5. Eclipse user interface overview

Eclipse provides *Perspectives*, *Views* and *Editors*. *Views* and *Editors* are grouped into *Perspectives*.

5.1. Workspace

The *workspace* is the physical location (file path) you are working in. Your projects, source files, images and other artifacts can be stored and saved in your workspace. It is also possible to refer to external resources, e.g. projects, in the *workspace*.

You can choose the workspace during startup of Eclipse or via the menu (*File → Switch Workspace → Others*) .

