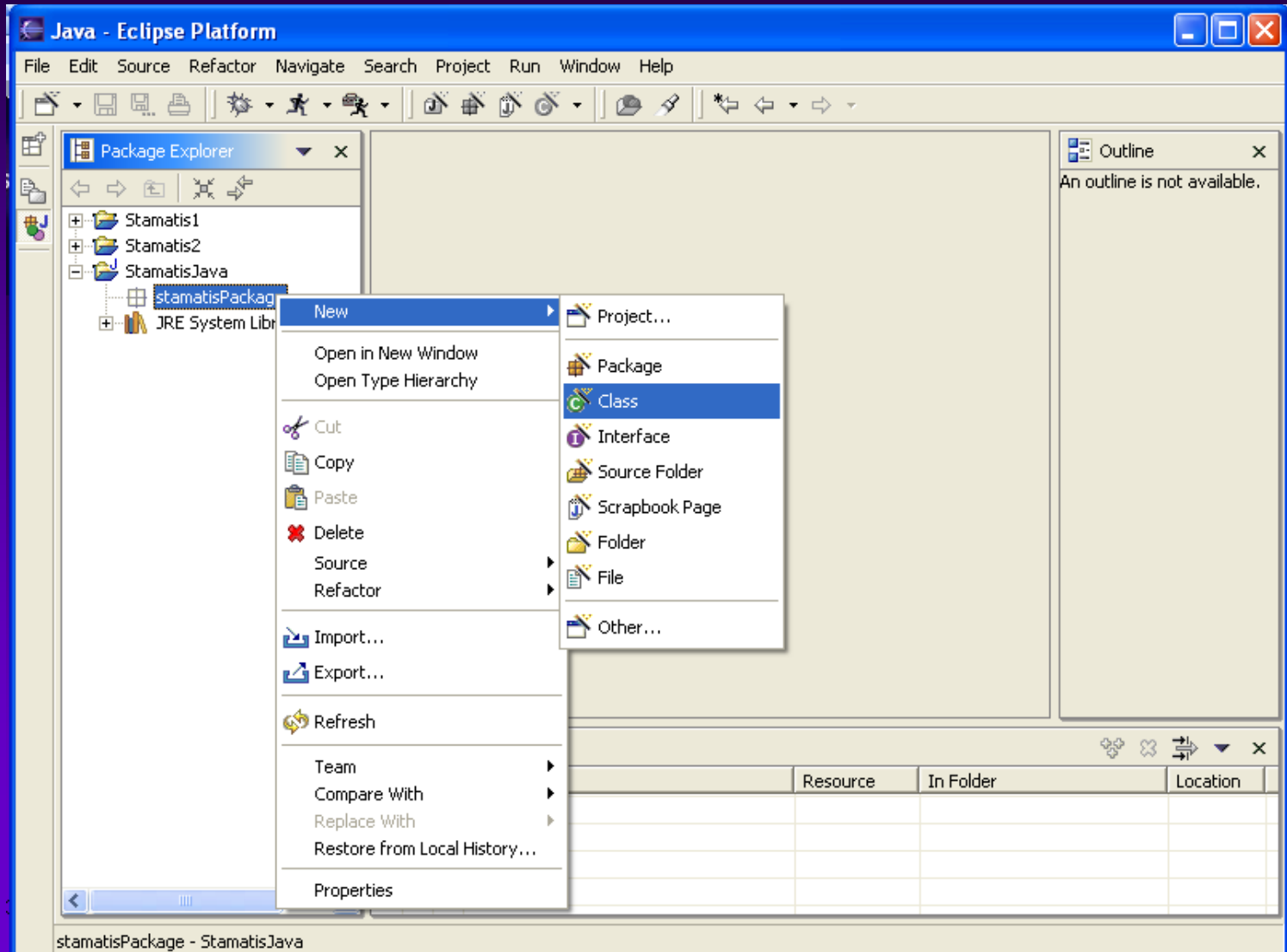
# Create a new Java Class

---

- There are various ways to create a new Java Class
  - Select the package where the new class will reside
    1. Use "New Class" icon in the workbench toolbar
    2. From the package's pop-up menu (right-click) choose New > Class
    3. From the toolbar select the "New" icon's drop-down and then select "Class"
    4. From the toolbar select the "New" icon and then "Java" in the left pane and "Class" in the right pane
    5. Select File > New > Class menu
  
- You can always type a new class in an existing Java file from the Package Explorer



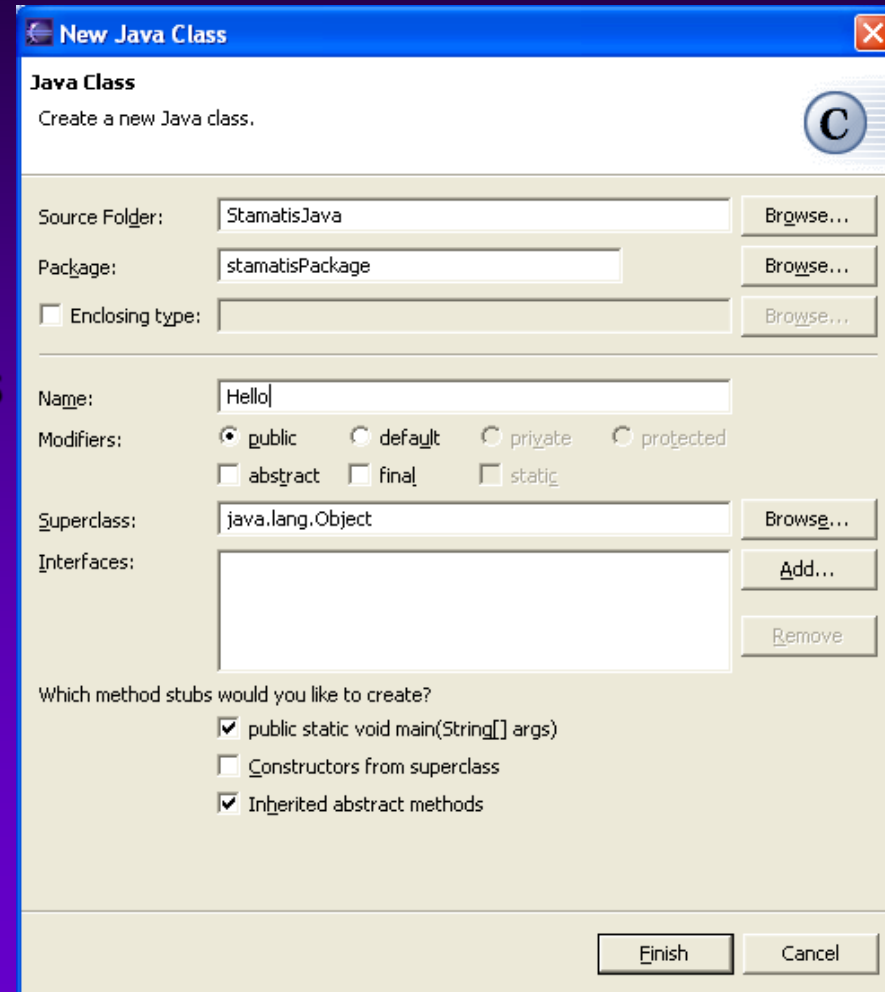
# Create a new Java Class



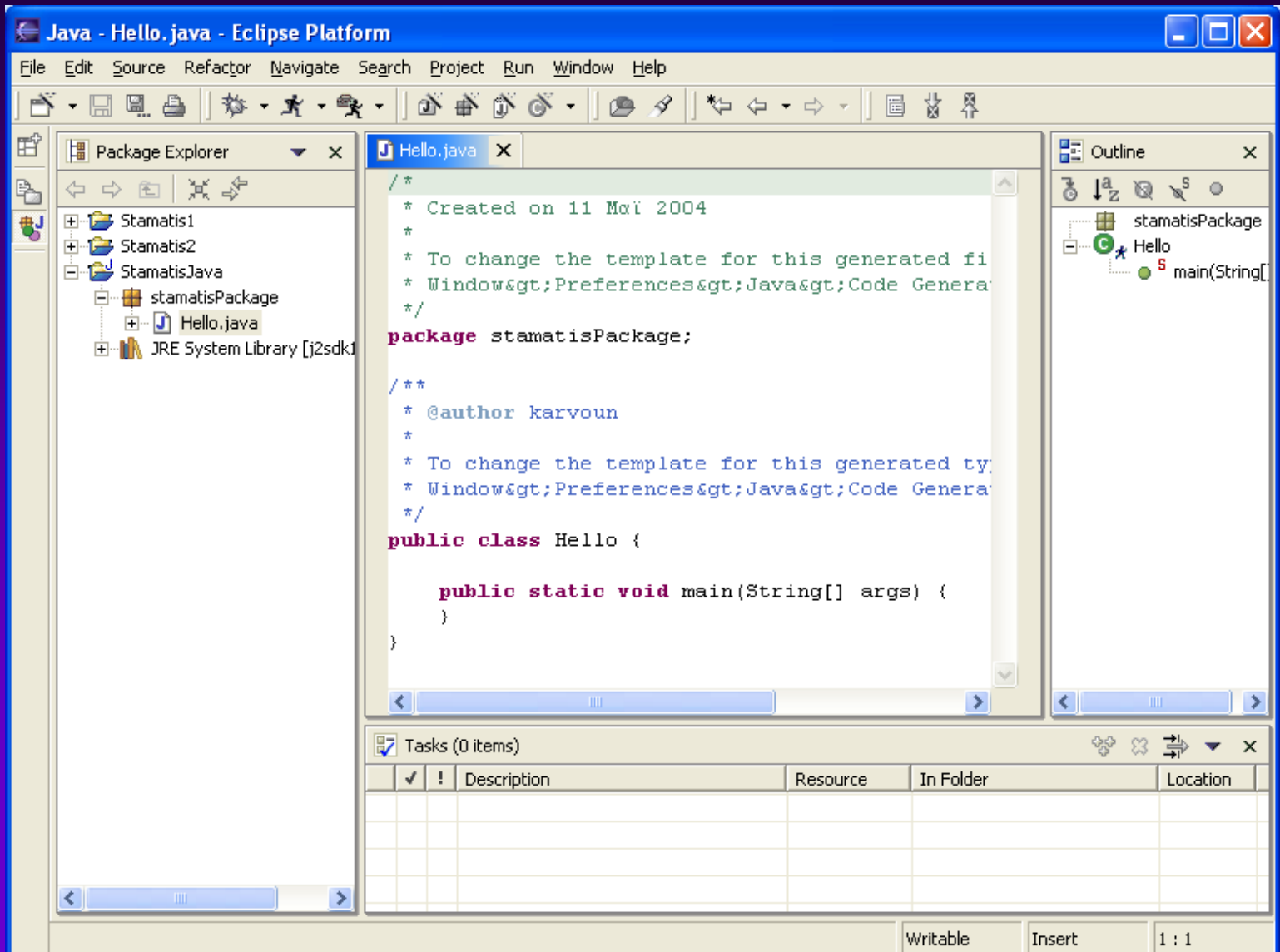


# Create a new Java Class

- Select a class name
- Select possible superclass
- Select possible implementing interfaces
- e.t.c



# Create a new Java Class



The screenshot shows the Eclipse IDE interface with the following components:

- Package Explorer:** Shows a project structure with folders 'Stamatis1', 'Stamatis2', and 'StamatisJava'. Under 'StamatisJava', there is a package 'stamatisPackage' containing a file 'Hello.java'.
- Outline:** Shows the class structure for 'stamatisPackage' with a class 'Hello' containing a method 'main(String[])'.
- Main Editor:** Displays the Java code for 'Hello.java':
 

```

/*
 * Created on 11 Mar 2004
 *
 * To change the template for this generated file
 * Window>Preferences>Java>Code Genera
 */
package stamatisPackage;

/**
 * @author karvoun
 *
 * To change the template for this generated type
 * Window>Preferences>Java>Code Genera
 */
public class Hello {

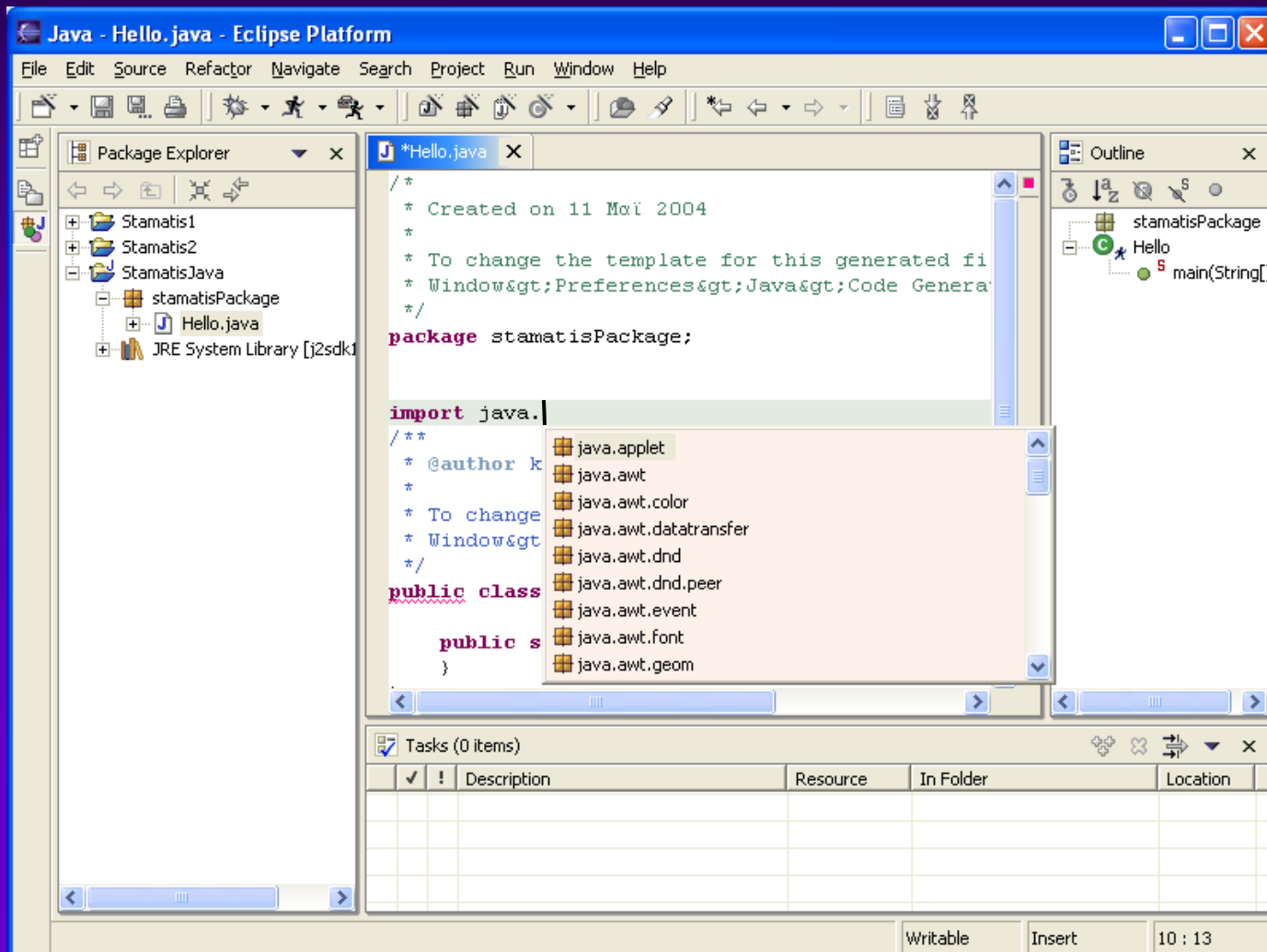
    public static void main(String[] args) {

    }

}
      
```
- Tasks:** A table at the bottom showing 0 items.

# Java editor features

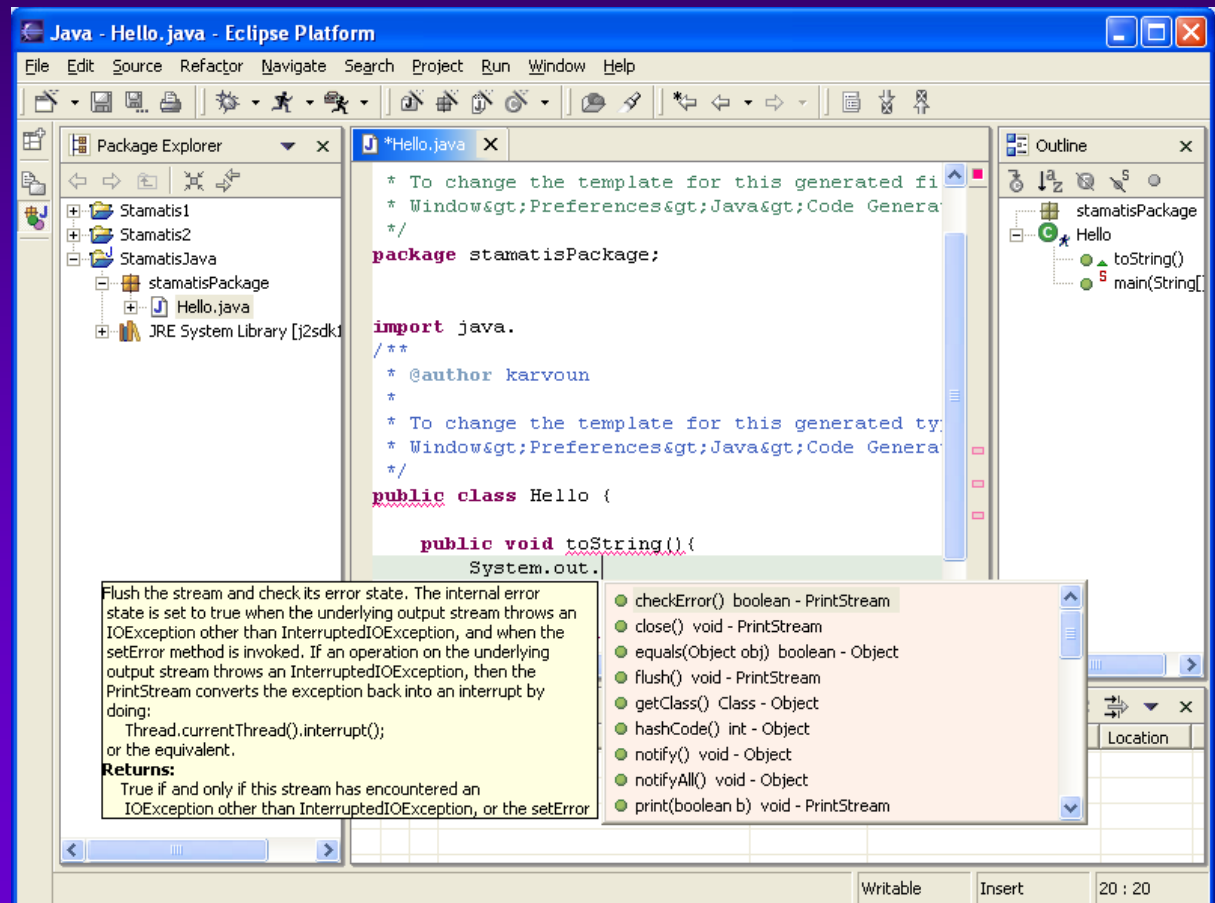
- Import assistance





# Java editor features

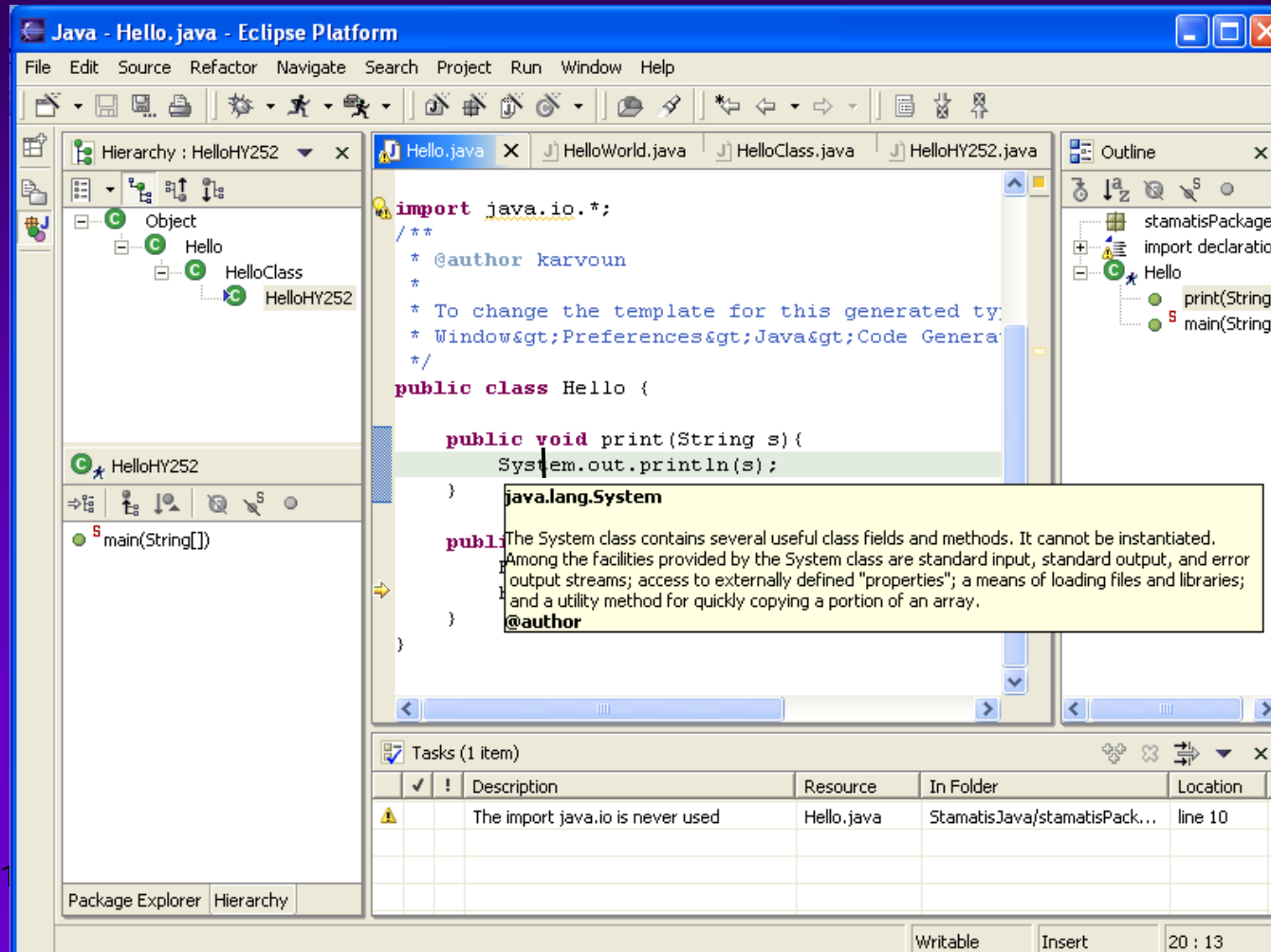
- Method Completion (automatically or using Ctrl+Space)





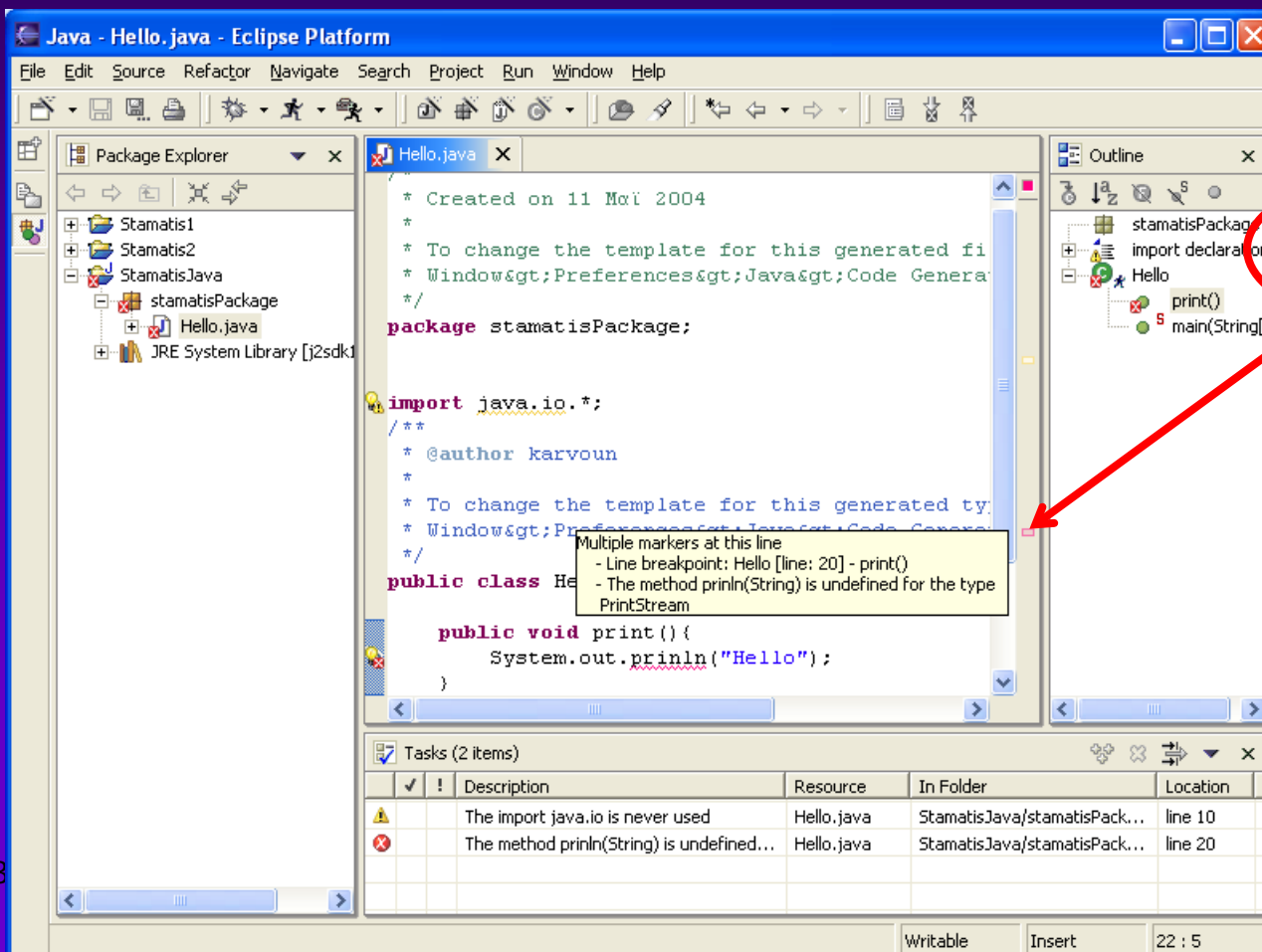
# Java editor features

- Hovering over identifier shows Javadoc spec



# Java editor features

- On-the-fly spell check

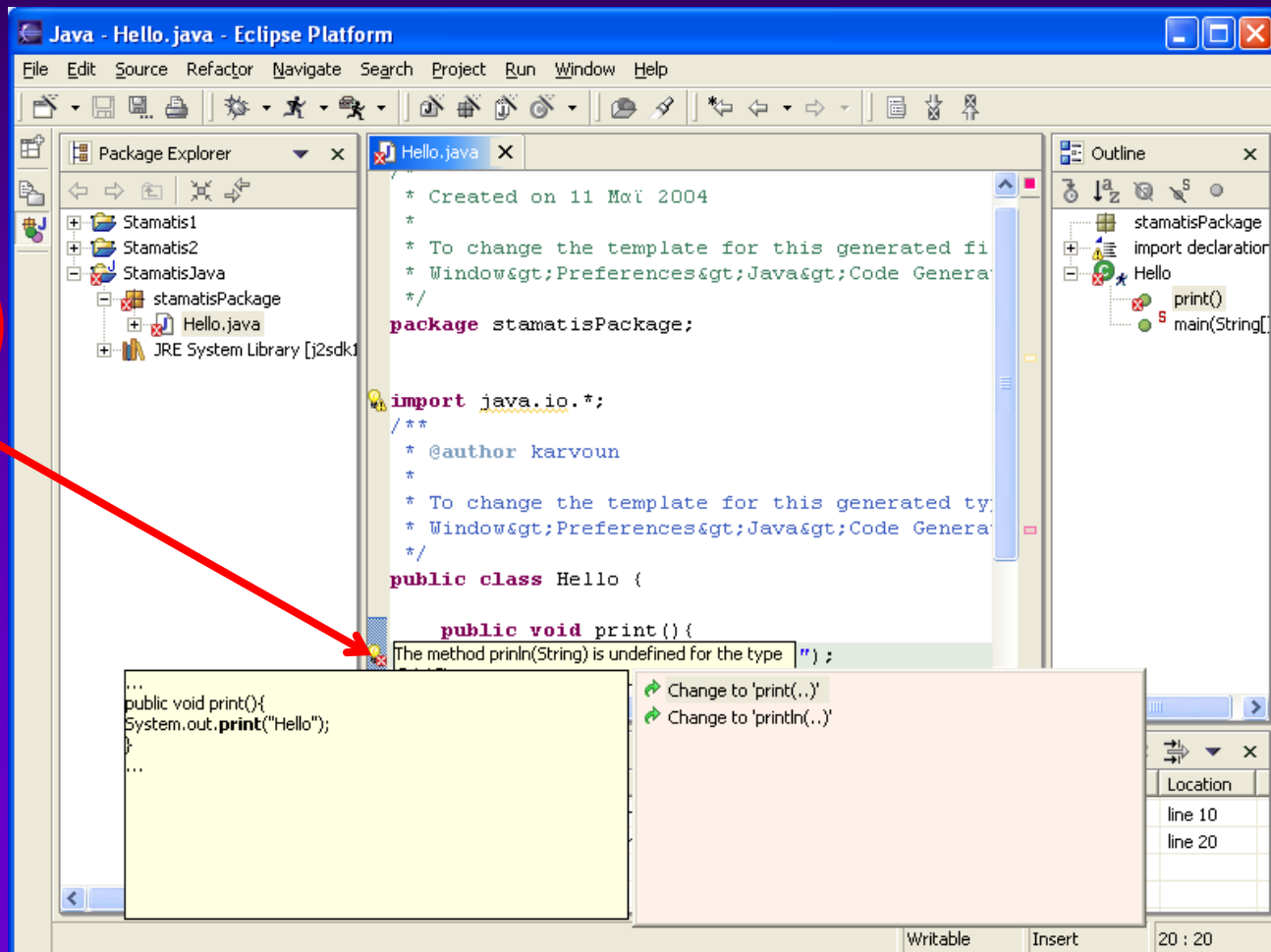


Hover over this to see mistake

# Java editor features

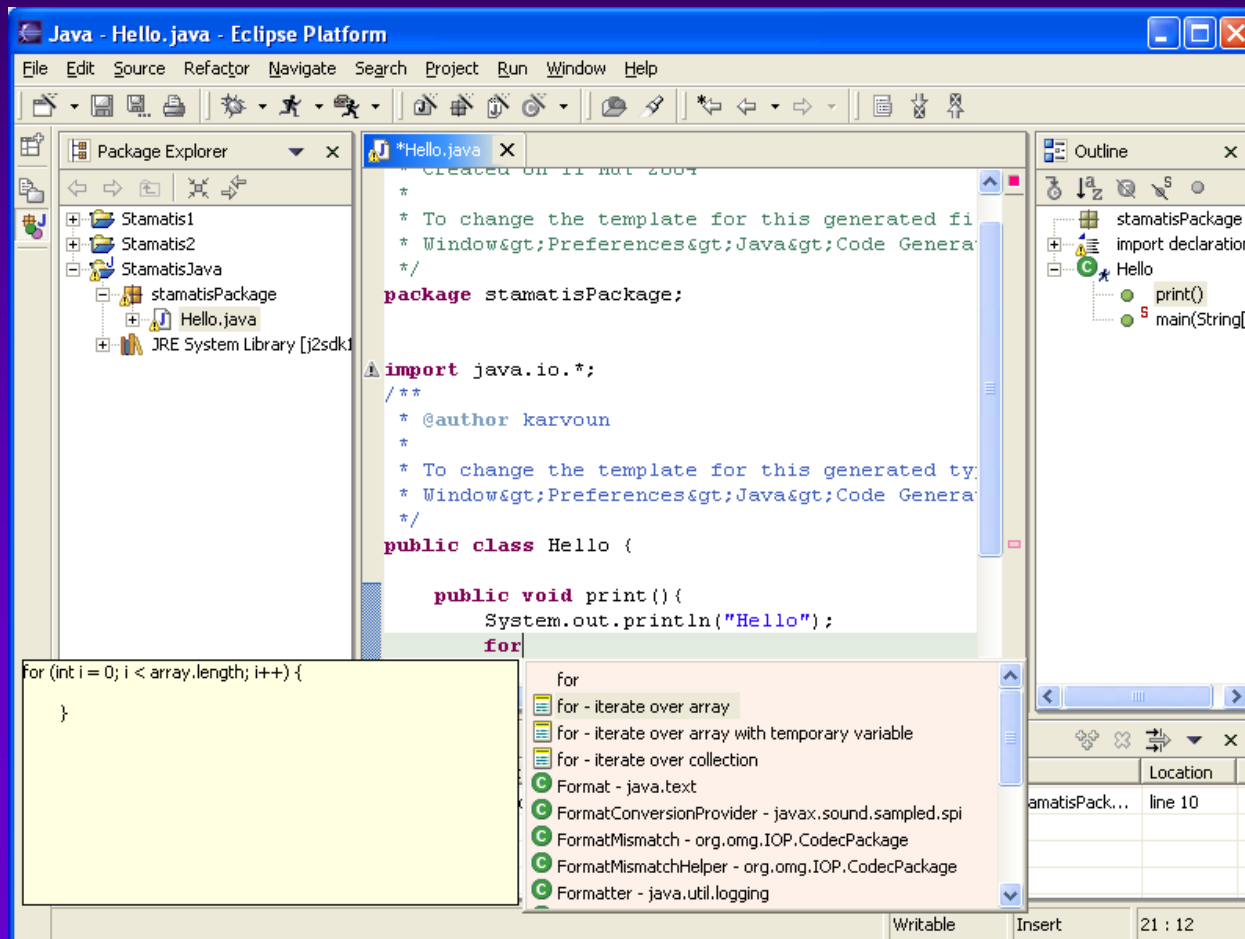
- Quick Fix

Double click here to get suggestions



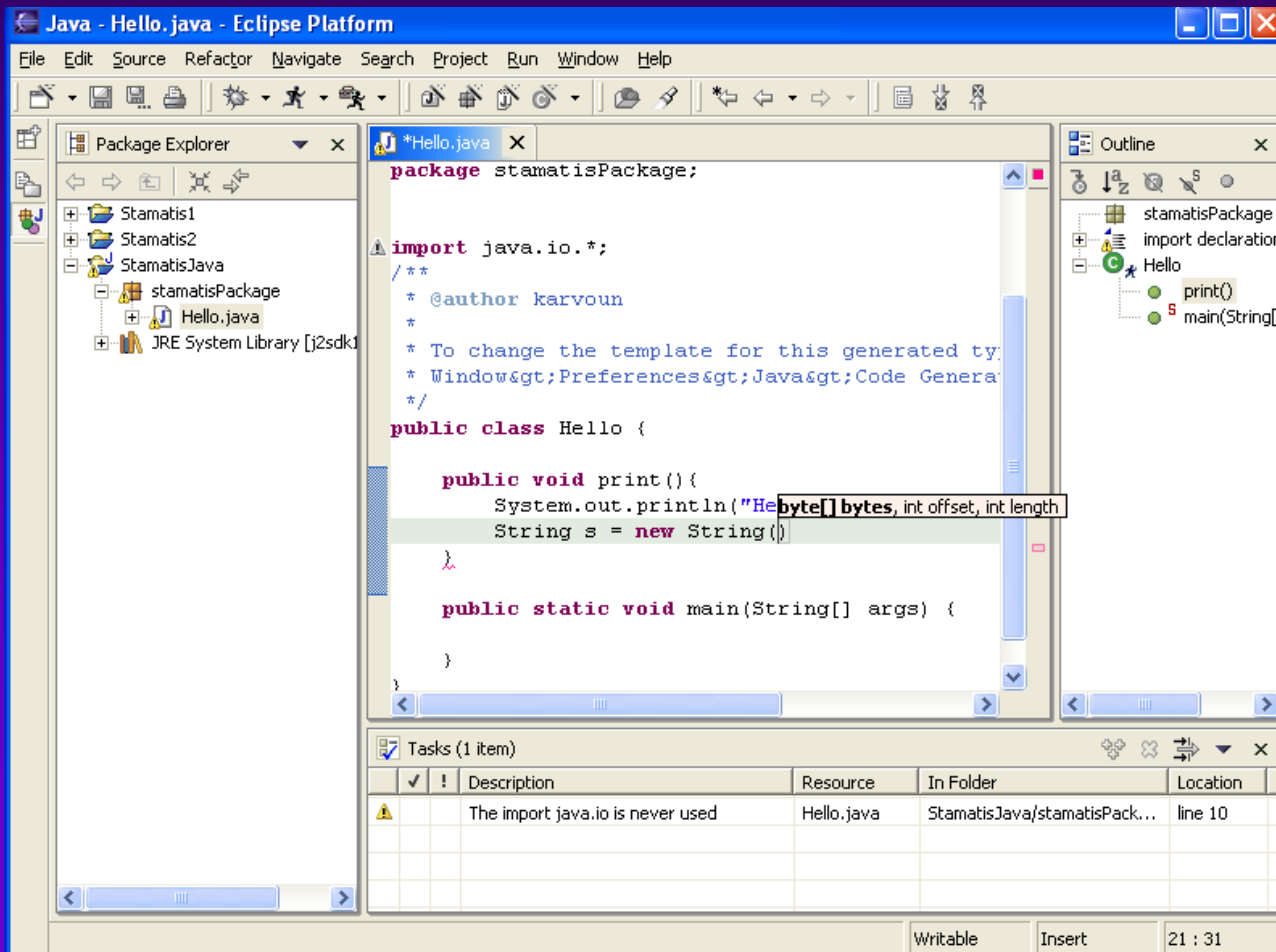
# Java editor features

- Code templates (Ctrl+Space)



# Java editor features

- Argument hints



The screenshot shows the Eclipse IDE with a Java editor window titled "Java - Hello.java - Eclipse Platform". The editor displays the following code:

```

package stamatisPackage;

import java.io.*;
/**
 * @author karvoun
 *
 * To change the template for this generated type
 * Window>Preferences>Java>Code Genera
 */
public class Hello {

    public void print() {
        System.out.println("Hello, world!");
        String s = new String();
    }

    public static void main(String[] args) {

    }
}
    
```

An argument hint tooltip is visible over the `String s = new String();` line, displaying: `byte[] bytes, int offset, int length`.

The Package Explorer on the left shows a project structure with folders "Stamatis1", "Stamatis2", and "StamatisJava", and files "Hello.java" and "JRE System Library [j2sdk1...". The Outline view on the right shows the class hierarchy: "stamatisPackage" containing "import declaration" and "Hello", with "Hello" containing "print()" and "main(String[])".

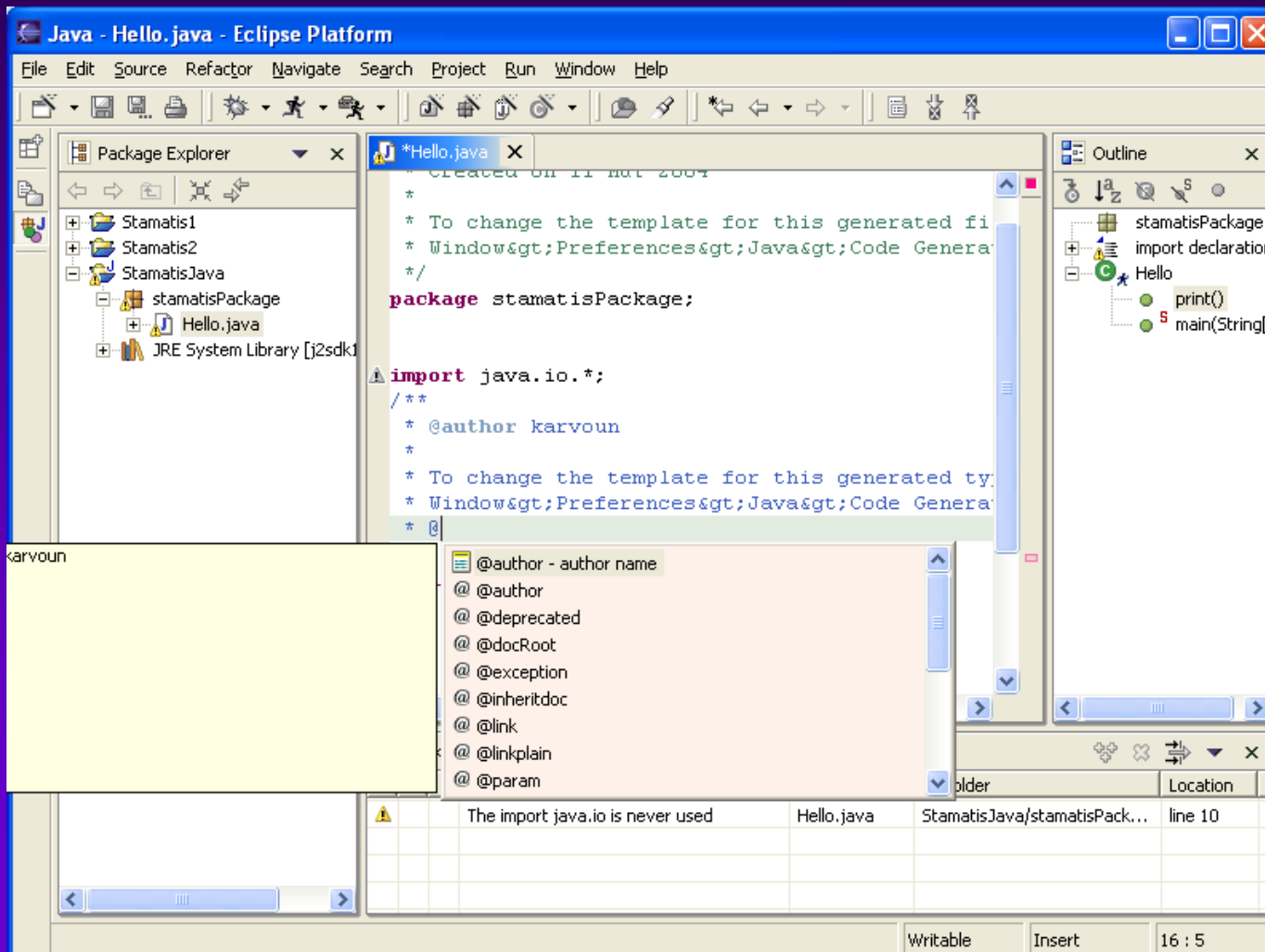
The Tasks view at the bottom shows one item:

✓	!	Description	Resource	In Folder	Location
	!	The import java.io is never used	Hello.java	StamatisJava/stamatisPack...	line 10

The status bar at the bottom indicates "Writable", "Insert", and "21 : 31".

# Java editor features

## ■ JavaDoc Assist



The screenshot shows the Eclipse IDE with the following components:

- Package Explorer:** Shows a project named 'StamatisJava' with a package 'stamatisPackage' containing a file 'Hello.java'.
- Main Editor:** Displays the content of 'Hello.java'. The code includes a package declaration, an import statement for 'java.io.\*', and a doc comment for the '@author' tag. The doc comment text is:
 

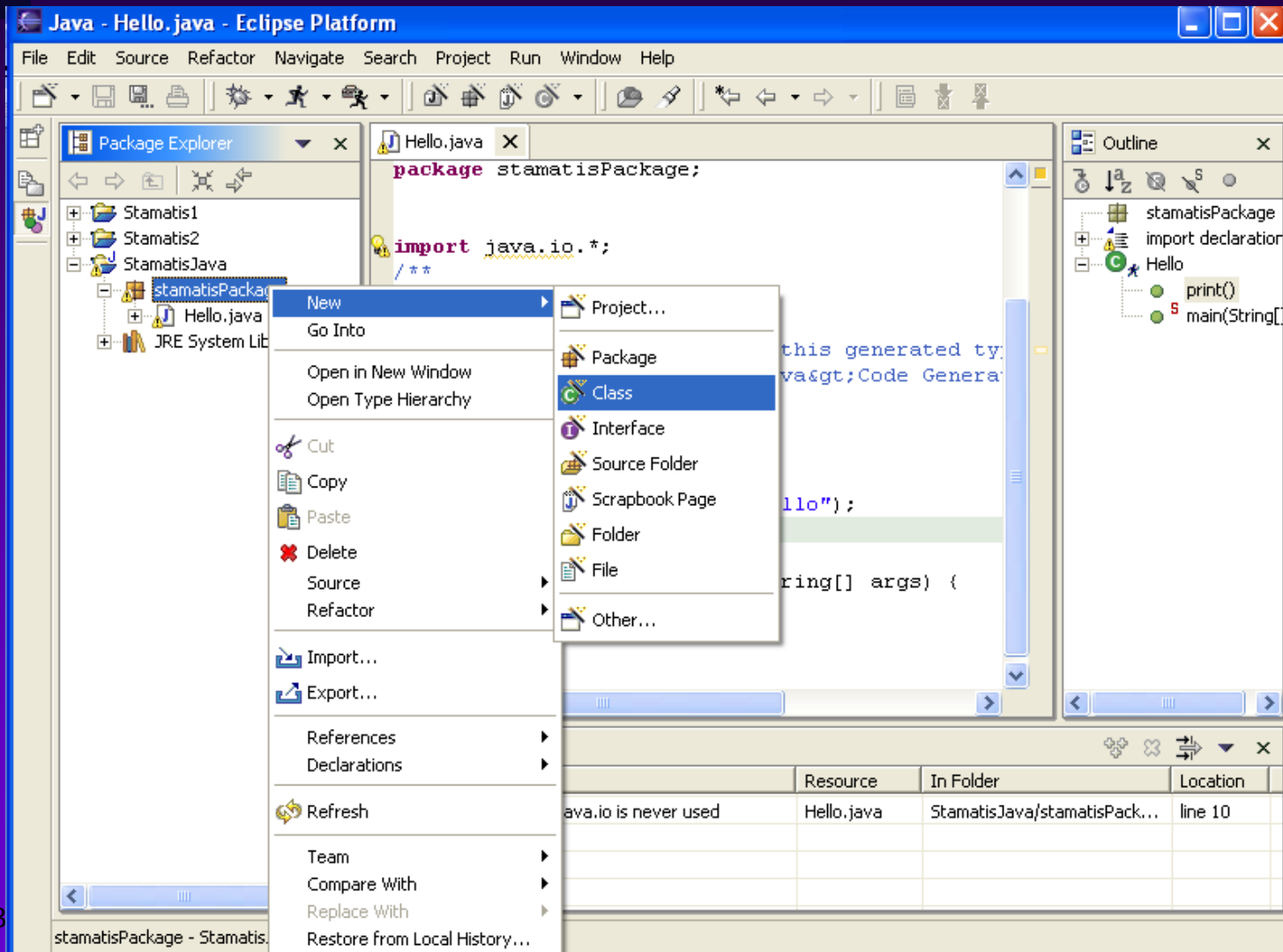
```

      /**
       * @author karvoun
       *
       * To change the template for this generated file
       * Window>Preferences>Java>Code Genera
       */
      package stamatisPackage;

      import java.io.*;

      /**
       * @author karvoun
       *
       * To change the template for this generated by
       * Window>Preferences>Java>Code Genera
       */
      
```
- JavaDoc Assist:** A tooltip is displayed over the '@author' tag, listing various JavaDoc tags:
  - @author - author name
  - @author
  - @deprecated
  - @docRoot
  - @exception
  - @inheritdoc
  - @link
  - @linkplain
  - @param
- Outline:** Shows the class hierarchy: 'stamatisPackage' containing 'Hello', which has methods 'print()' and 'main(String[])'.
- Problems View:** At the bottom, a warning is shown: 'The import java.io is never used' in 'Hello.java' at 'StamatisJava/stamatisPack...' on 'line 10'.

# Class Hierarchies



The screenshot shows the Eclipse IDE interface. The Package Explorer on the left displays a project structure with a package named 'stamatisPackage' containing a file 'Hello.java'. The main editor window shows the source code of 'Hello.java' with a context menu open over it. The 'Class' option is selected in the menu. The Outline view on the right shows the class hierarchy for 'stamatisPackage', including 'import declaration', 'Hello', 'print()', and 'main(String[])'. A warning message at the bottom of the IDE states 'java.io is never used'.

	Resource	In Folder	Location
java.io is never used	Hello.java	StamatisJava/stamatisPack...	line 10

# Class Hierarchies

The image shows two overlapping dialog boxes in the Eclipse IDE. The 'New Java Class' dialog is in the foreground, and the 'Superclass Selection' dialog is behind it.

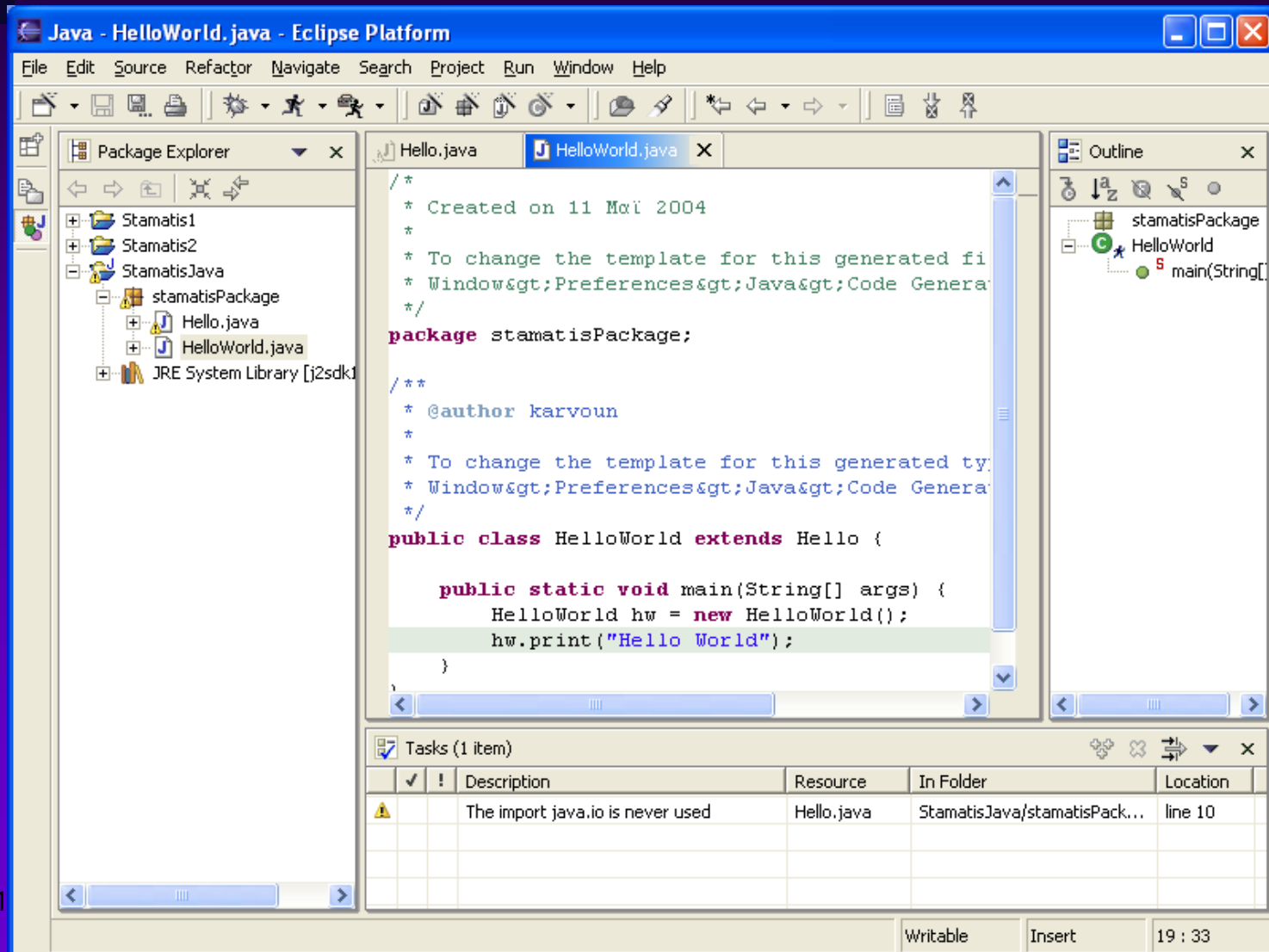
**New Java Class Dialog:**

- Java Class:** Create a new Java class. (Icon: C)
- Source Folder:** StamatisJava (Browse...)
- Package:** stamatisPackage (Browse...)
- Enclosing type:** (Browse...)
- Name:** HelloWorld
- Modifiers:**  public,  default,  private,  protected;  abstract,  final,  static
- Superclass:** stamatisPackage.Hello (Browse...)
- Interfaces:** (Add... Remove)
- Which method stubs would you like to create?**
  - public static void main(String[] args)
  - Constructors from superclass
  - Inherited abstract methods
- Buttons:** Finish, Cancel

**Superclass Selection Dialog:**

- Choose a type:** Hel
- Matching types:**
  - Hello
  - Help
- Qualifier:**
  - stamatisPackage - StamatisJava
- Buttons:** OK, Cancel

# Class Hierarchies



The screenshot shows the Eclipse IDE interface with the following components:

- Package Explorer:** Shows a project structure with packages `Stamatis1`, `Stamatis2`, and `StamatisJava`. Under `StamatisJava`, there is a package `stamatisPackage` containing `Hello.java` and `HelloWorld.java`. A `JRE System Library [j2sdk1` is also visible.
- Code Editor:** Displays the source code for `HelloWorld.java`. The code includes a package declaration, a class declaration extending `Hello`, and a `main` method that prints "Hello World".
- Outline:** Shows a class hierarchy starting with `stamatisPackage` containing `HelloWorld`, which has a `main(String[] args)` method.
- Tasks:** A table at the bottom shows a task: "The import java.io is never used" in `Hello.java` at line 10.

```

/*
 * Created on 11 May 2004
 *
 * To change the template for this generated file
 * Window>Preferences>Java>Code Genera
 */
package stamatisPackage;

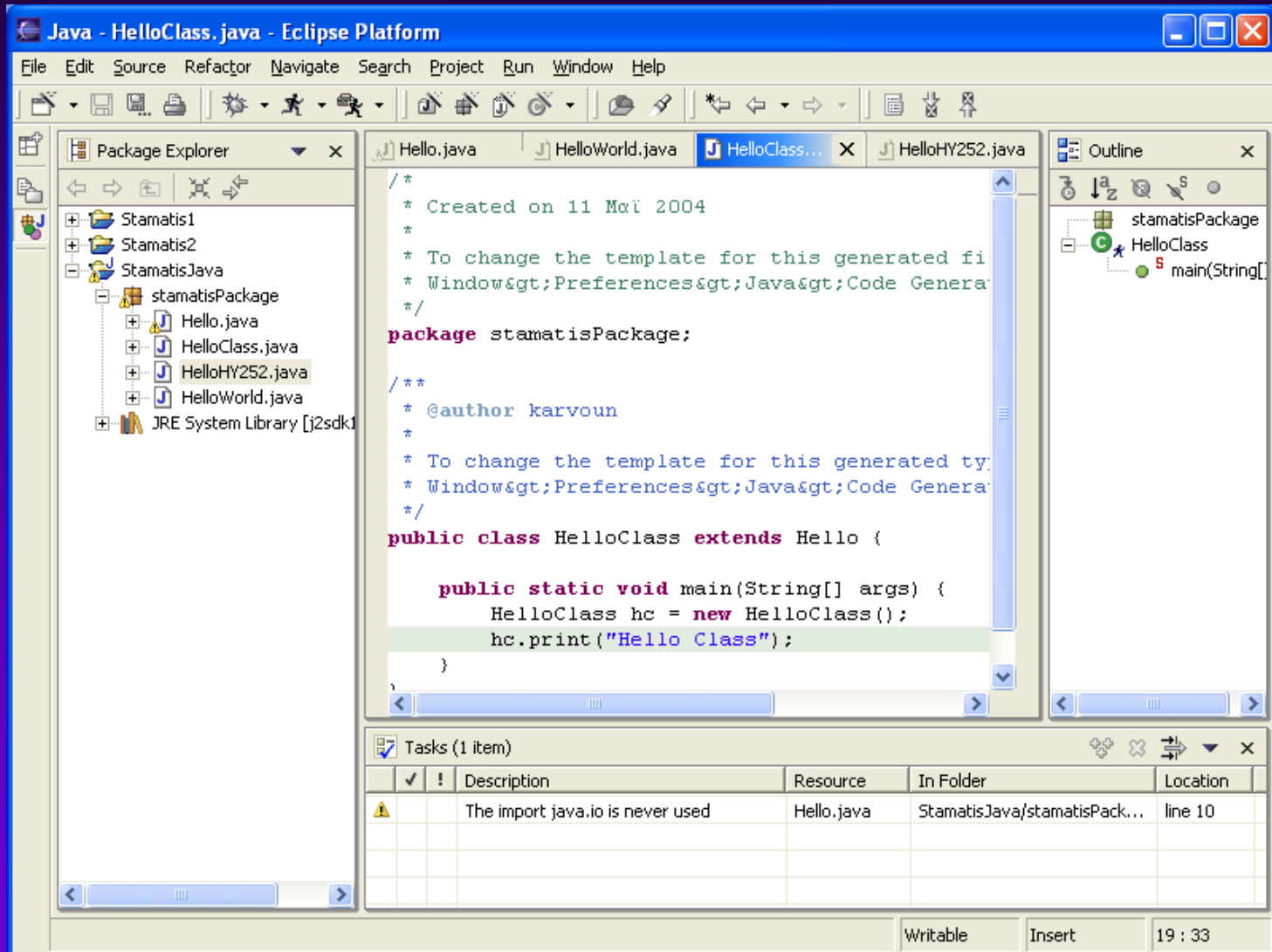
/**
 * @author karvoun
 *
 * To change the template for this generated ty
 * Window>Preferences>Java>Code Genera
 */
public class HelloWorld extends Hello {

    public static void main(String[] args) {
        HelloWorld hw = new HelloWorld();
        hw.print("Hello World");
    }
}

```

✓	!	Description	Resource	In Folder	Location
	⚠	The import java.io is never used	Hello.java	StamatisJava/stamatisPack...	line 10

# Class Hierarchies



The screenshot shows the Eclipse IDE interface with the following components:

- Package Explorer:** Shows a project structure with packages `Stamatis1`, `Stamatis2`, and `StamatisJava`. Under `StamatisJava`, there is a package `stamatisPackage` containing files `Hello.java`, `HelloClass.java`, `HelloHY252.java`, and `HelloWorld.java`. The `JRE System Library [j2sdk1` is also visible.
- Code Editor:** Displays the source code for `HelloClass.java`. The code includes a package declaration, a class declaration extending `Hello`, and a `main` method.
 

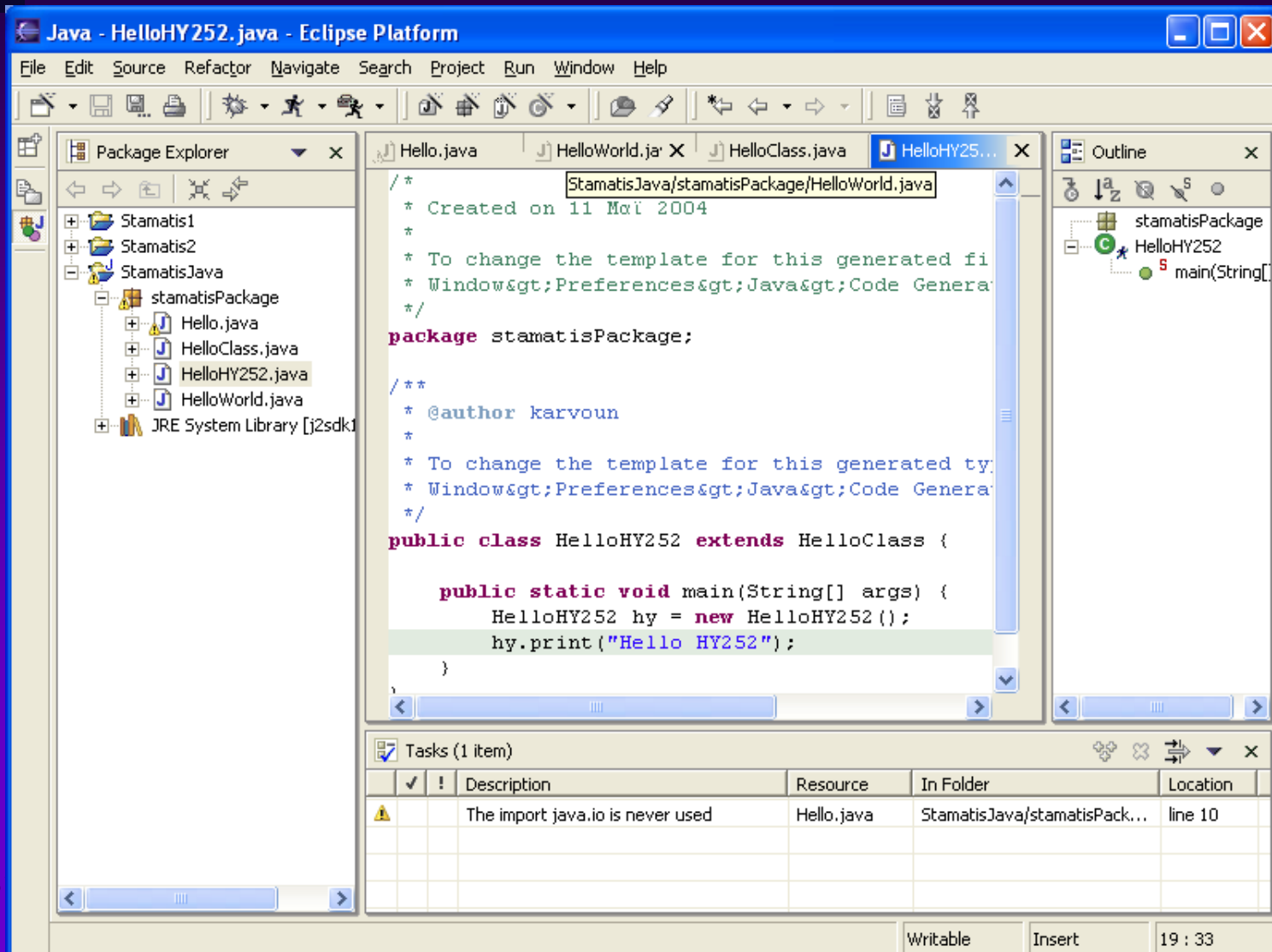
```

      /*
       * Created on 11 Mai 2004
       *
       * To change the template for this generated file
       * Window>Preferences>Java>Code Genera
       */
      package stamatisPackage;

      /**
       * @author karvoun
       *
       * To change the template for this generated type
       * Window>Preferences>Java>Code Genera
       */
      public class HelloClass extends Hello {

          public static void main(String[] args) {
              HelloClass hc = new HelloClass();
              hc.print("Hello Class");
          }
      }
      
```
- Outline:** Shows a class hierarchy starting with `stamatisPackage` containing `HelloClass`, which has a `main(String[] args)` method.
- Tasks:** A table at the bottom shows a task: "The import java.io is never used" in `Hello.java` at line 10.

# Class Hierarchies



Java - HelloHY252.java - Eclipse Platform

File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer

- Stamatis1
- Stamatis2
- StamatisJava
  - stamatisPackage
    - Hello.java
    - HelloClass.java
    - HelloHY252.java
    - HelloWorld.java
  - JRE System Library [j2sdk1

```

/*
 * Created on 11 May 2004
 *
 * To change the template for this generated file
 * Window>Preferences>Java>Code Genera
 */
package stamatisPackage;

/**
 * @author karvoun
 *
 * To change the template for this generated ty
 * Window>Preferences>Java>Code Genera
 */
public class HelloHY252 extends HelloClass {

    public static void main(String[] args) {
        HelloHY252 hy = new HelloHY252 ();
        hy.print("Hello HY252");
    }
}

```

Outline

- stamatisPackage
  - HelloHY252
    - main(String[]

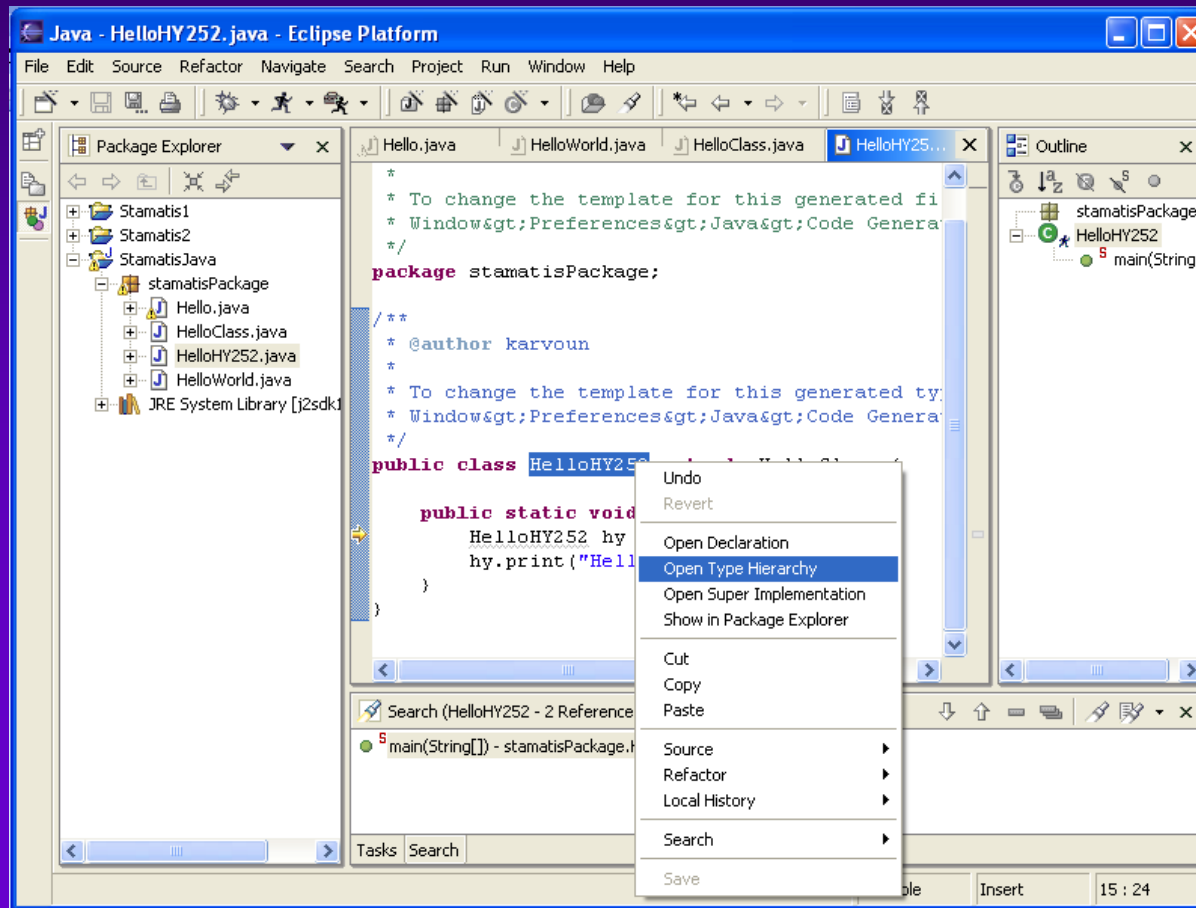
Tasks (1 item)

	✓	!	Description	Resource	In Folder	Location
	✓	!	The import java.io is never used	Hello.java	StamatisJava/stamatisPack...	line 10

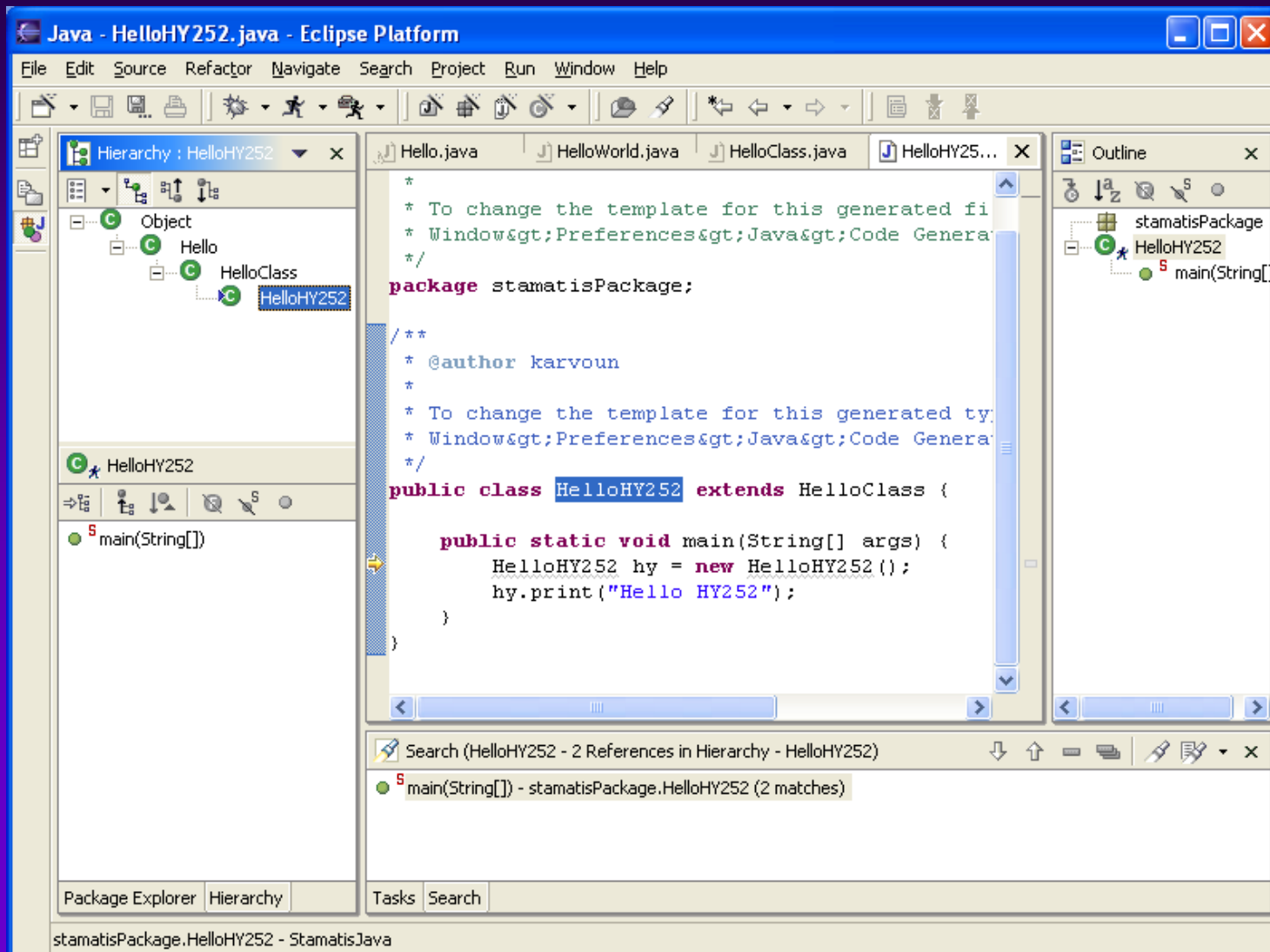
Writable Insert 19:33

# Class Hierarchies

- Select a type and from its pop-up select "Open Type Hierarchy"
- The Hierarchy View will appear



# Class Hierarchies



The screenshot shows the Eclipse IDE interface with the following components:

- Package Explorer (Left):** Shows a class hierarchy starting from `Object`, with `Hello` extending `Object`, `HelloClass` extending `Hello`, and `HelloHY252` extending `HelloClass`.
- Source Editor (Center):** Displays the source code for `HelloHY252.java`. The code includes package declarations, comments, and a class definition:
 

```

package stamatisPackage;

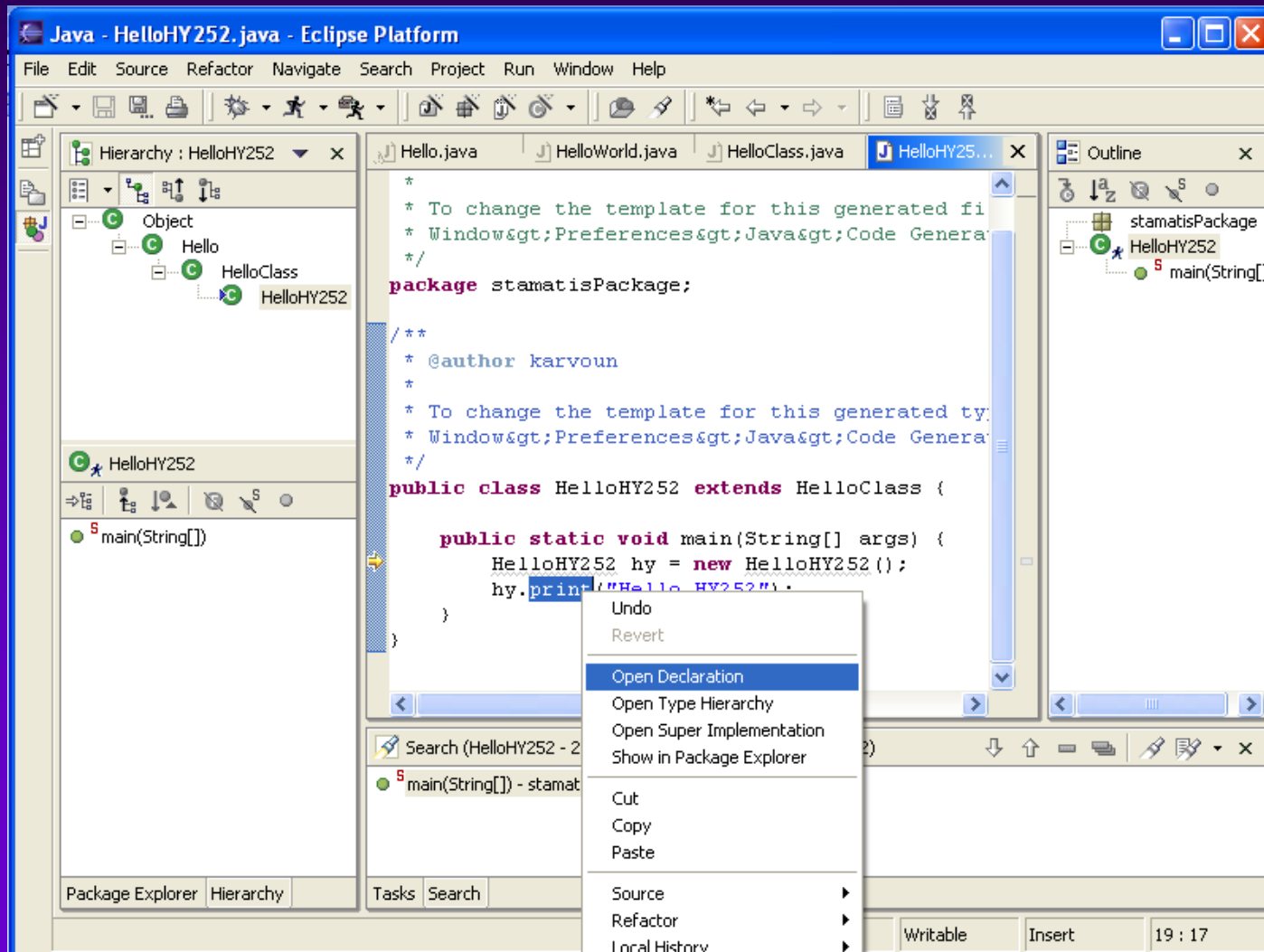
/**
 * @author karvoun
 *
 * To change the template for this generated type, go to the
 * Window>Preferences>Java>Code Generation>Comments page.
 */
public class HelloHY252 extends HelloClass {

    public static void main(String[] args) {
        HelloHY252 hy = new HelloHY252();
        hy.print("Hello HY252");
    }
}

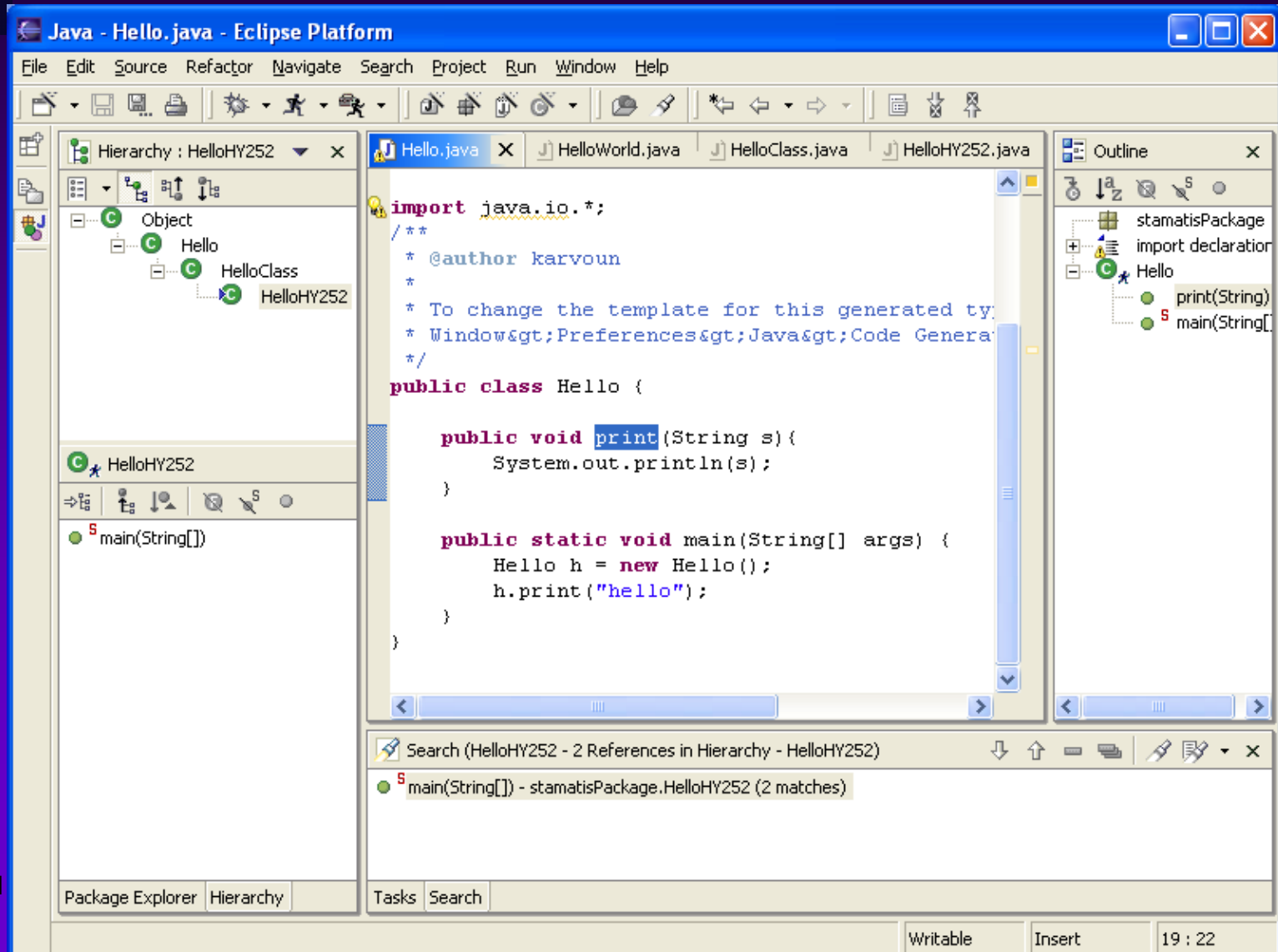
```
- Outline (Right):** Shows the project structure with `stamatisPackage` containing `HelloHY252` and its `main(String[] args)` method.
- Search (Bottom):** A search window showing 2 references in the hierarchy for `HelloHY252`, specifically pointing to the `main` method in `stamatisPackage.HelloHY252`.

# Search for Java Elements

- Declarations or References



# Search for Java Elements



The screenshot shows the Eclipse IDE interface with the following components:

- Package Explorer:** Shows the project hierarchy with 'HelloHY252' selected.
- Hierarchy:** Shows the class hierarchy: Object -> Hello -> HelloClass -> HelloHY252. The 'main(String[])' method is highlighted under 'HelloHY252'.
- Editor:** Displays the source code of 'Hello.java'. The 'print' method is highlighted in blue.
- Outline:** Shows the class structure with 'main(String[])' listed under the 'Hello' class.
- Search Window:** Shows the search results for 'main(String[])' in the 'HelloHY252' hierarchy, with 2 matches found.

```

import java.io.*;
/**
 * @author karvoun
 *
 * To change the template for this generated type
 * Window>Preferences>Java>Code Genera
 */
public class Hello {

    public void print(String s) {
        System.out.println(s);
    }

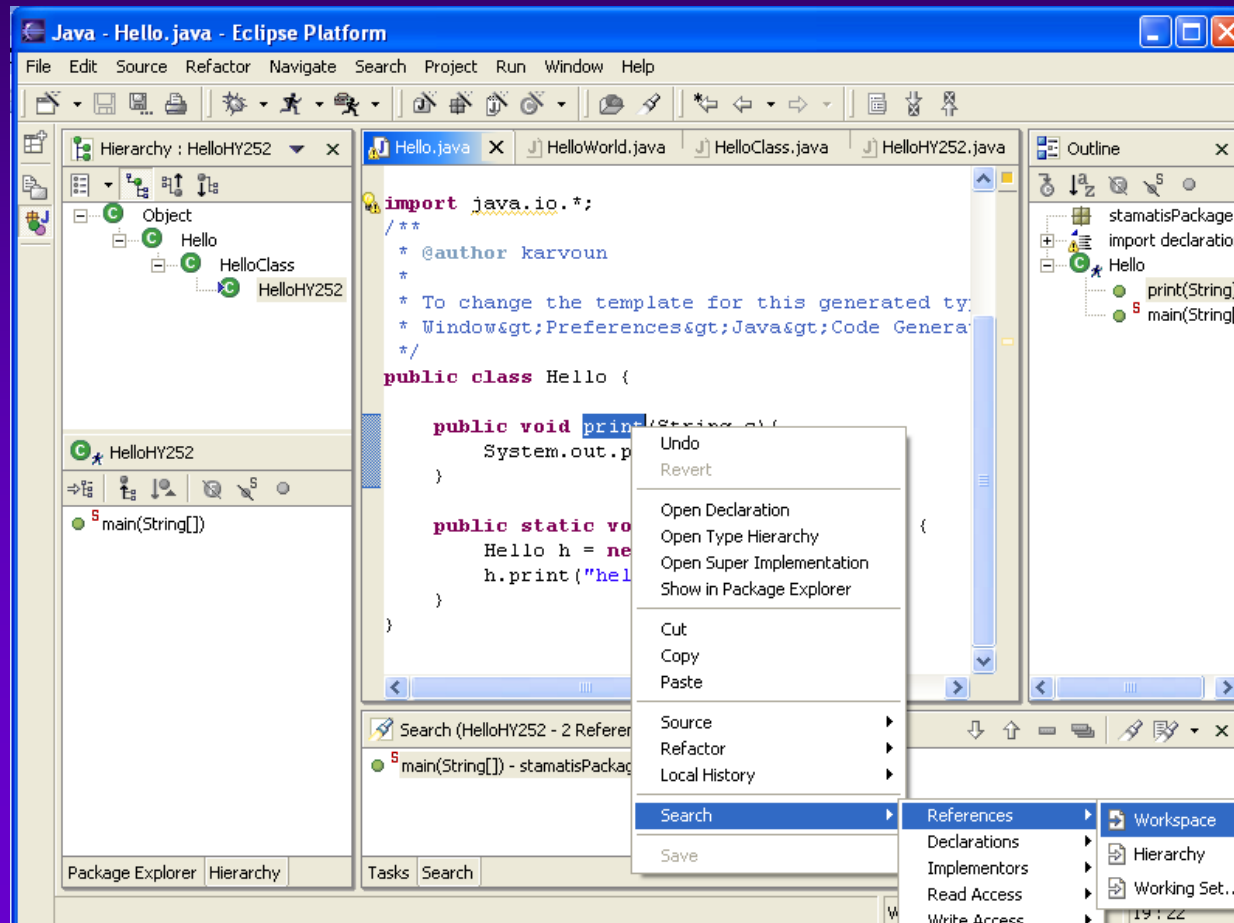
    public static void main(String[] args) {
        Hello h = new Hello();
        h.print("hello");
    }
}
    
```

Search (HelloHY252 - 2 References in Hierarchy - HelloHY252)

- main(String[]) - stamatisPackage.HelloHY252 (2 matches)

# Search for Java Elements

- Select a type or method
- From its pop-up select Search > References > Workspace
- You will retrieve all it's references in the Search View





# Search for Java Elements

The screenshot shows the Eclipse IDE interface with the following components:

- Package Explorer:** Shows a hierarchy starting with 'Object', containing 'Hello', 'HelloClass', and 'HelloHY252'.
- Outline:** Shows the structure of the 'Hello' class, including 'print(String)' and 'main(String[])'.
- Editor:** Displays the source code of 'Hello.java', which includes an import for 'java.io.\*', a class comment, and the implementation of 'Hello' with 'print' and 'main' methods.
- Search Window:** Located at the bottom, it shows the search criteria 'print(String)' and lists 4 references in the workspace:
  - main(String[]) - stamatisPackage.Hello
  - main(String[]) - stamatisPackage.HelloClass
  - main(String[]) - stamatisPackage.HelloHY252
  - main(String[]) - stamatisPackage.HelloWorld

Search  
View

20030331



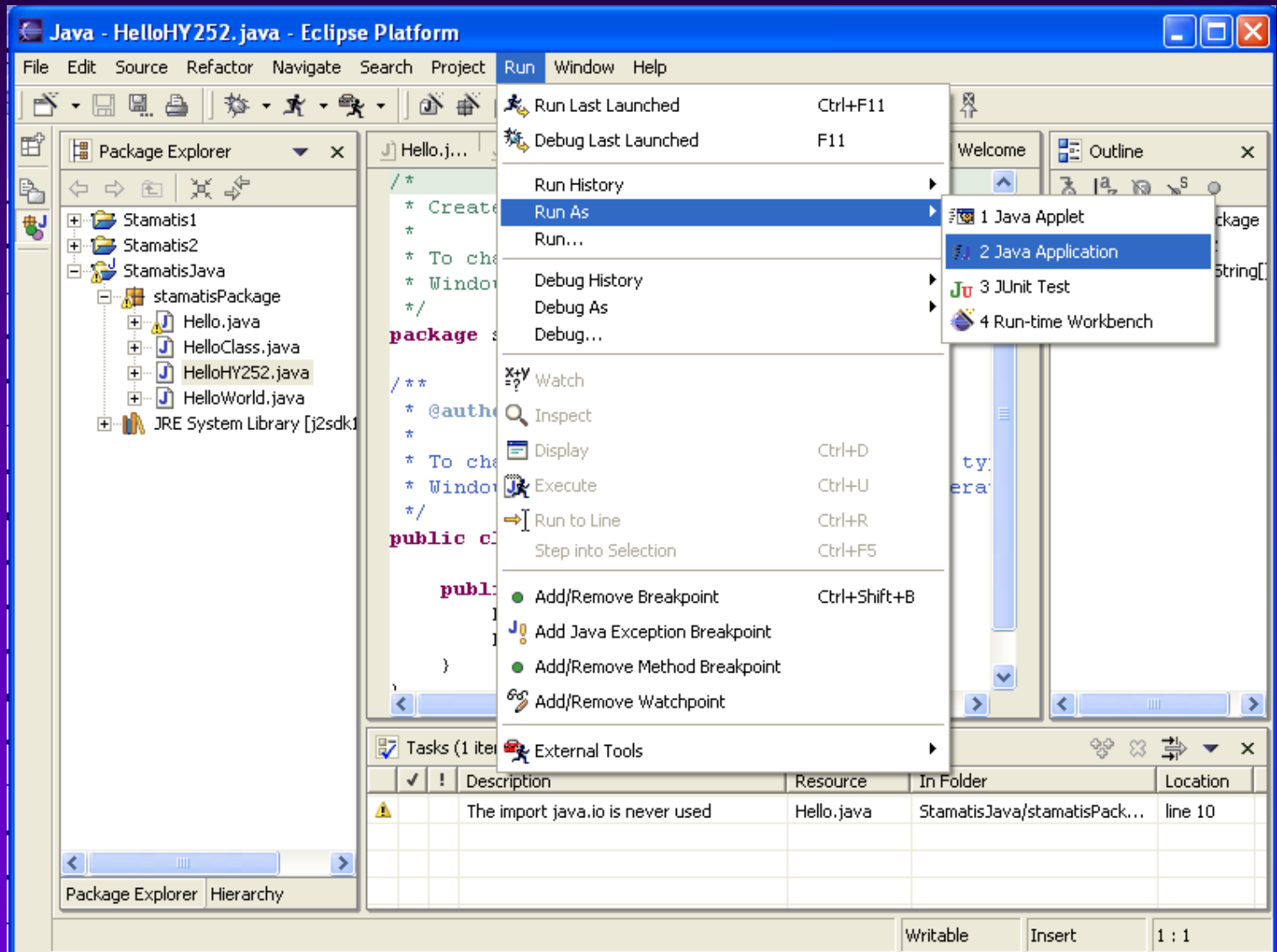
# Run Java Programs

---

- Build is performed automatically upon resource modification
- If you want to do it explicitly select menu Project>Rebuild project
- Two ways to run a project:
  - Select menu Run > Run As > Java Application
  - Click on the "Run" icon in the toolbar



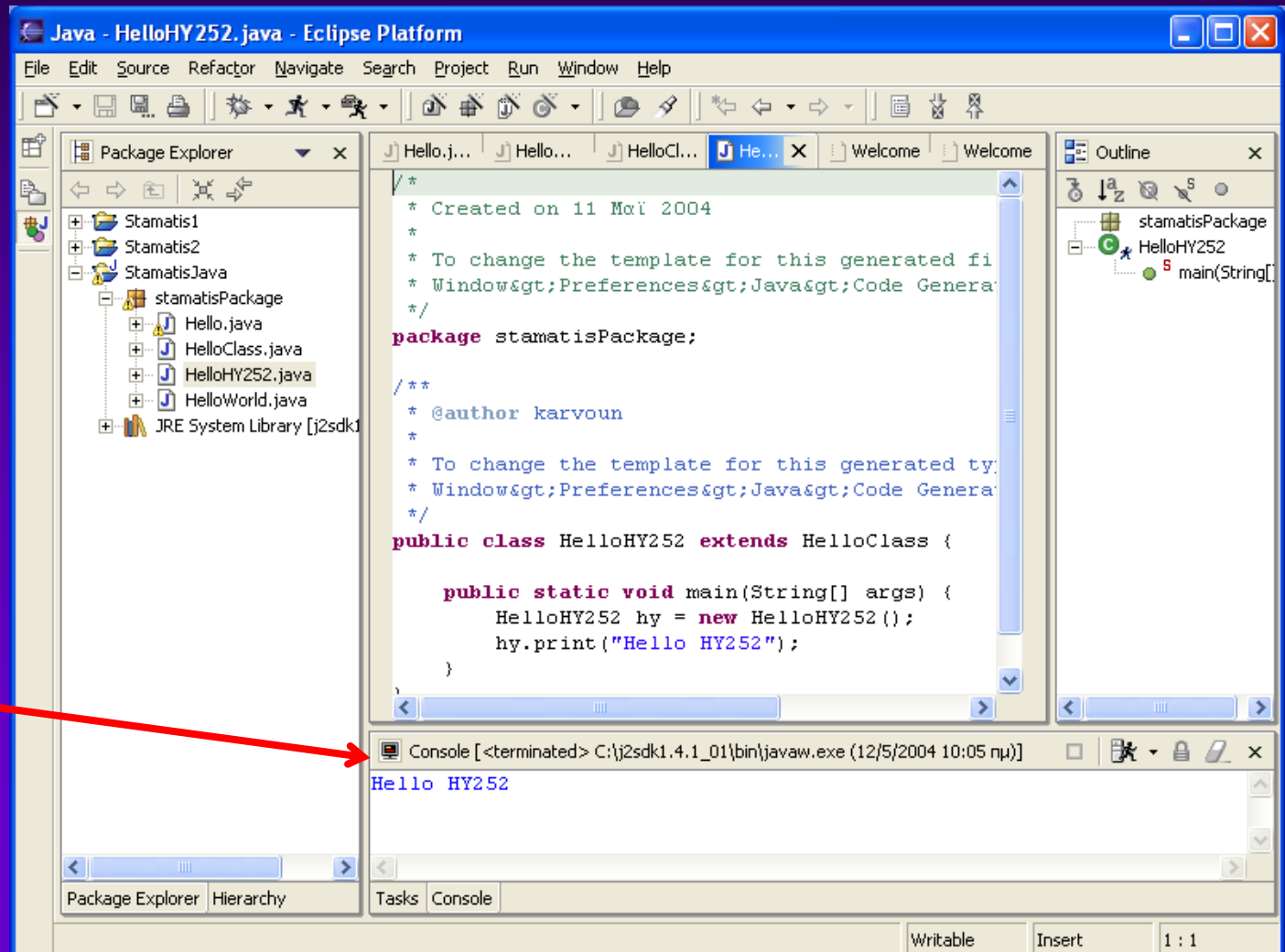
# Run Java Programs





# Run Java Programs

- The Console View shows up with the output!



Console View



# Part4

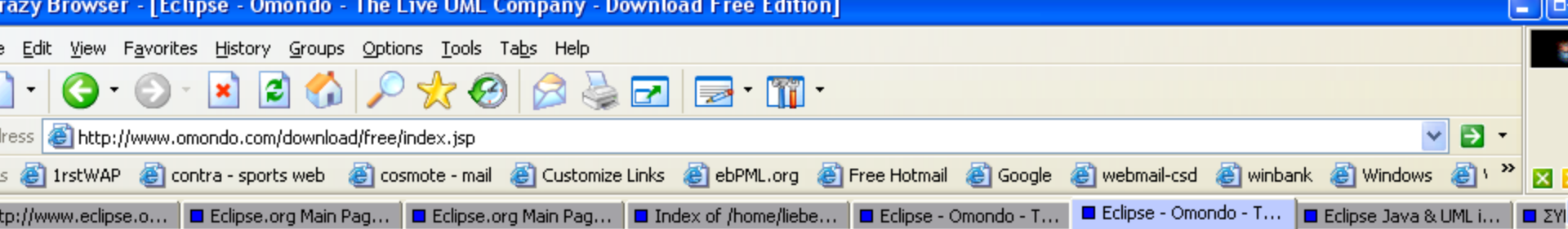
UML



# Eclipse UML Installation

---

- Exit eclipse
- Download EMF (Eclipse Modeling Framework) zip file from <http://www.omondo.com/download/free/index.jsp>
- Unzip EMF zip in %ECLIPSE\_HOME% directory
- Re-start eclipse
- Exit eclipse



## Zip File

Eclipse 2.1.1 is required.

- [Get Eclipse 2.1.1 R-2.1.1-200306271545.](#)

The EMF (Eclipse Modeling Framework ) plugin is required.

- [Get EMF 1.1.0 Build 20030620 1105VL.](#)

Download this

The proper GEF (Graphical Editor Framework) plugin is included in our zip file.

Previous GEF installation should be uninstalled.

TYPE	VERSION	DATE	SIZE
EclipseUML	1.2.1.20031103	11/04/2003	9 932 806 Bytes
EclipseUML	1.2.1.20030806	08/06/2003	10 769 882 Bytes

Eclipse 2.1 is required.

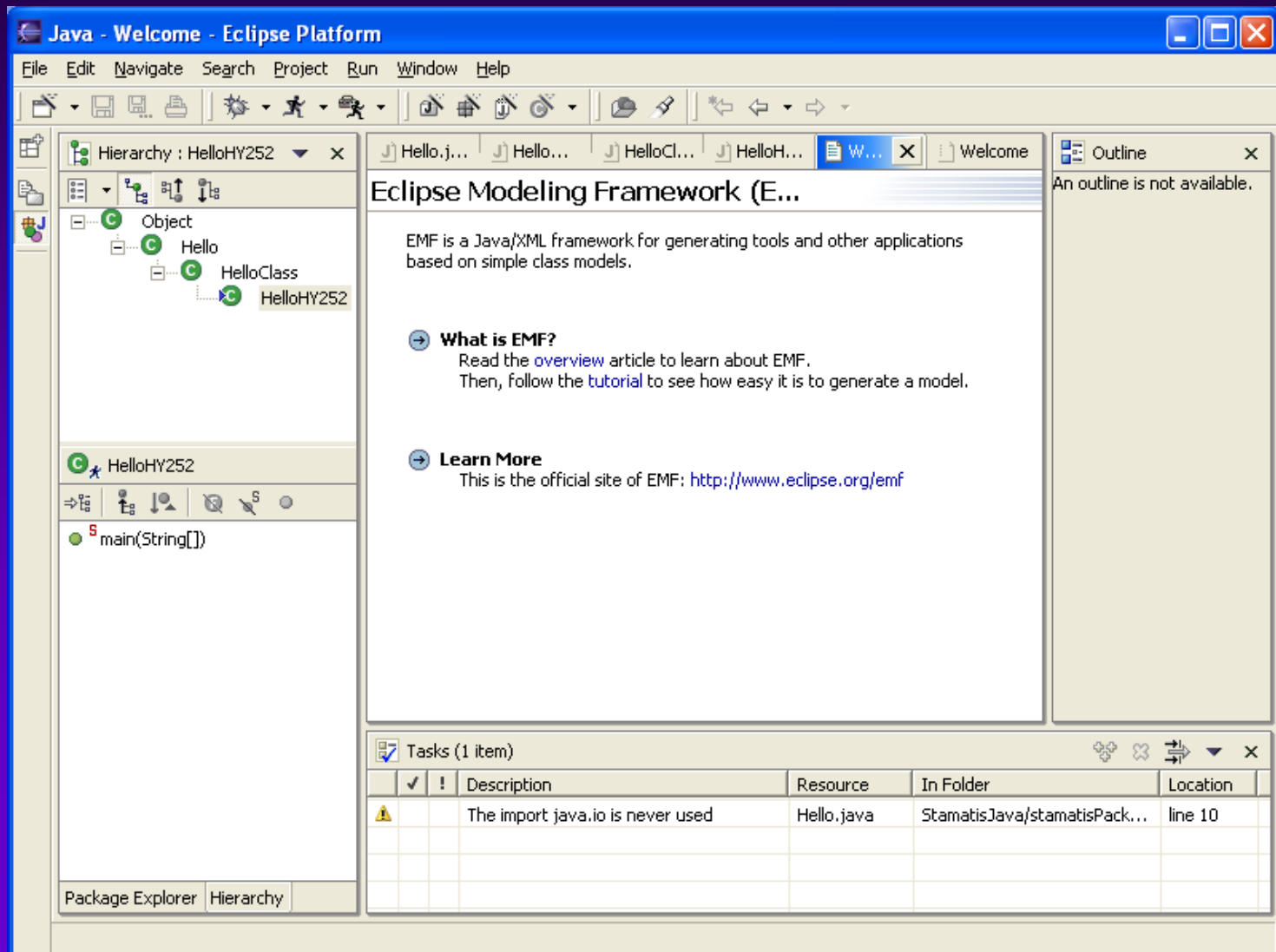
- [Get Eclipse 2.1 R-2.1-200303272130.](#)

The EMF (Eclipse Modeling Framework ) plugin is required.

- [Get EMF 1.1.0 Build 20030620 1105VL.](#)

The proper GEF (Graphical Editor Framework) plugin is included in our zip file.

# Eclipse UML Installation

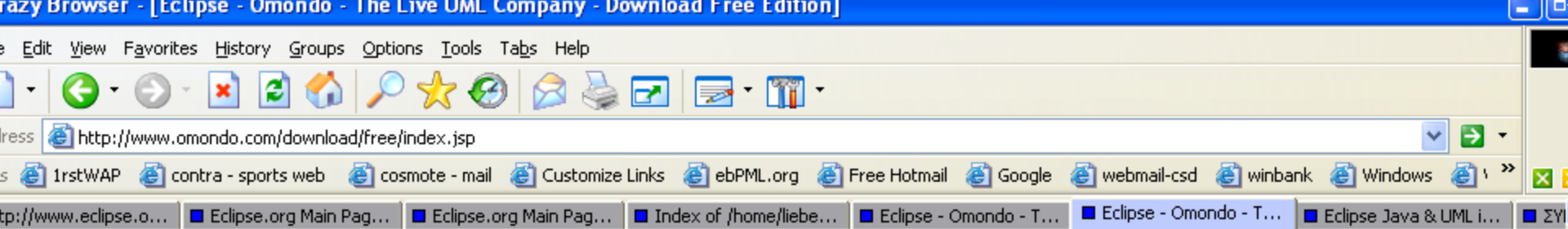




# Eclipse UML Installation

---

- Download Eclipse UML zip file from <http://www.omondo.com/download/free/index.jsp>
  - Unzip Eclipse UML zip in %ECLIPSE\_HOME% directory
  - Re-start eclipse
  - That's it!
- 
- More details in:
    - <http://www.omondo.com/faq/free/index.jsp#emfzipi>
    - [http://www.lifl.fr/~offroy/eclipse\\_HTML/internal/installation/Eclipse\\_Java\\_&\\_UML\\_install.html](http://www.lifl.fr/~offroy/eclipse_HTML/internal/installation/Eclipse_Java_&_UML_install.html)



## Zip File

Eclipse 2.1.1 is required.

- [Get Eclipse 2.1.1 R-2.1.1-200306271545.](#)

The EMF (Eclipse Modeling Framework ) plugin is required.

- [Get EMF 1.1.0 Build 20030620 1105VL.](#)

The proper GEF (Graphical Editor Framework) plugin is included in our zip file.

**Download This**  
Previous GEF installation should be uninstalled.

TYPE	VERSION	DATE	SIZE
EclipseUML	<b>1.2.1.20031103</b>	11/04/2003	9 932 806 Bytes
EclipseUML	1.2.1.20030806	08/06/2003	10 769 882 Bytes

Eclipse 2.1 is required.

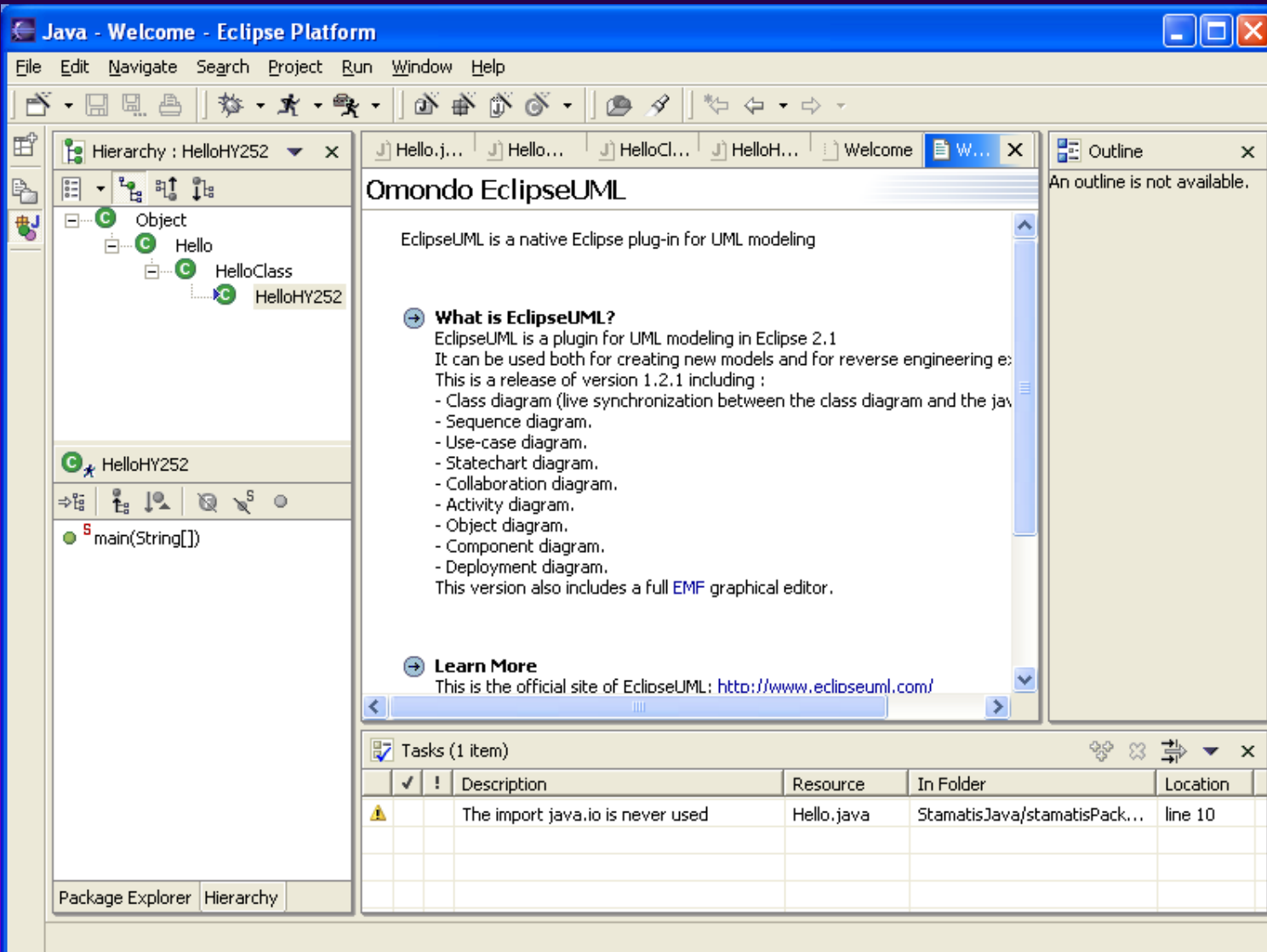
- [Get Eclipse 2.1 R-2.1-200303272130.](#)

The EMF (Eclipse Modeling Framework ) plugin is required.

- [Get EMF 1.1.0 Build 20030620 1105VL.](#)

The proper GEF (Graphical Editor Framework) plugin is included in our zip file.

# Eclipse UML Installation



The screenshot shows the Eclipse IDE interface with the EclipseUML plugin documentation open in the main editor. The Package Explorer on the left shows a project named 'HelloHY252' with a class hierarchy: Object -> Hello -> HelloClass -> HelloHY252. The main editor displays the 'Omondo EclipseUML' page, which includes a title, a description, and a list of supported UML diagram types. The Tasks view at the bottom shows a warning for an unused import.

**Omondo EclipseUML**

EclipseUML is a native Eclipse plug-in for UML modeling

**What is EclipseUML?**  
 EclipseUML is a plugin for UML modeling in Eclipse 2.1  
 It can be used both for creating new models and for reverse engineering ex:  
 This is a release of version 1.2.1 including :

- Class diagram (live synchronization between the class diagram and the java)
- Sequence diagram.
- Use-case diagram.
- Statechart diagram.
- Collaboration diagram.
- Activity diagram.
- Object diagram.
- Component diagram.
- Deployment diagram.

This version also includes a full EMF graphical editor.

**Learn More**  
 This is the official site of EclipseUML: <http://www.eclipseuml.com/>

**Tasks (1 item)**

	✓	!	Description	Resource	In Folder	Location
⚠			The import java.io is never used	Hello.java	StamatisJava/stamatisPack...	line 10

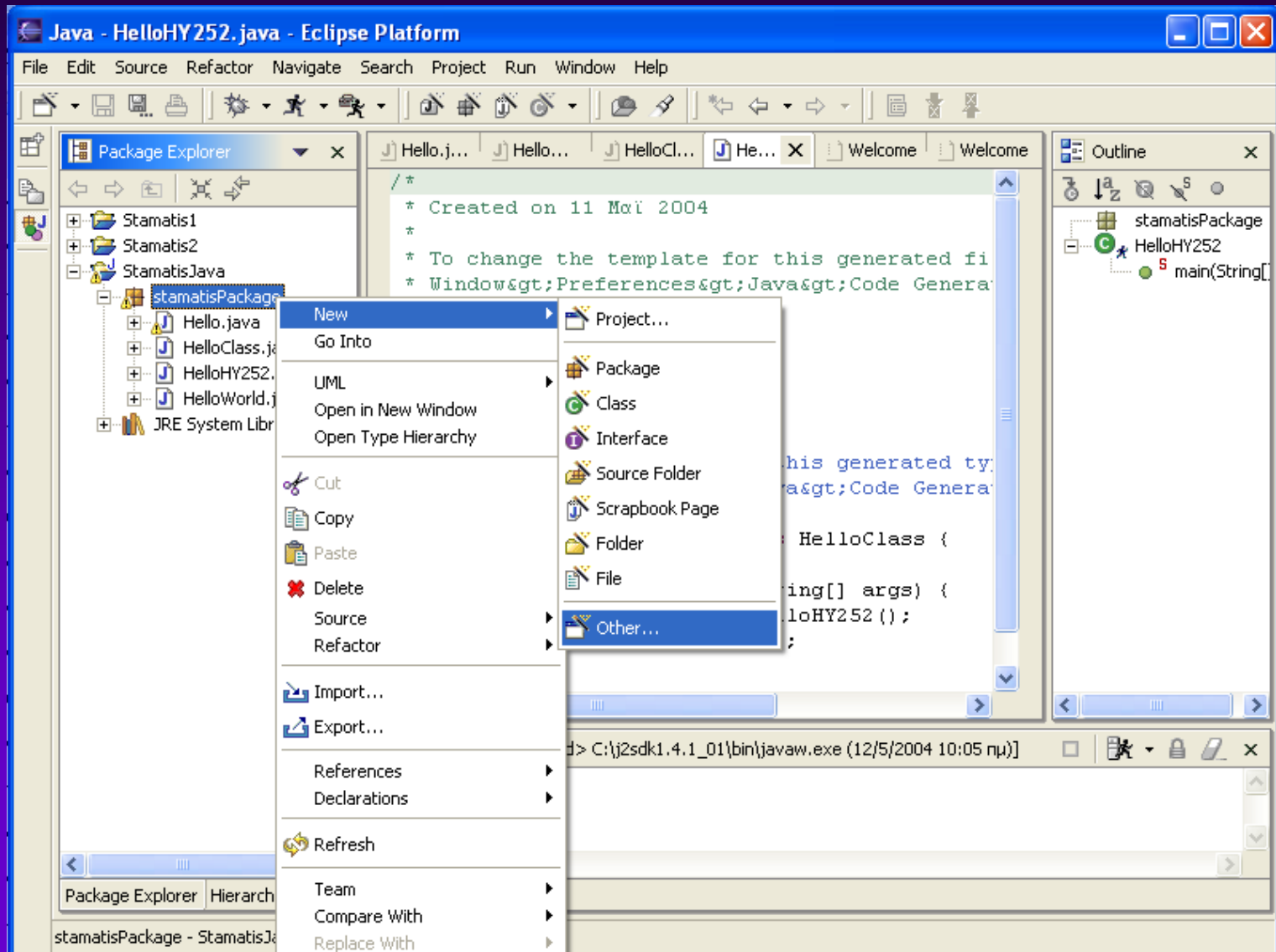


# Create your first class diagram

---

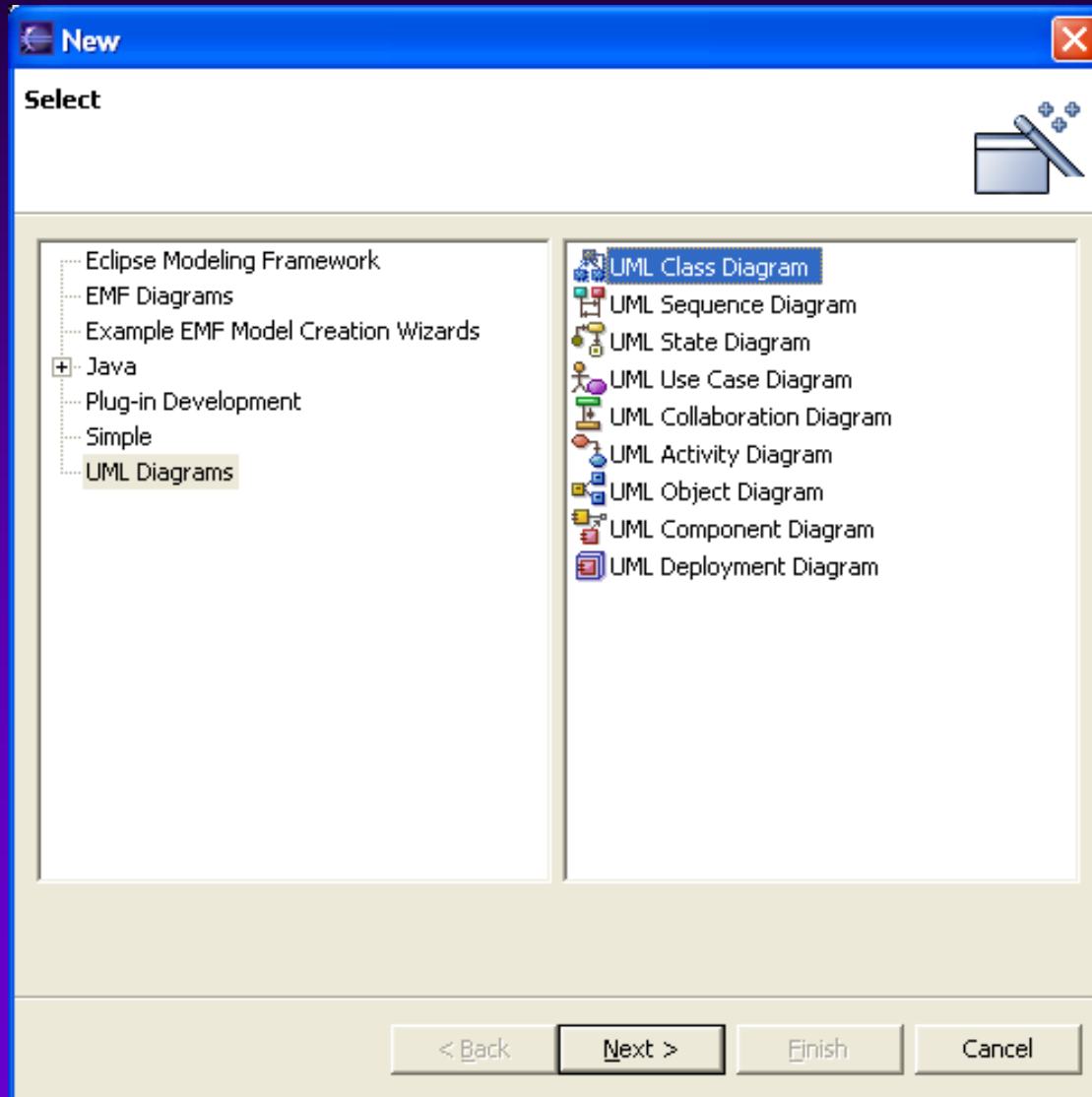
- Select a package
- From its context menu (right-click) select New>Other>UML->UML Class Diagram

# Create your first class diagram





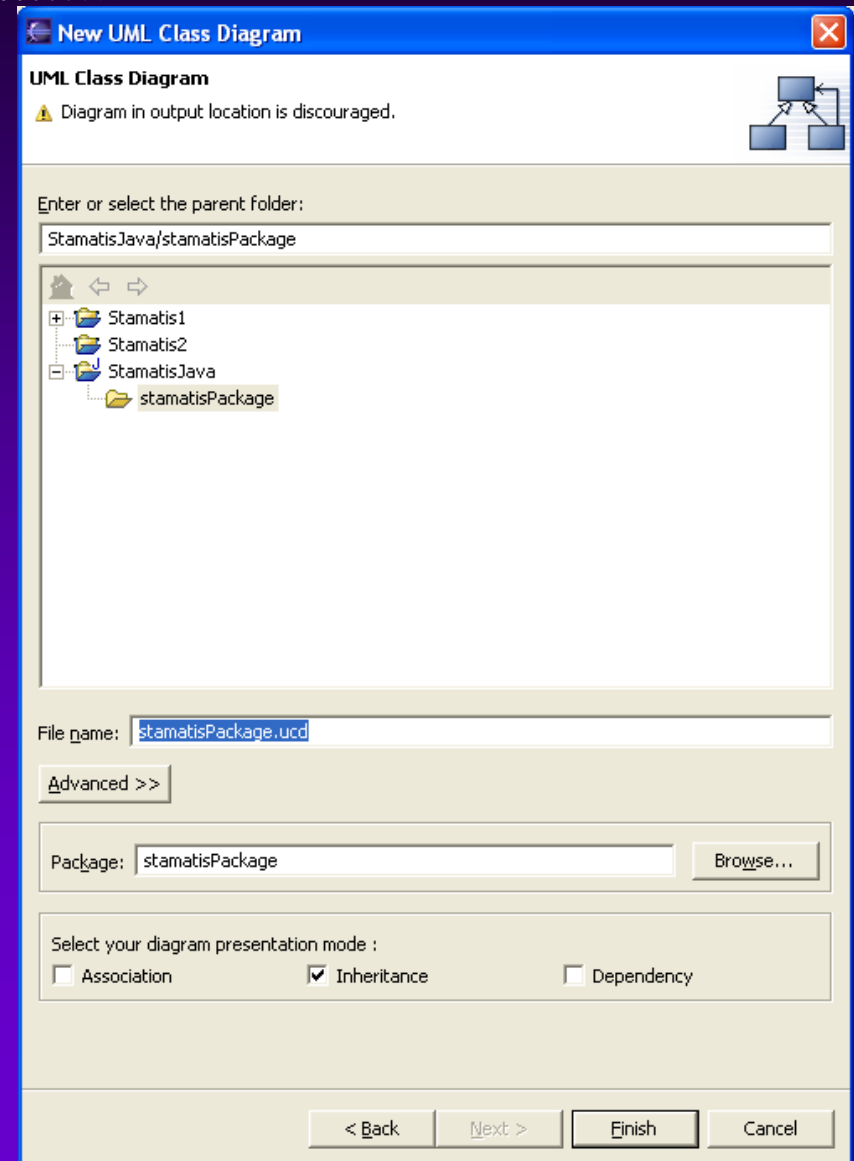
# Create your first class diagram





# Create your first class diagram

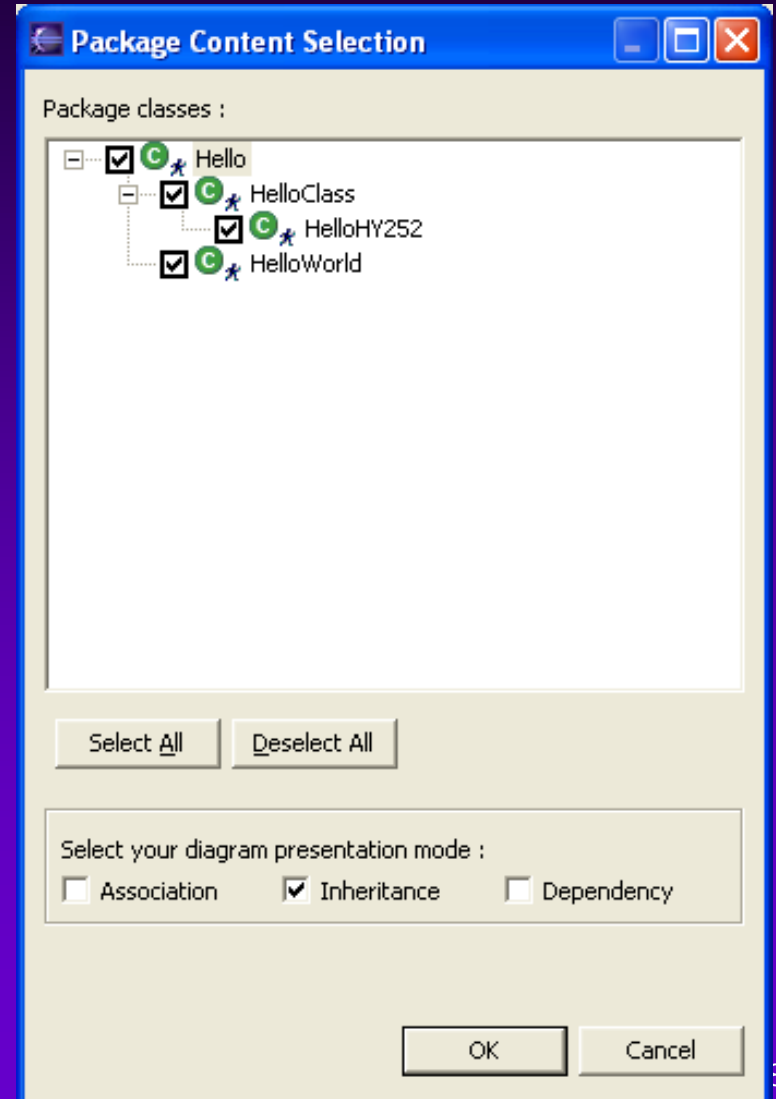
- Choose the kind of relationship you want to be depicted
  - Association
  - Inheritance
  - Dependency





# Create your first class diagram

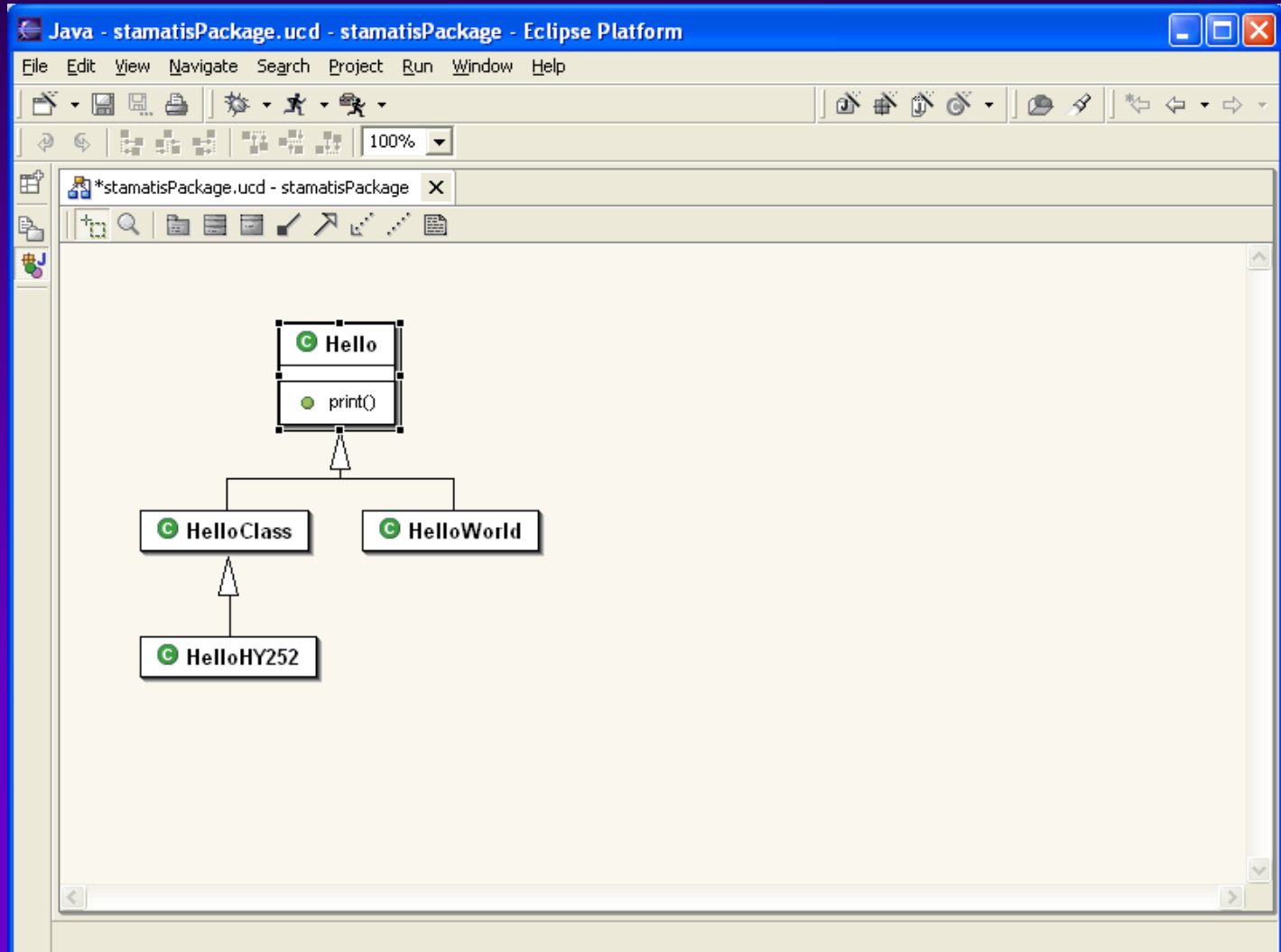
- Choose the classes that will participate in the diagram





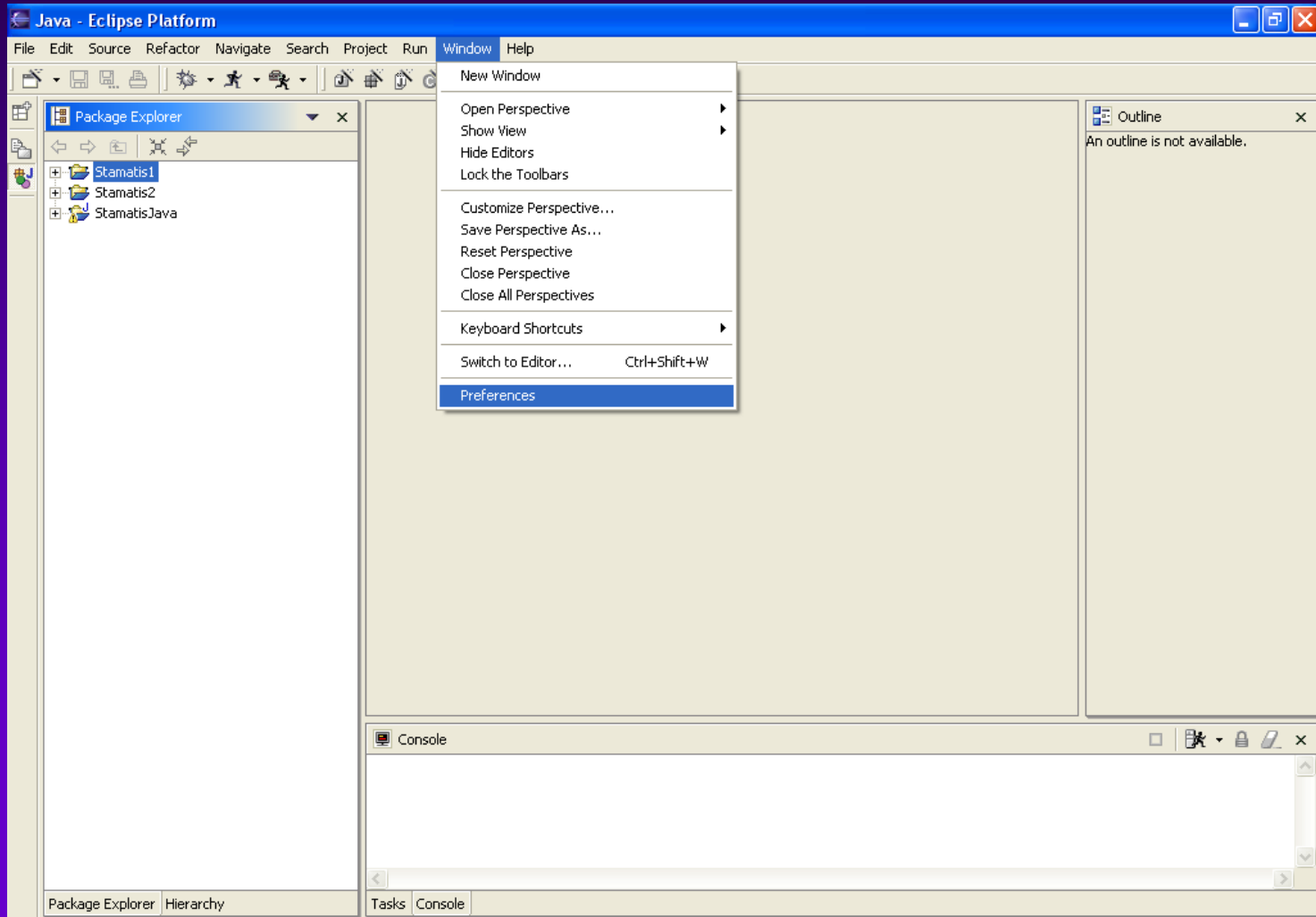
# Create your first class diagram

- Here is it!



# Prepare Workbench for UML design

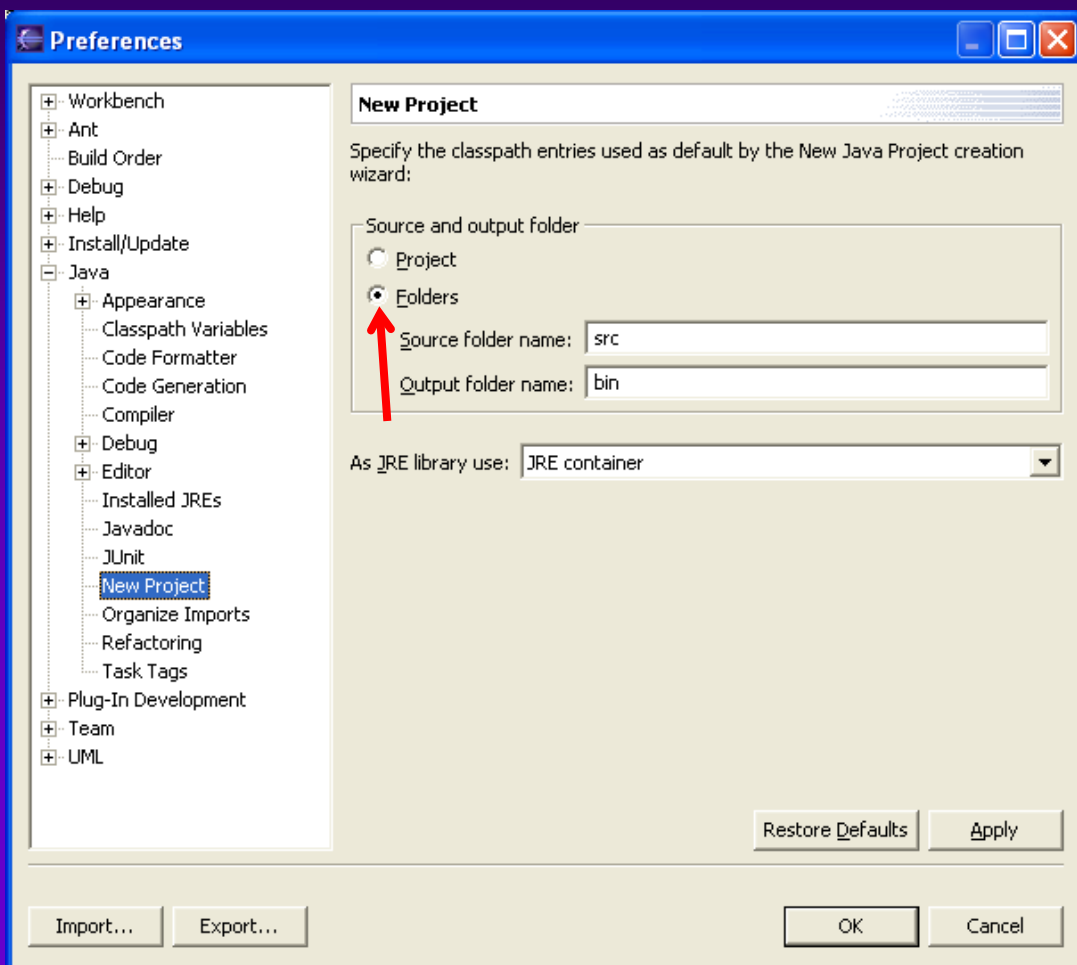
- Select menu **Window > Preferences**





# Prepare Workbench for UML design

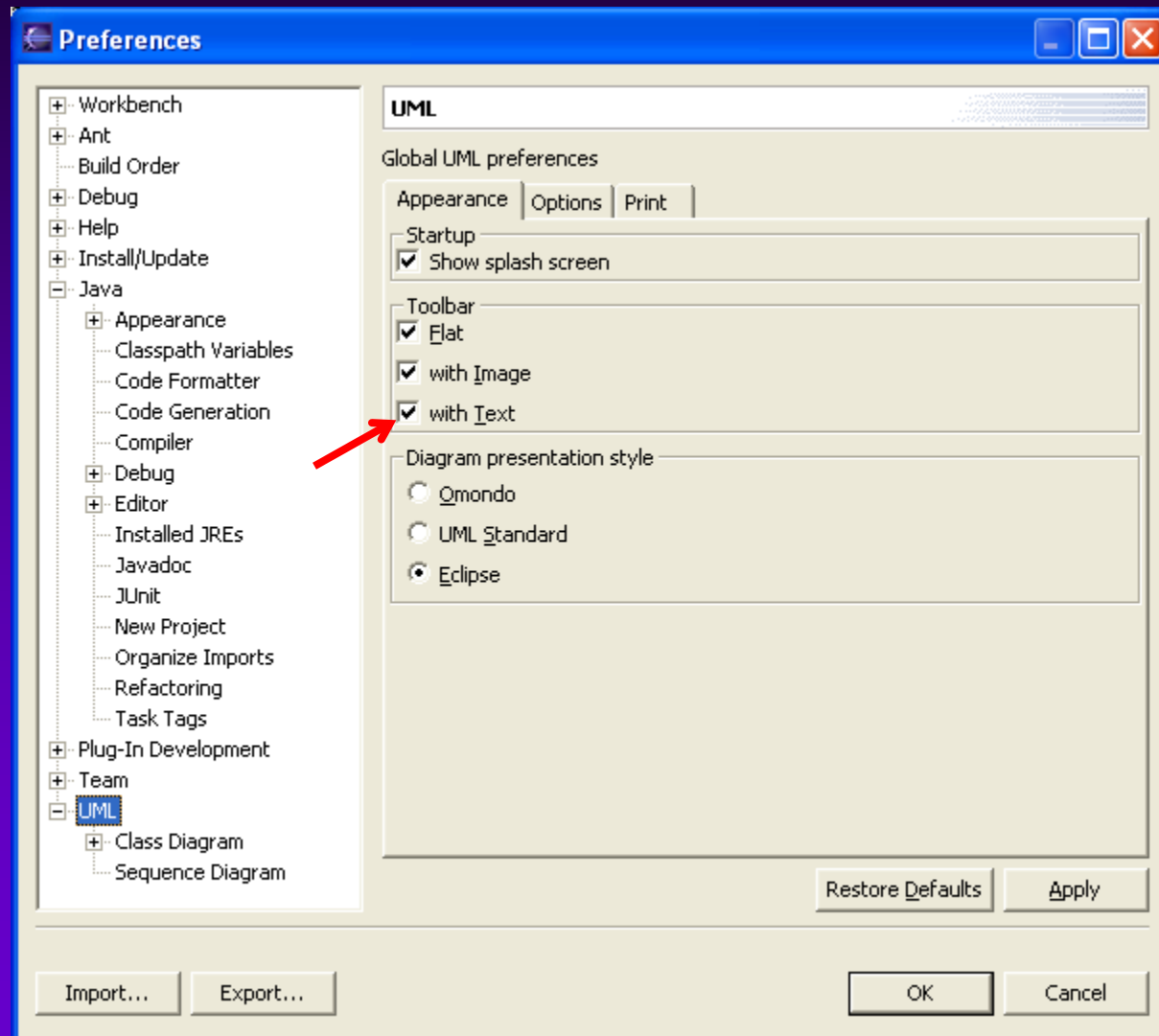
- We will create a src and a bin output for the UML project
- Select Java > New Projects
- Click on "Folders" checkbox





# Prepare Workbench for UML design

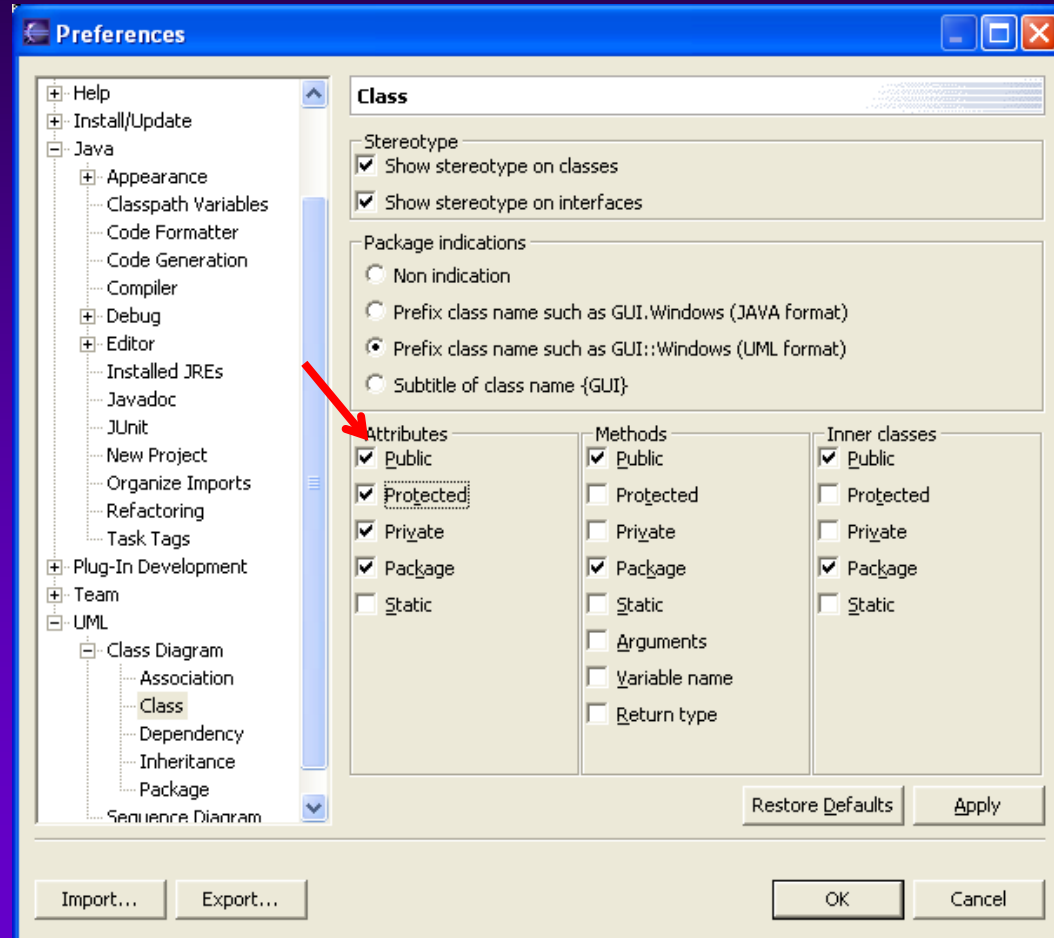
- Select "UML"
- Click on checkbox "with text"





# Prepare Workbench for UML design

- Select UML > Class Diagrams > Class
- In attributes pane check boxes "Public", "Private", "Protected" and "Package"

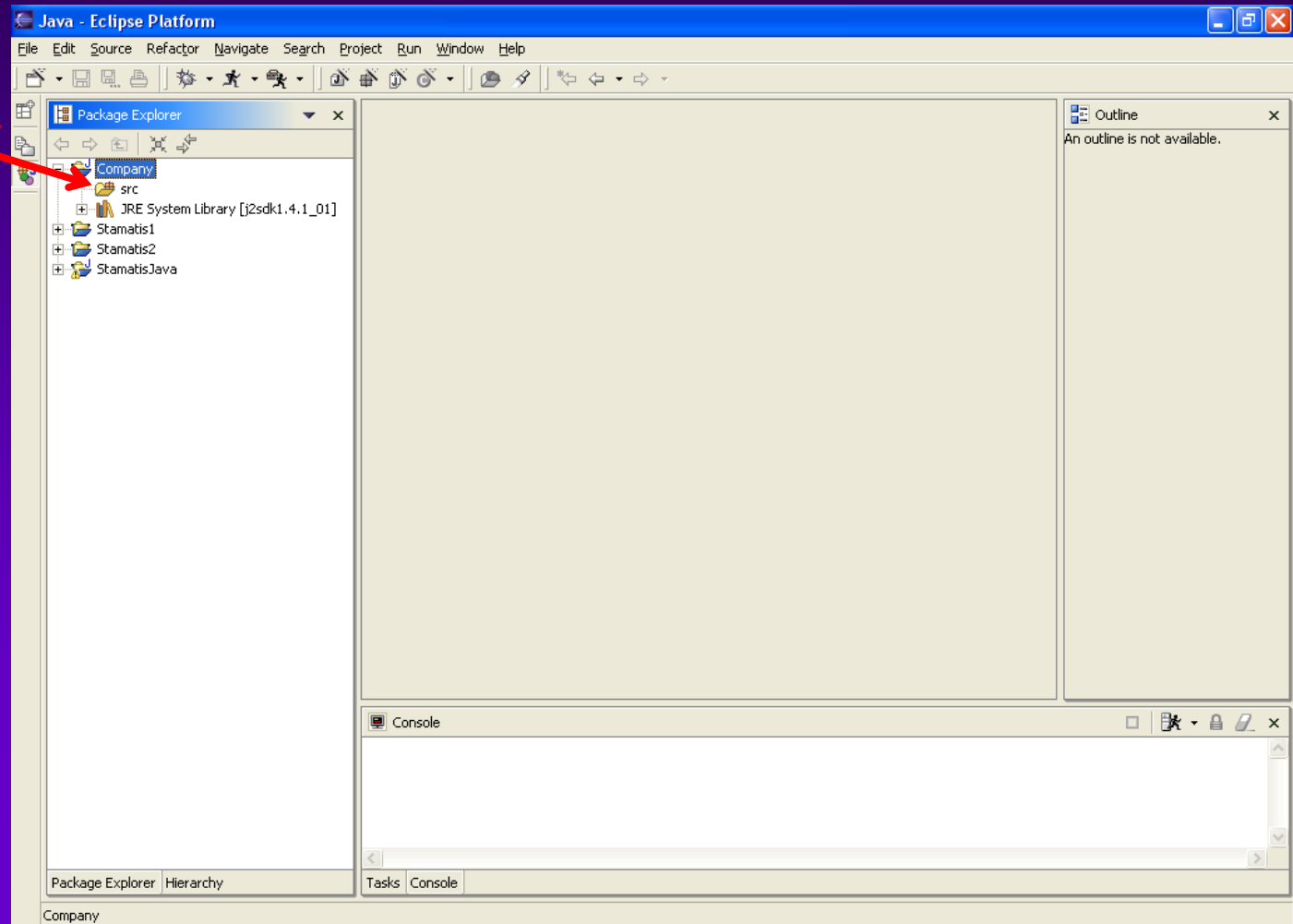




# Eclipse UML Class Diagrams

- Select Java Perspective
- Create a new Java project named "Company"

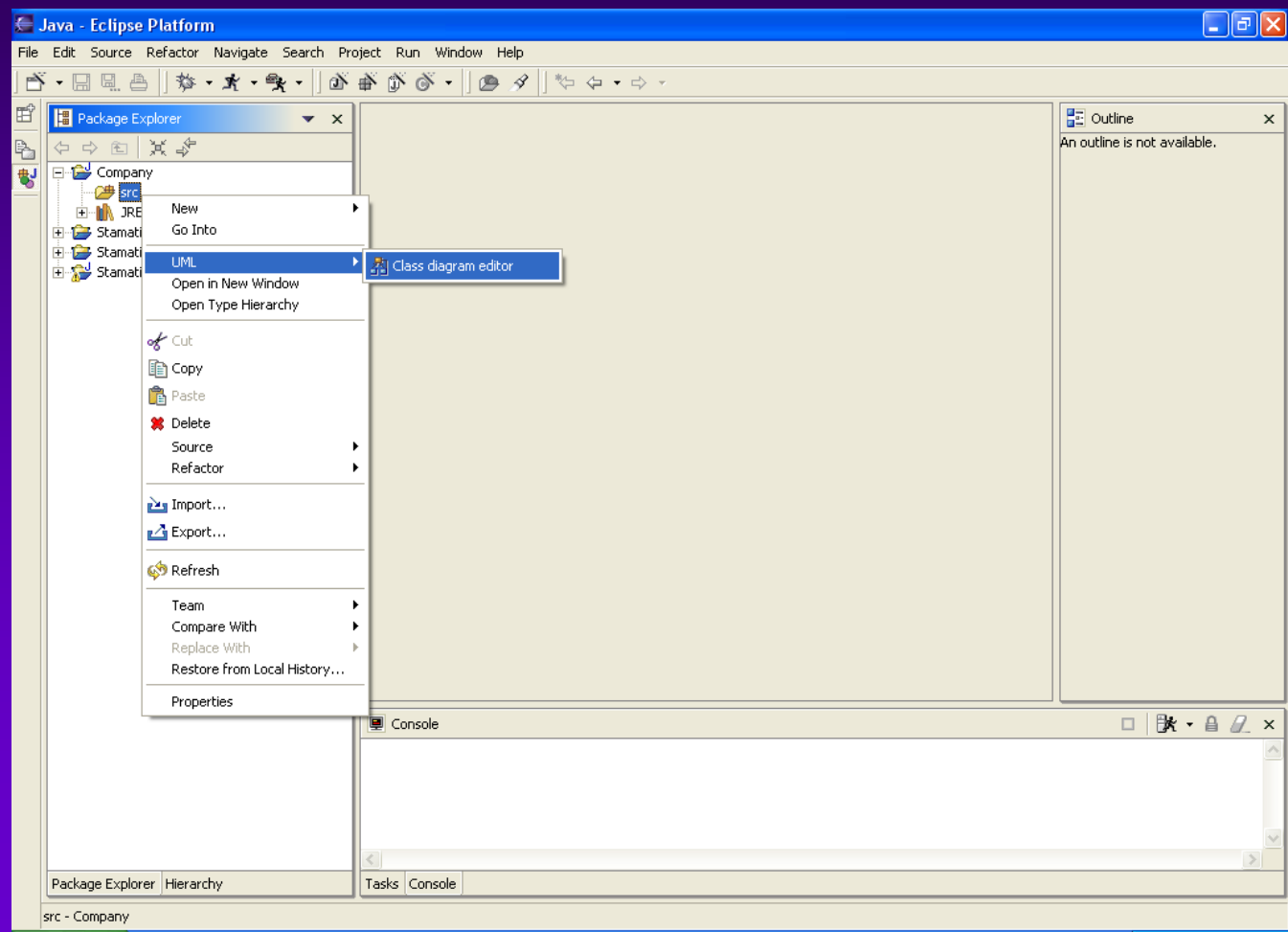
The src folder is created automatically



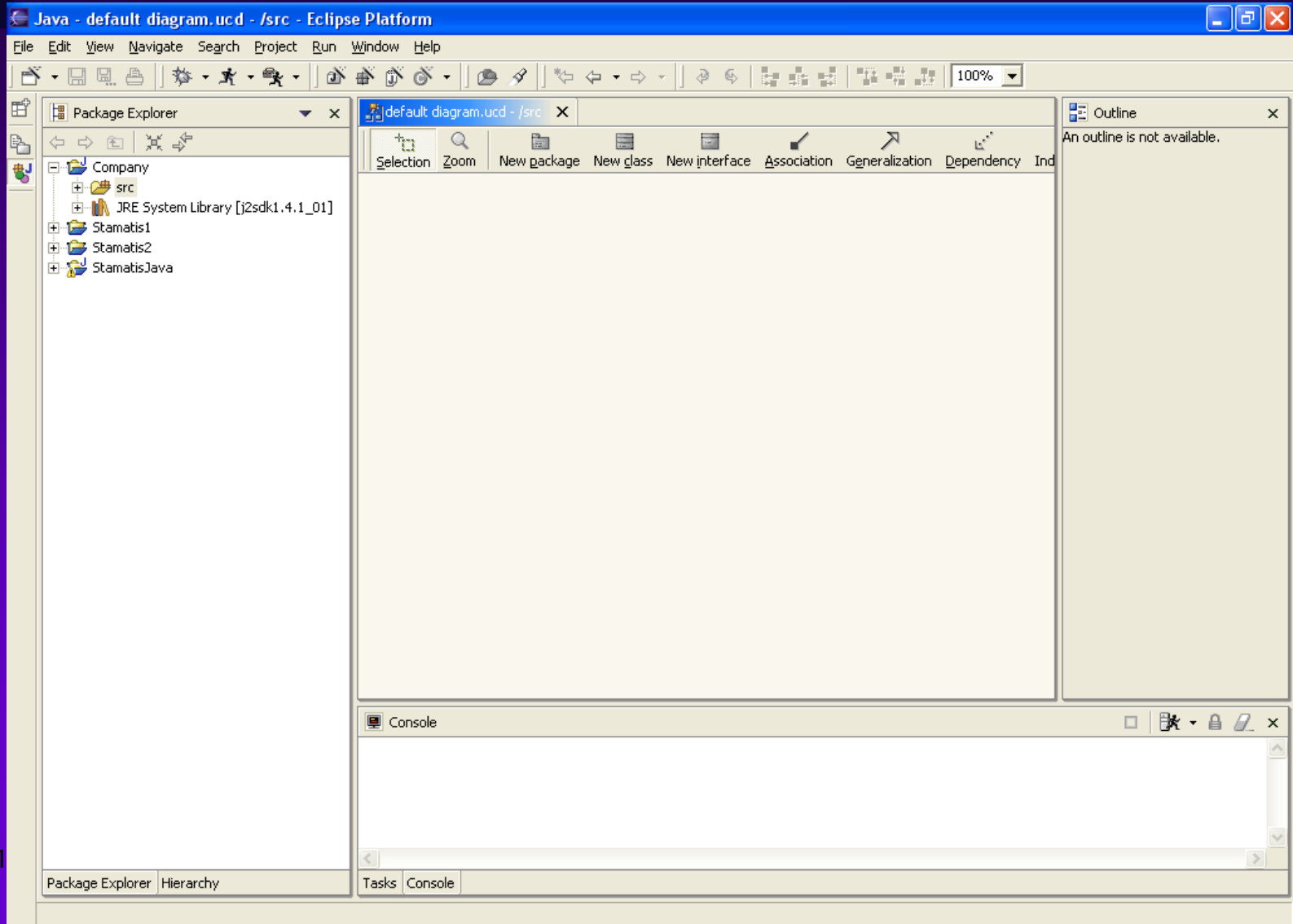


# Eclipse UML Class Diagrams

- Select src folder
- From it's pop-up menu select UML>Class diagram editor



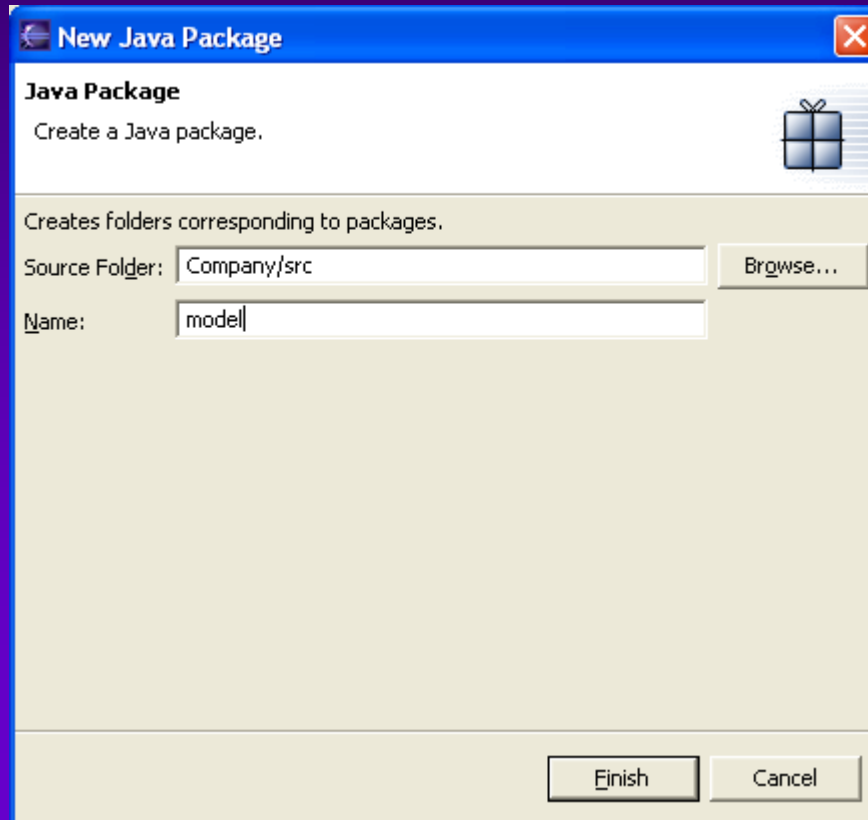
# Eclipse UML Class Diagrams



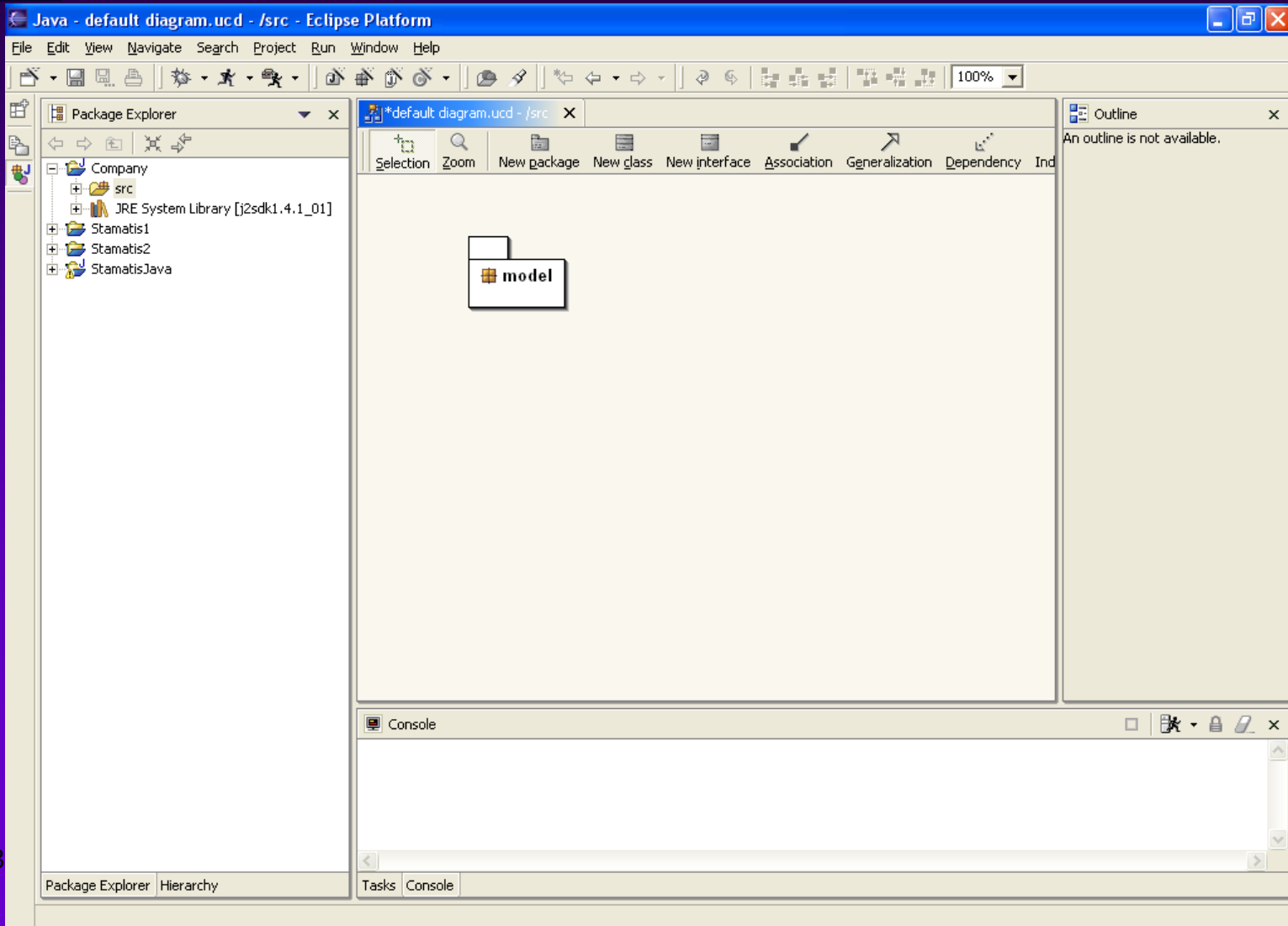


# Eclipse UML Class Diagrams

- Create a New Package.
- Select "new package" in the toolbar and click on the diagram.
- Name it "model"

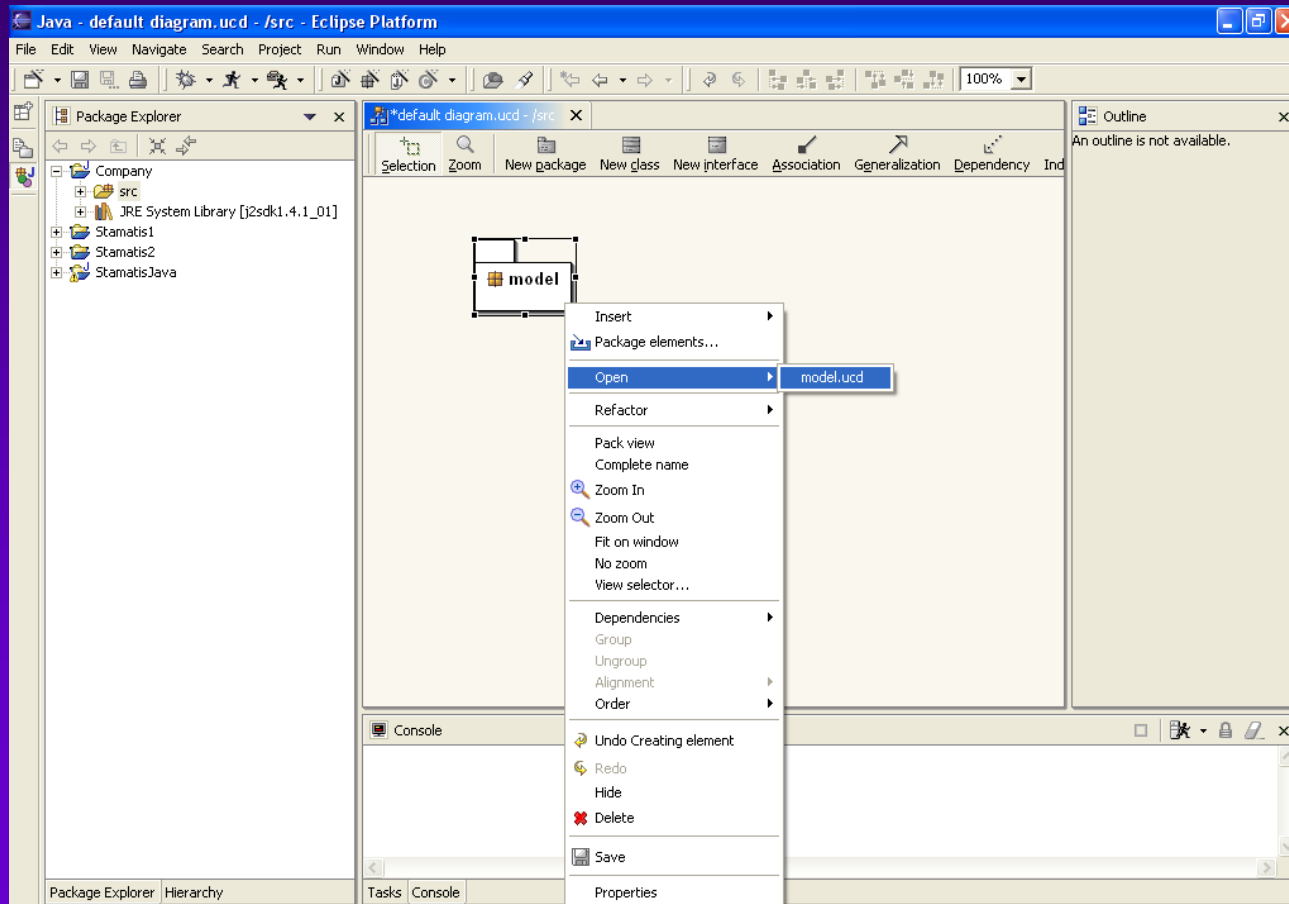


# Eclipse UML Class Diagrams

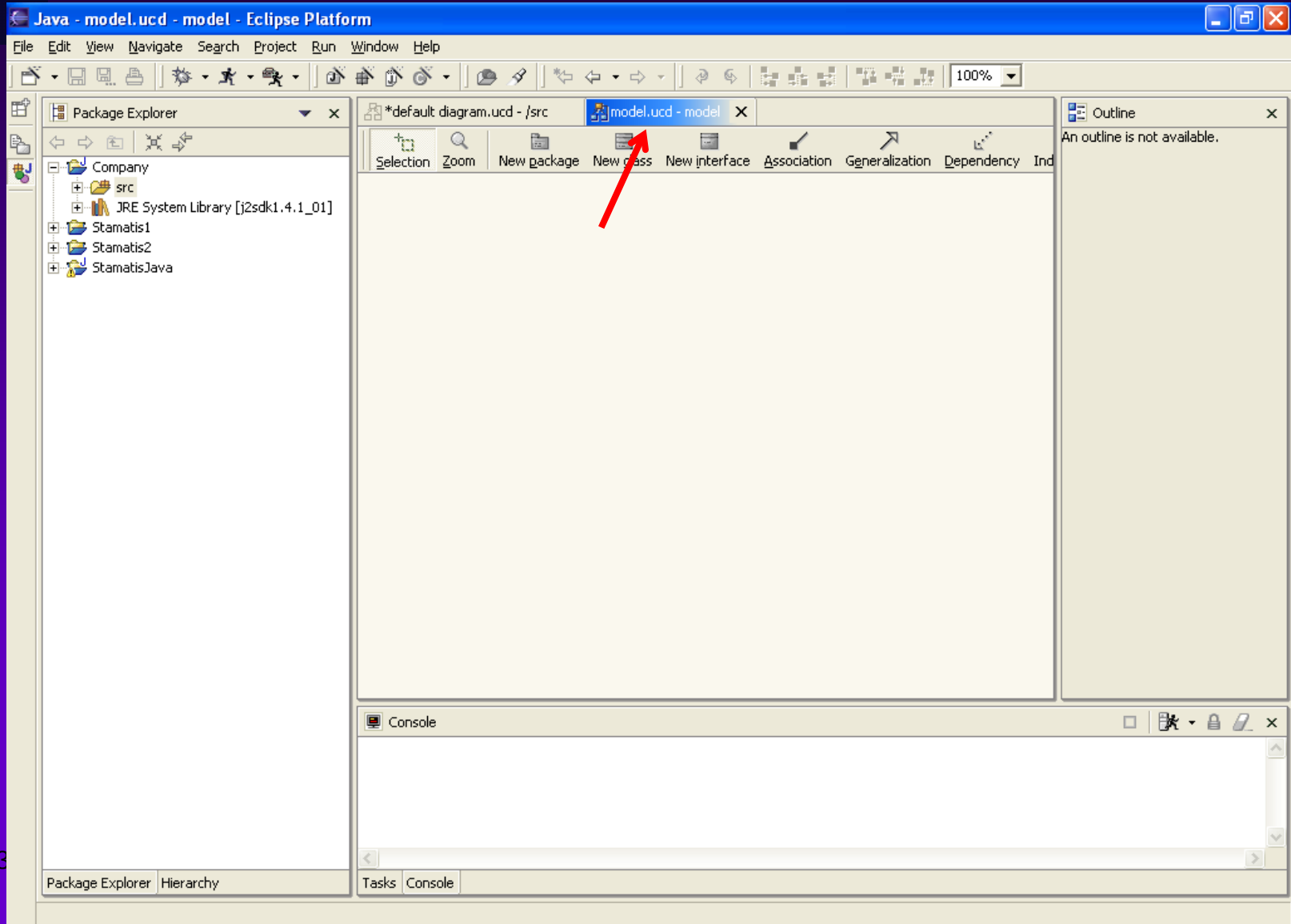


# Eclipse UML Class Diagrams

- Create a new Class Diagram inside the "model" Package
- Right-click on model then Open>model.ucd (ucd is the class diagram extension).



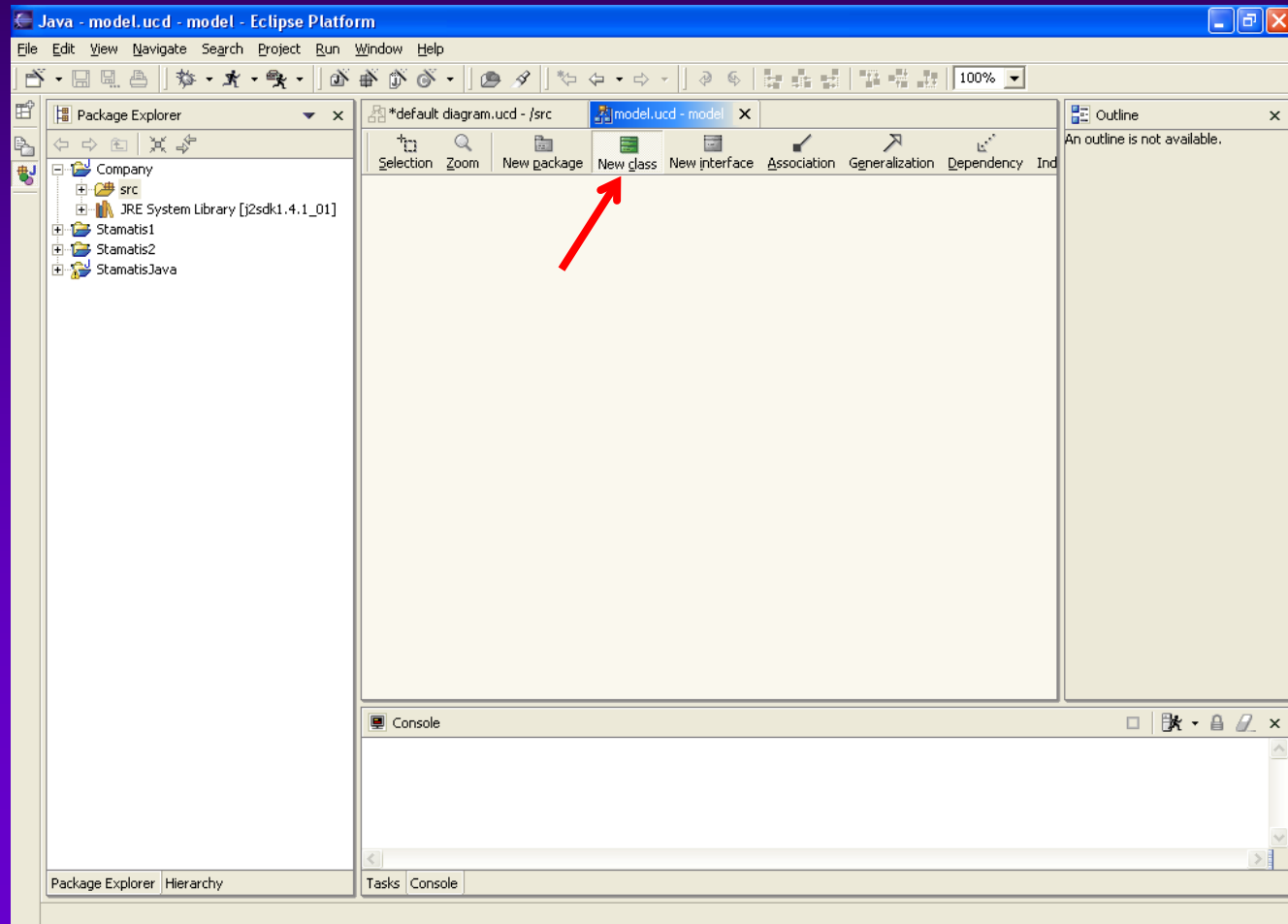
# Eclipse UML Class Diagrams





# Eclipse UML Class Diagrams

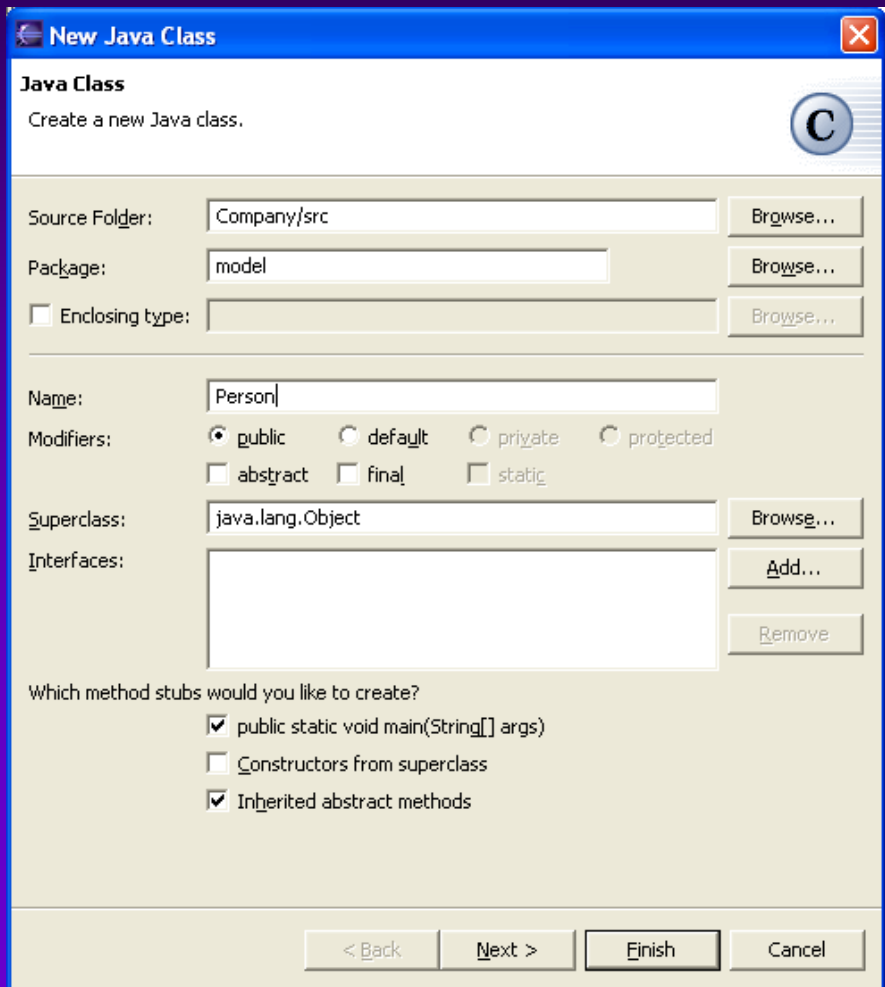
- Create a New class
- Select New class in the toolbar and click on the diagram.



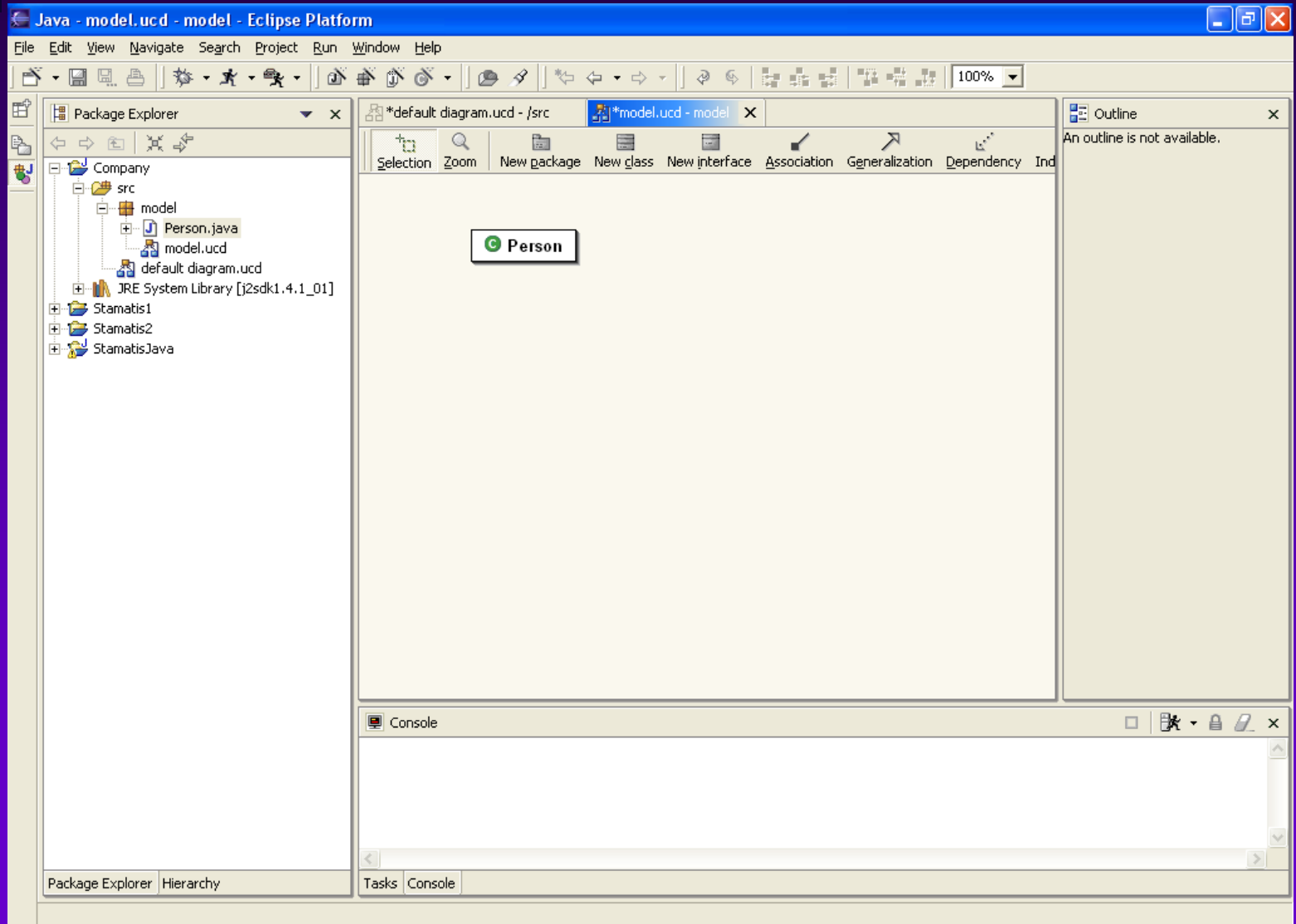


# Eclipse UML Class Diagrams

- Enter the name of the New Java Class and check
- Enter Person and click on the Finish button.

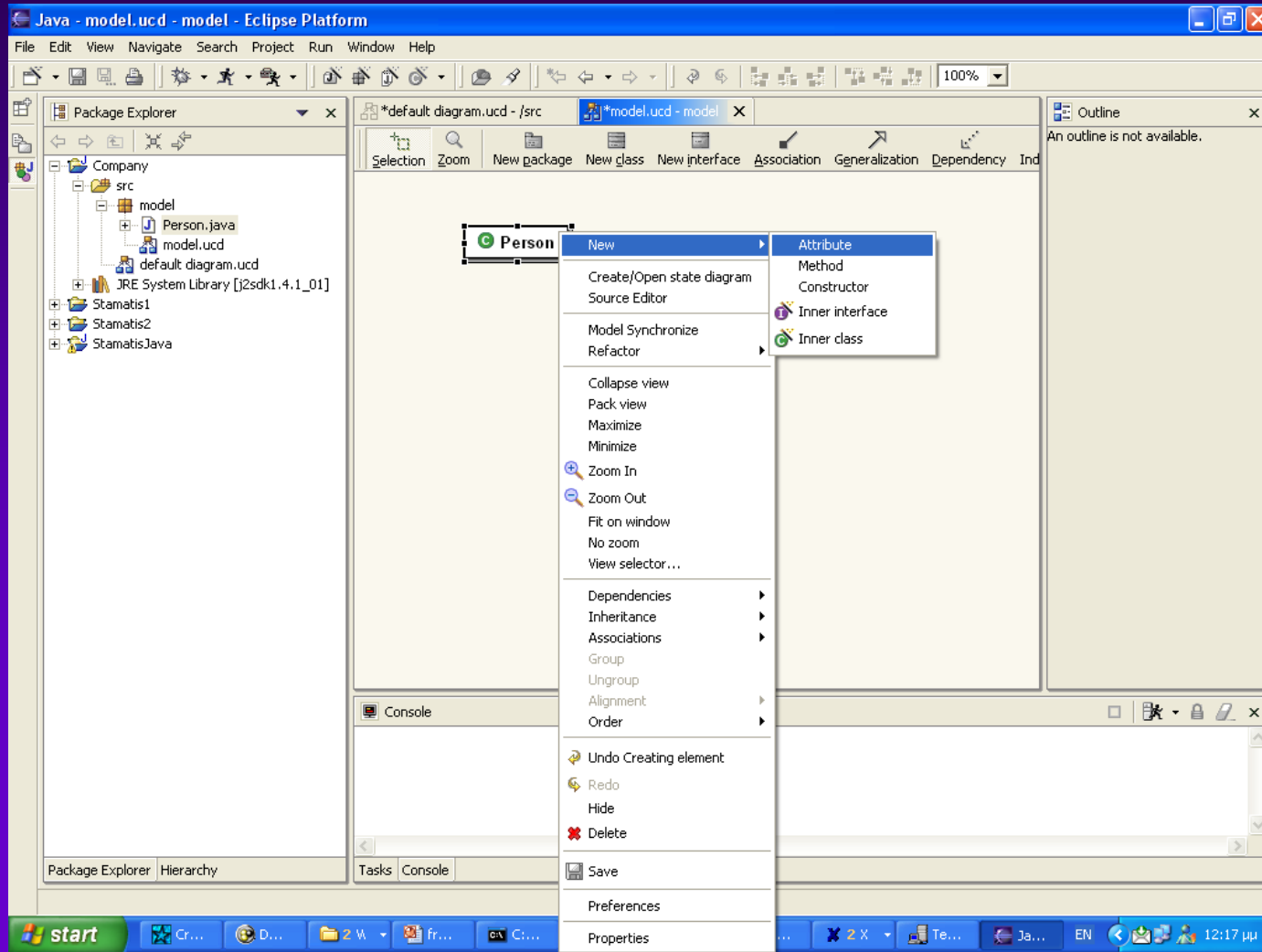


# Eclipse UML Class Diagrams



# Eclipse UML Class Diagrams

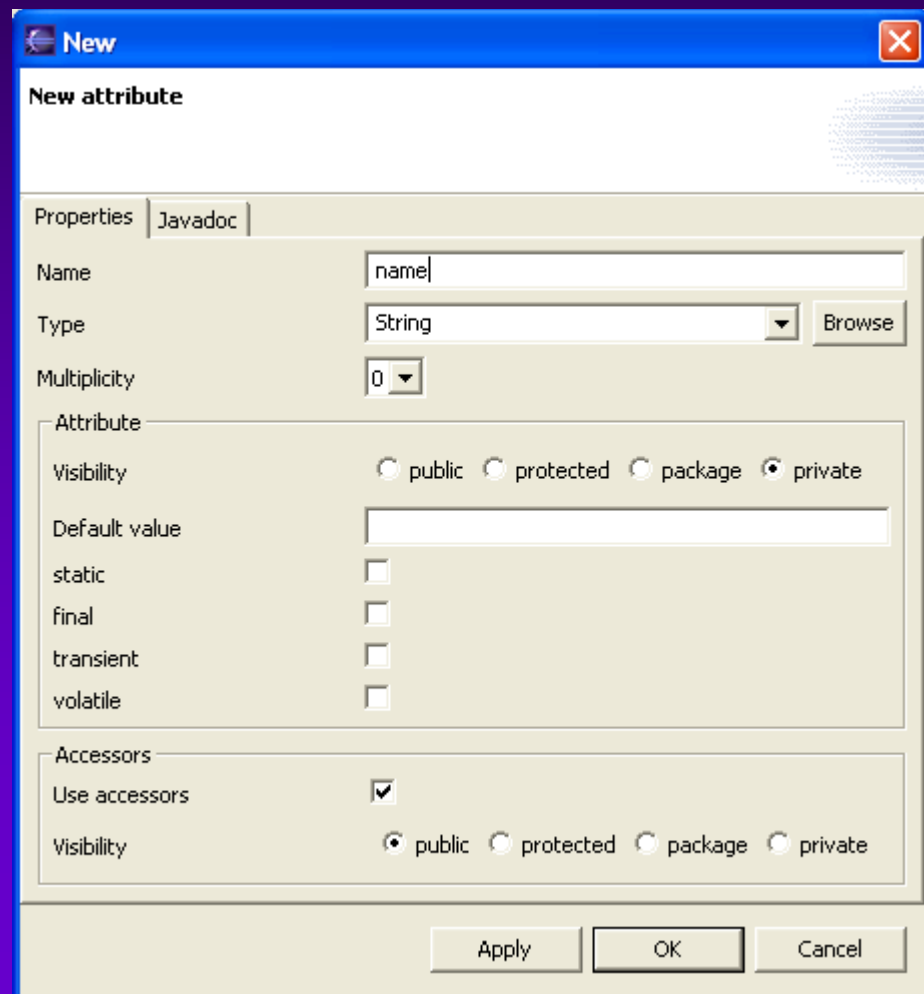
- We are going to add two new Attributes.
- Click on Person>New>Attribute.



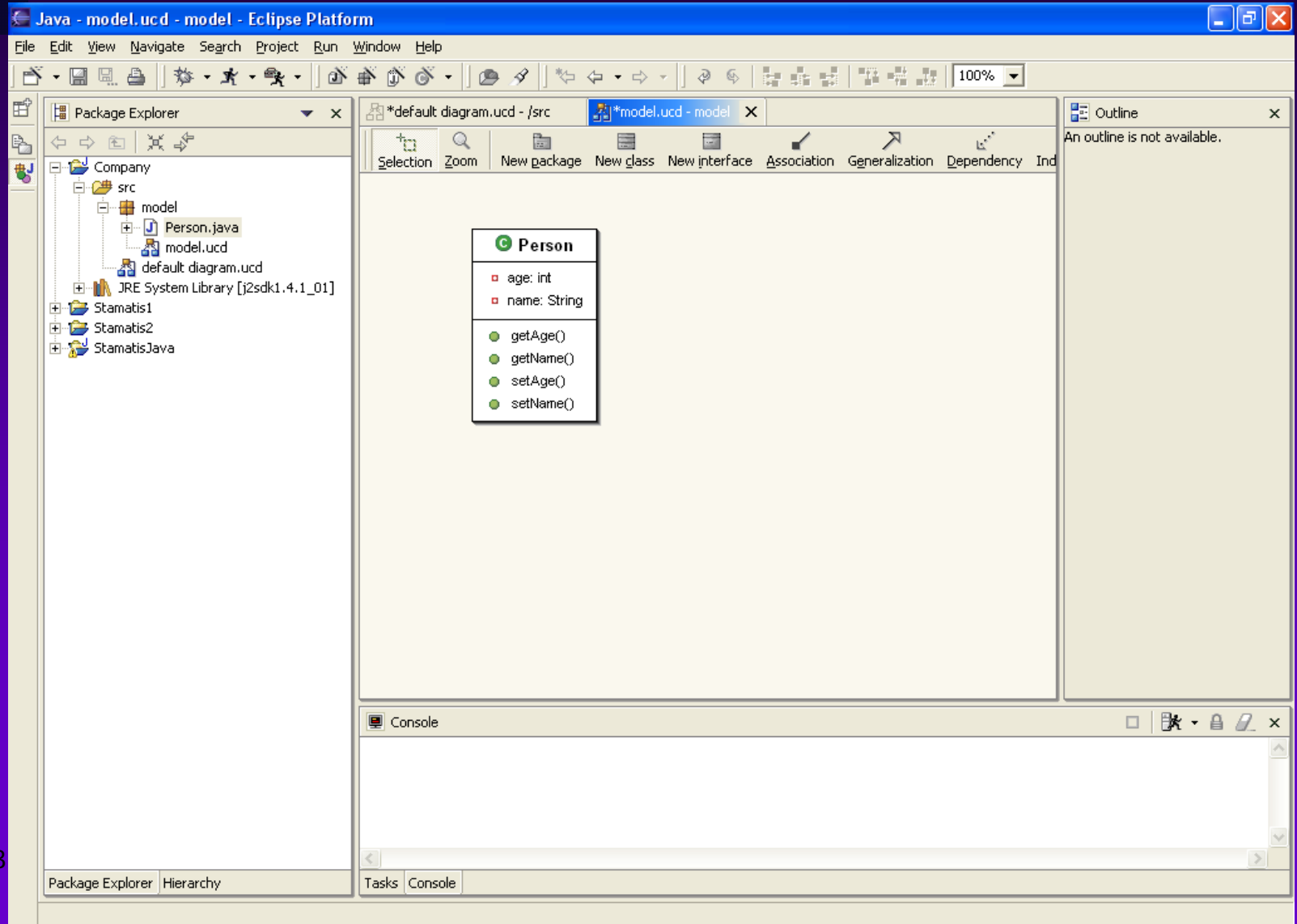


# Eclipse UML Class Diagrams

- Enter Attributes properties: "age: int" and "name: String"



# Eclipse UML Class Diagrams



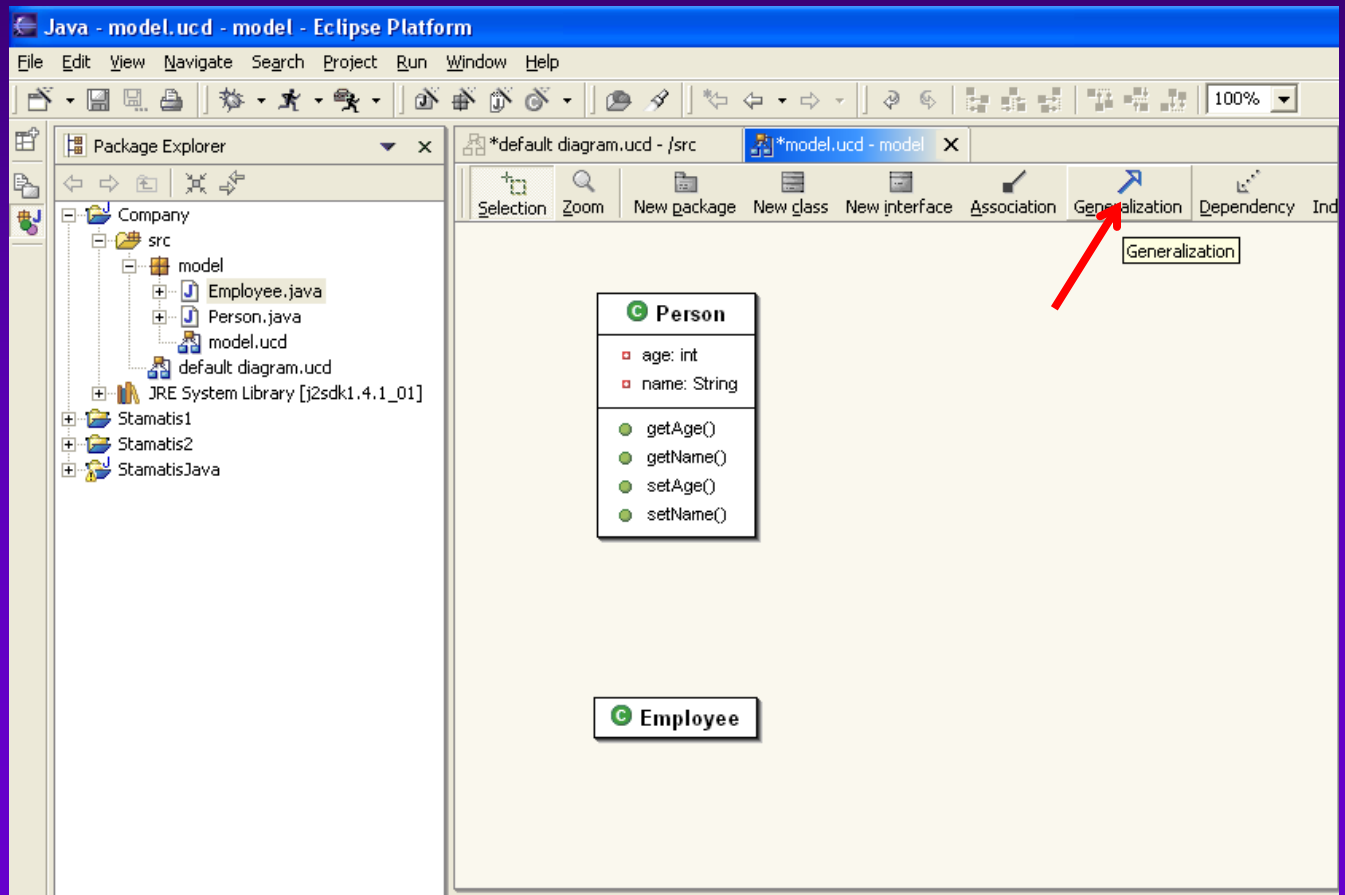
The screenshot shows the Eclipse IDE interface with a UML Class Diagram for a class named **Person**. The Package Explorer on the left shows a project structure with a 'model' package containing 'Person.java' and 'model.ucd'. The main editor displays the class diagram with the following details:

Person	
age: int	
name: String	
getAge()	
getName()	
setAge()	
setName()	

The Outline view on the right shows the message: "An outline is not available." The Console view at the bottom is empty.

# Eclipse UML Class Diagrams

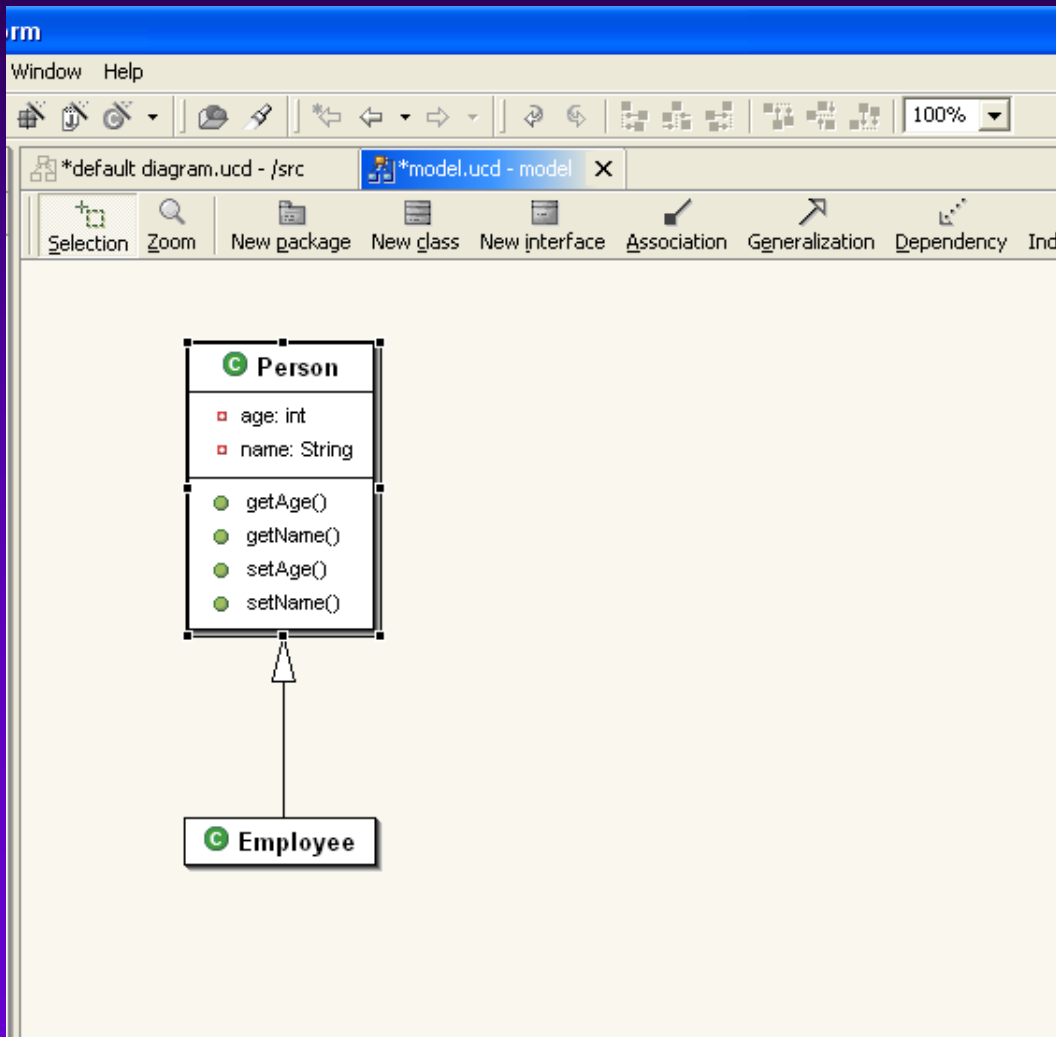
- Create a New class named Employee.
- We would like to add inheritance from Employee to Person.
- Select Generalization in the Toolbar





# Eclipse UML Class Diagrams

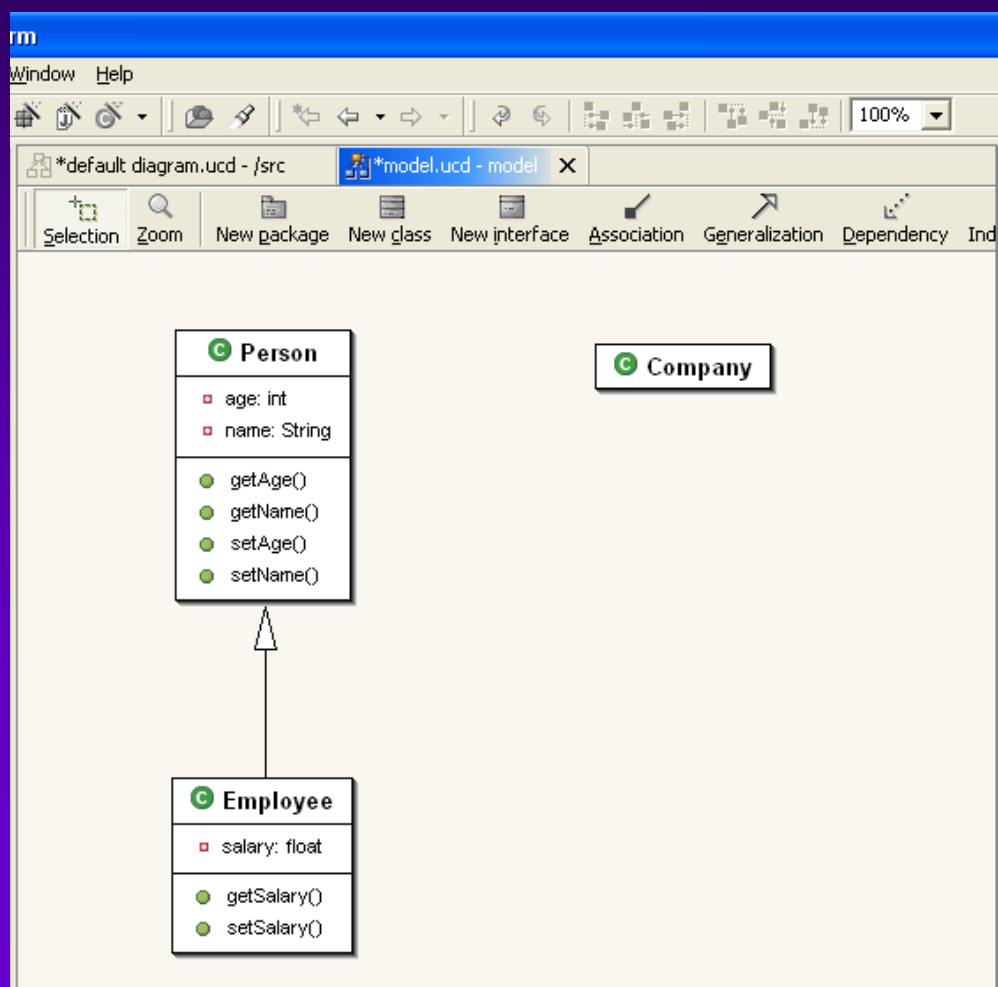
Drag the Inheritance from Employee to Person.





# Eclipse UML Class Diagrams

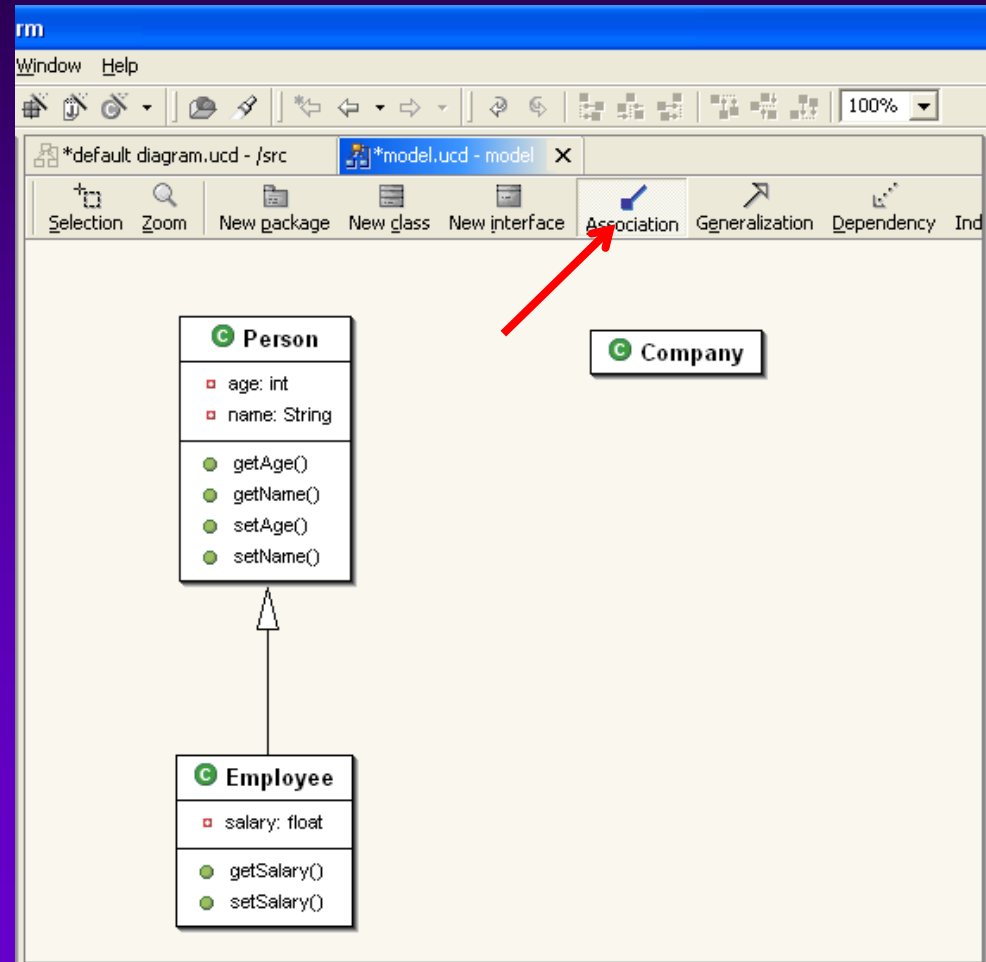
- Add "salary: float" attribute to Employee
- Create a New class named Company.



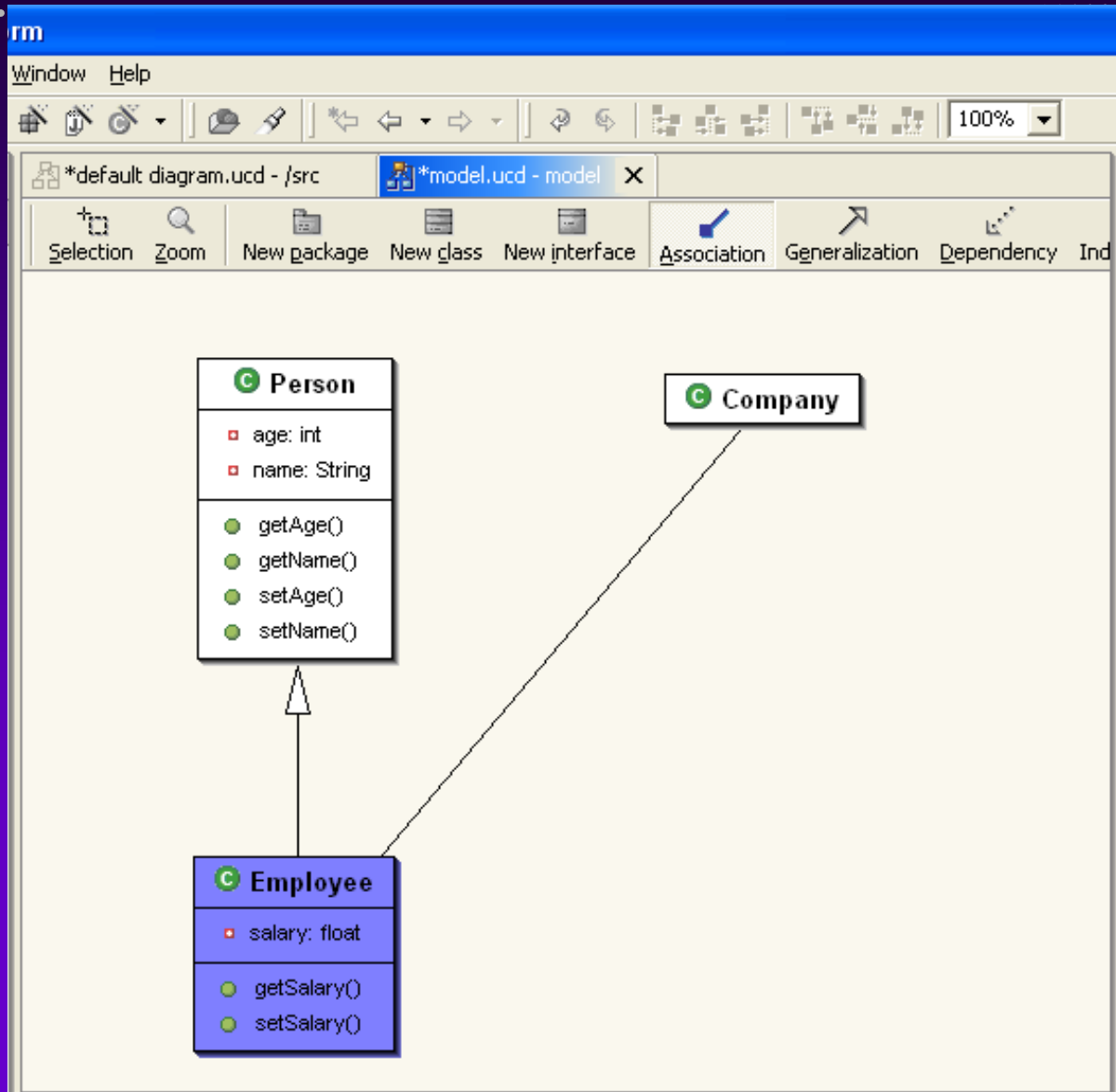


# Eclipse UML Class Diagrams

- We want to create a new Association between Company and Employee.
- Select Association in the Toolbar then drag from Company to Employee.



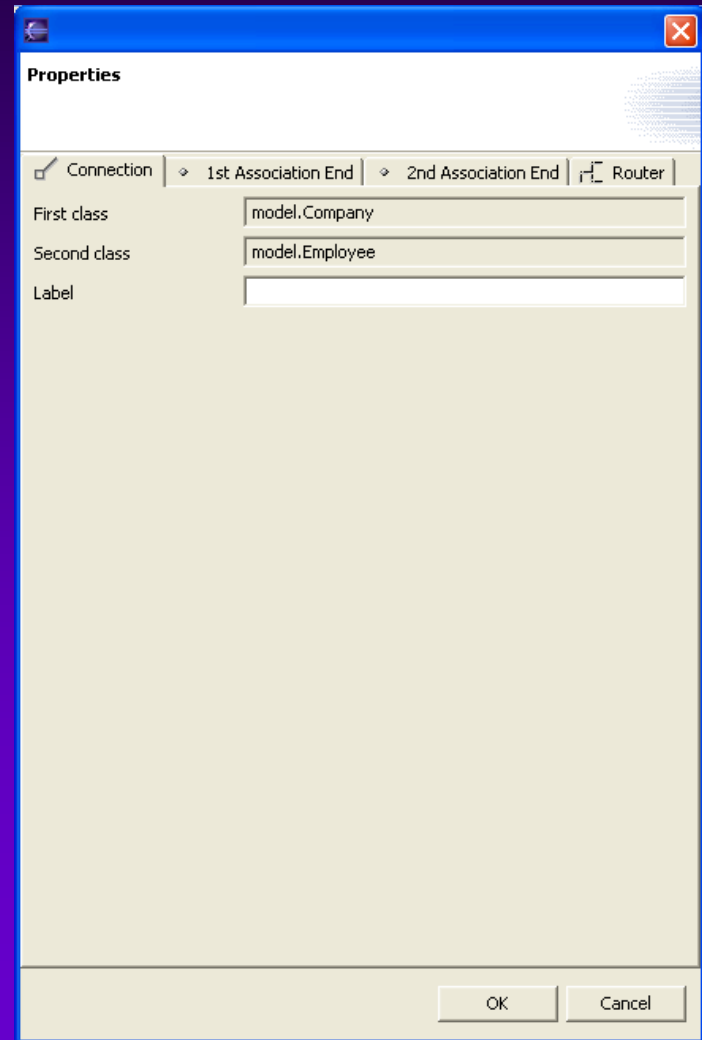
# Eclipse UML Class Diagrams





# Eclipse UML Class Diagrams

It is possible to add a name to the Label, but we will not in this example.





# Eclipse UML Class Diagrams

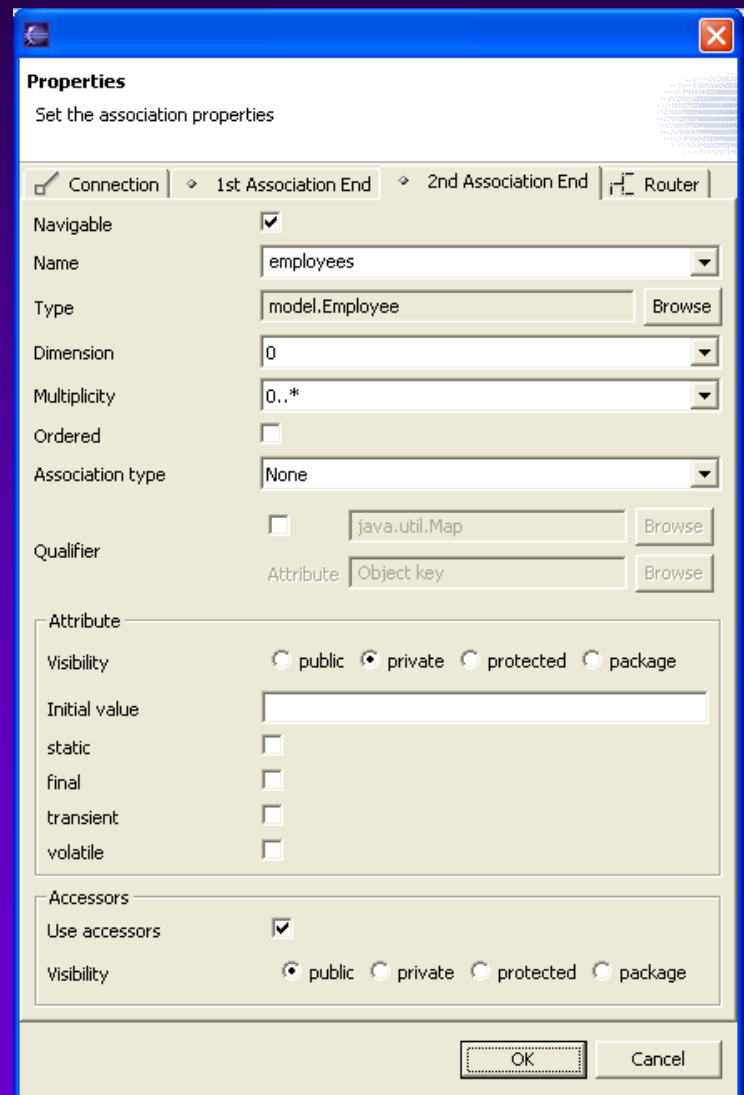
- Select the "1<sup>st</sup> Association End" tab
- Set multiplicity to 1

The screenshot shows the 'Properties' dialog box in Eclipse, specifically for an association end. The '1st Association End' tab is active. The 'Multiplicity' is set to 1. Other settings include Name: company, Type: model.Company, Dimension: 0, Association type: None, and Visibility: private. The 'Use accessors' checkbox is checked, and its visibility is set to public.



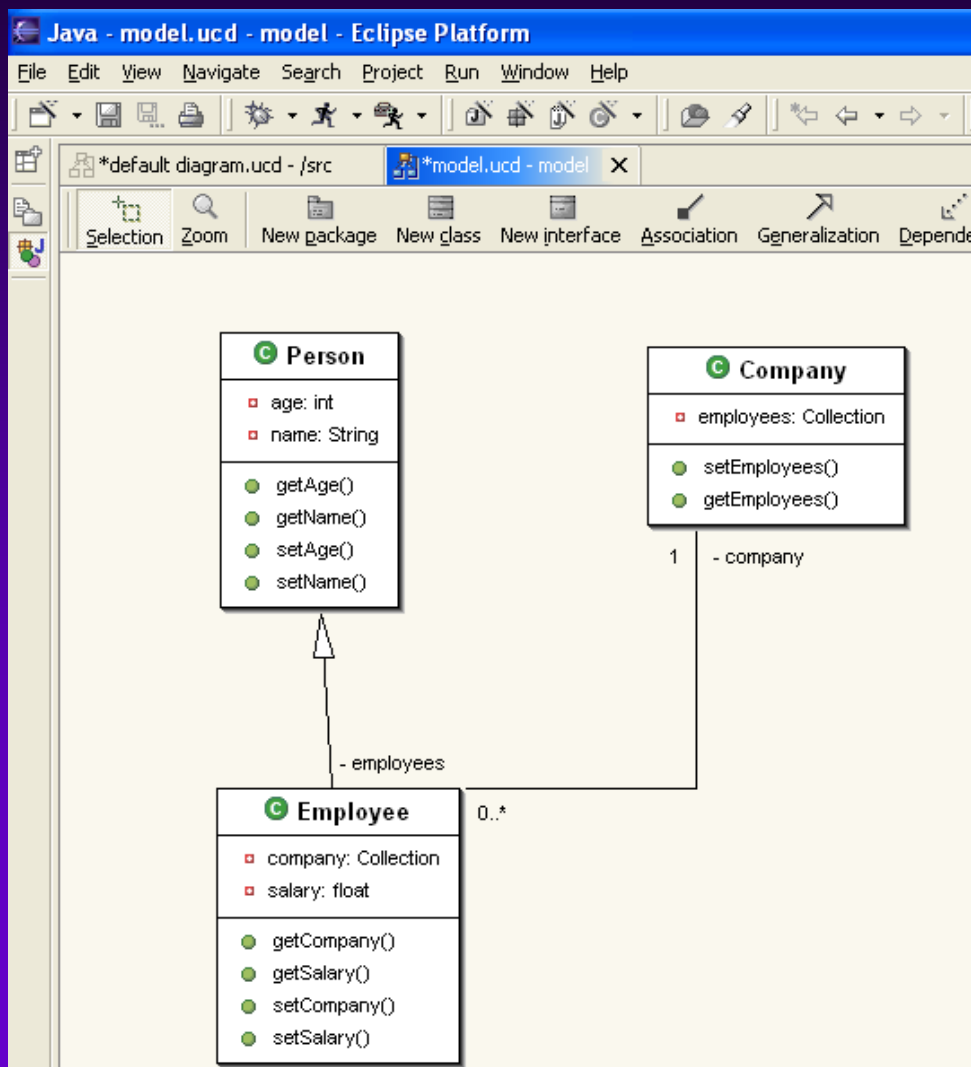
# Eclipse UML Class Diagrams

- Select the "2<sup>nd</sup> Association End" tab
- Set name "employees"
- Set multiplicity "0..\*"





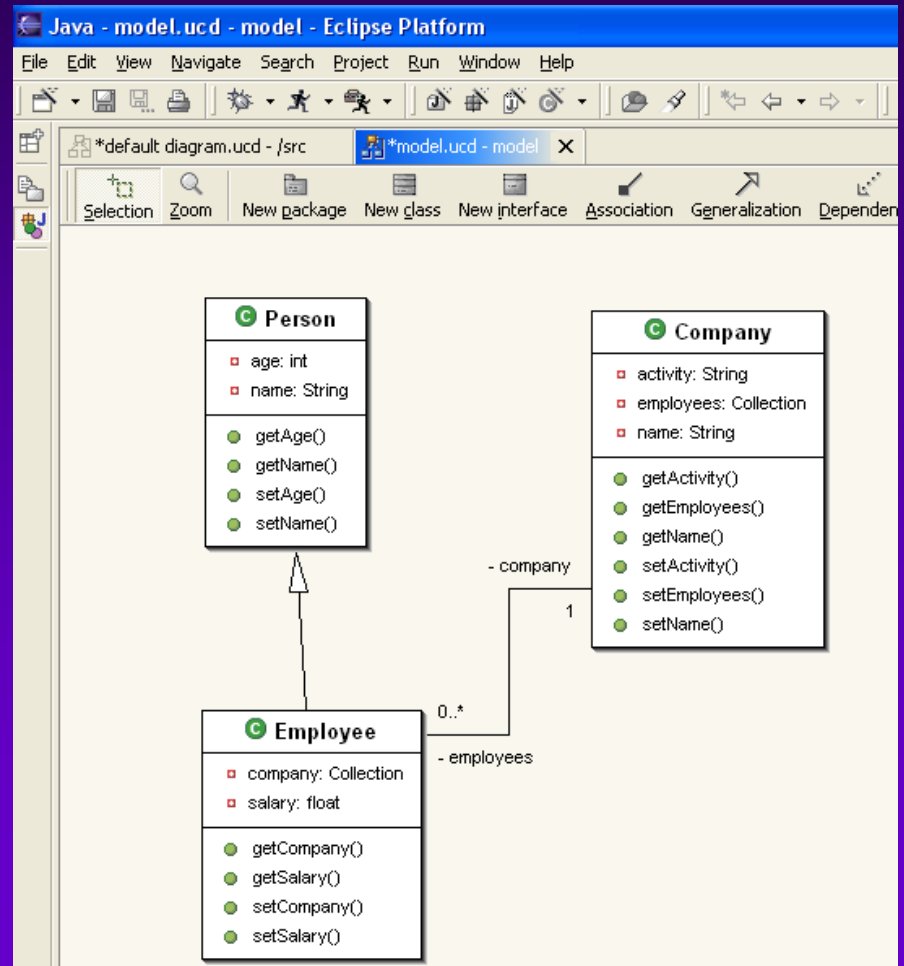
# Eclipse UML Class Diagrams





# Eclipse UML Class Diagrams

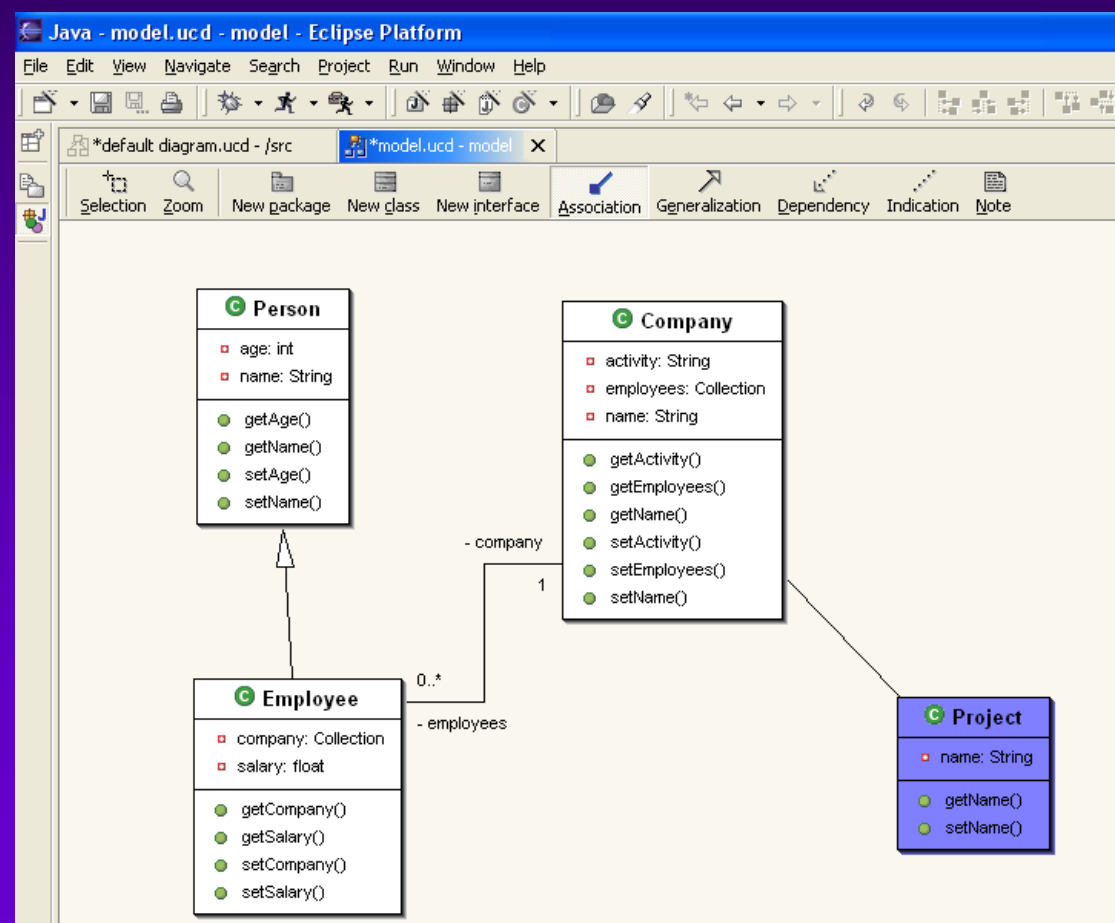
- Click on Company and create two new Attributes: "name: String" and "activity: String"





# Eclipse UML Class Diagrams

- Create a New class named Project
- Add attribute "name:String: to class Project
- Add an association between Company and Project.



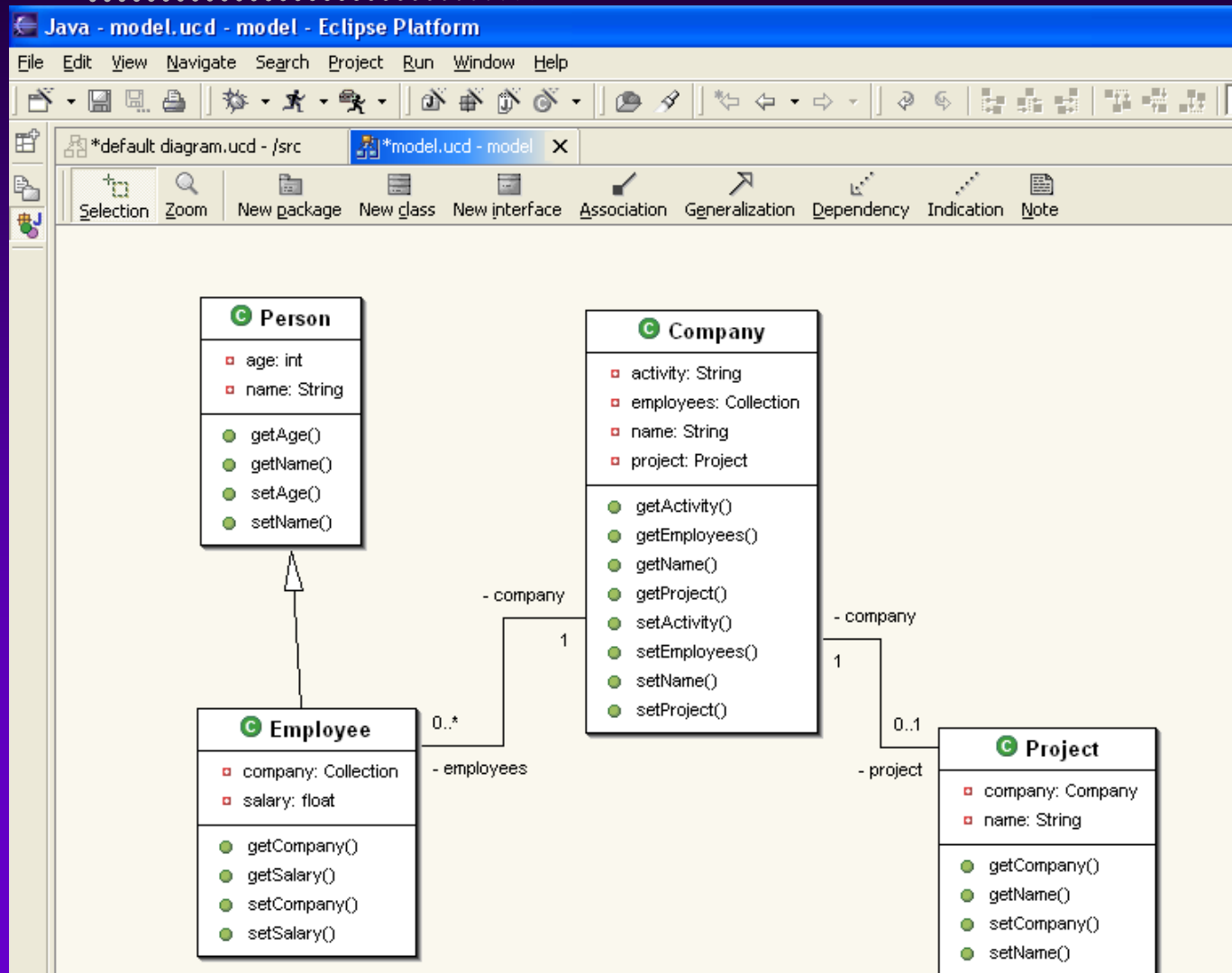


# Eclipse UML Class Diagrams

---

- In association properties
  - Label: leave blank
  - 1<sup>st</sup> associations end:
    - Name: company
    - Multiplicity: 1
  - 2<sup>nd</sup> associations end:
    - Name: project
    - Multiplicity: 0..1

# Eclipse UML Class Diagrams





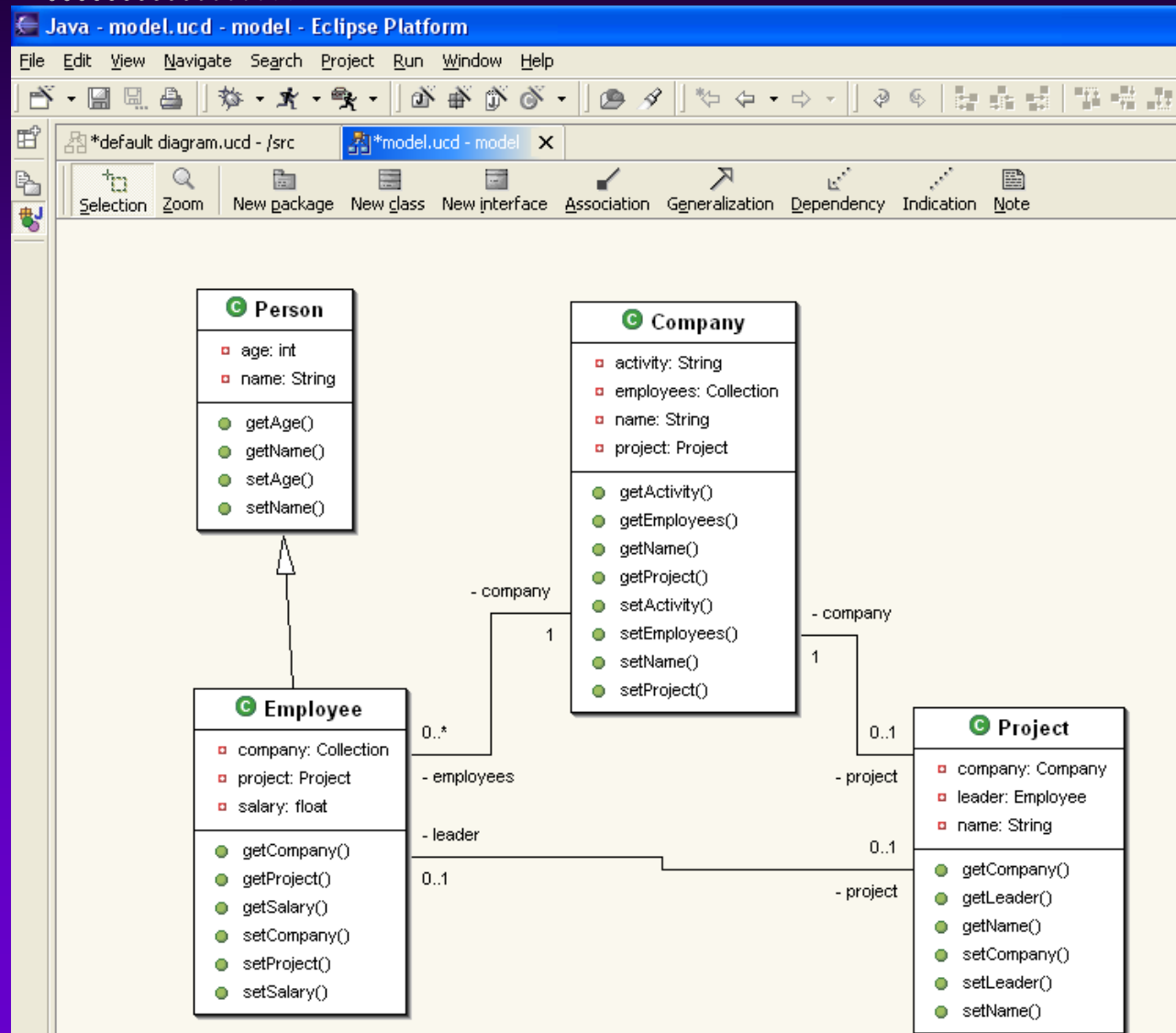
# Eclipse UML Class Diagrams

---

- Add an Association between Employee and Project
- Select Association and drag from Employee to Project
- In the properties menu:
  - Label: leave blank
  - 1st Association End: Name: leader, Dimension 0, Multiplicity 0..1
  - 2nd Association End: Name: Project, Dimension 0, Multiplicity 0..1



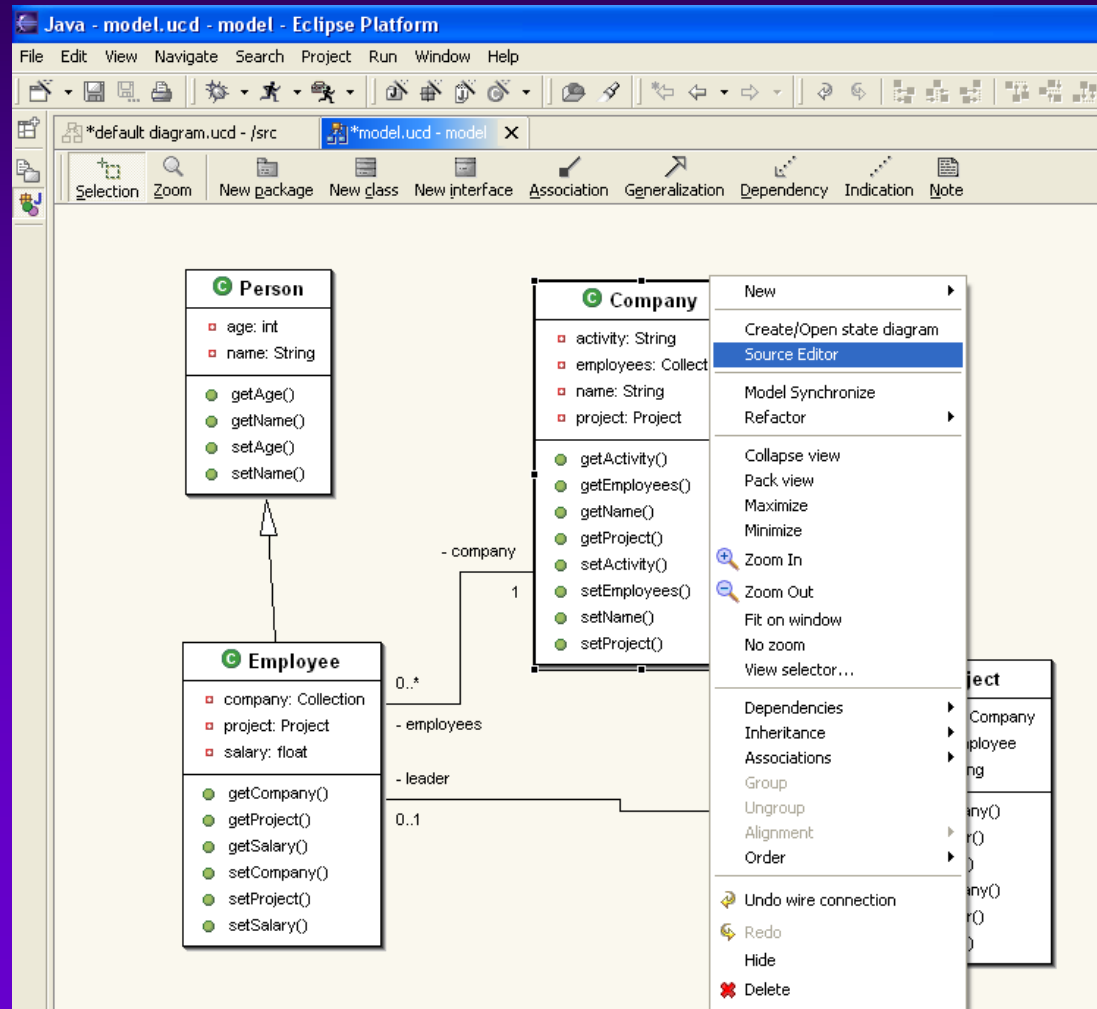
# Eclipse UML Class Diagrams





# Show code

- Select a Class
- From the pop-up menu select "Source editor"

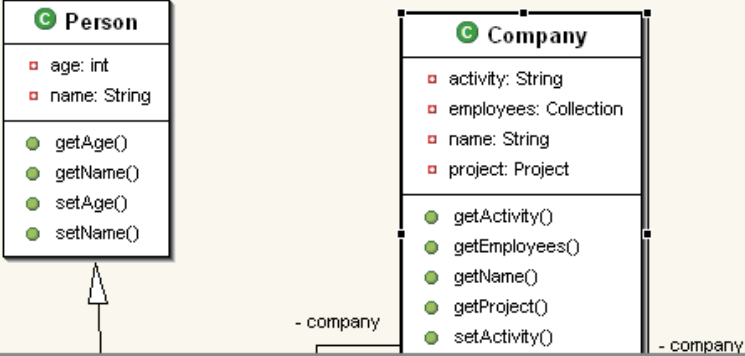


Java - Company.java - Eclipse Platform

File Edit Source Refactor Navigate Search Project Run Window Help

\*default diagram.ucd - /src    \*model.ucd - model X

Selection Zoom New package New class New interface Association Generalization Dependency Indication Note



```

classDiagram
    class Person {
        +age: int
        +name: String
        +getAge()
        +getName()
        +setAge()
        +setName()
    }
    class Company {
        +activity: String
        +employees: Collection
        +name: String
        +project: Project
        +getActivity()
        +getEmployees()
        +getName()
        +getProject()
        +setActivity()
    }
    Person <|-- Company
    Company --> Person : - company
    Company --> Person : - company
  
```

Company.java X

```

/**
 * @author karvoun
 *
 * To change the template for this generated type comment go to
 * Window>Preferences>Java>Code Generation>Code and Comments
 */
public class Company {

    private Collection employees;

    private String name;

    private String activity;

    private Project project;

    public static void main(String[] args) {
  
```

Writable Insert 10 : 1



# The end

---

- Experiment!!!
- Eclipse Tutorials:
  - <http://www.eclipse.org/>
  - [http://www.3plus4software.de/eclipse/index\\_en.html](http://www.3plus4software.de/eclipse/index_en.html)
  - <http://www.omondo.com/>
  - <http://www.tutorial-omondo.com/jdo/classdiagram.html>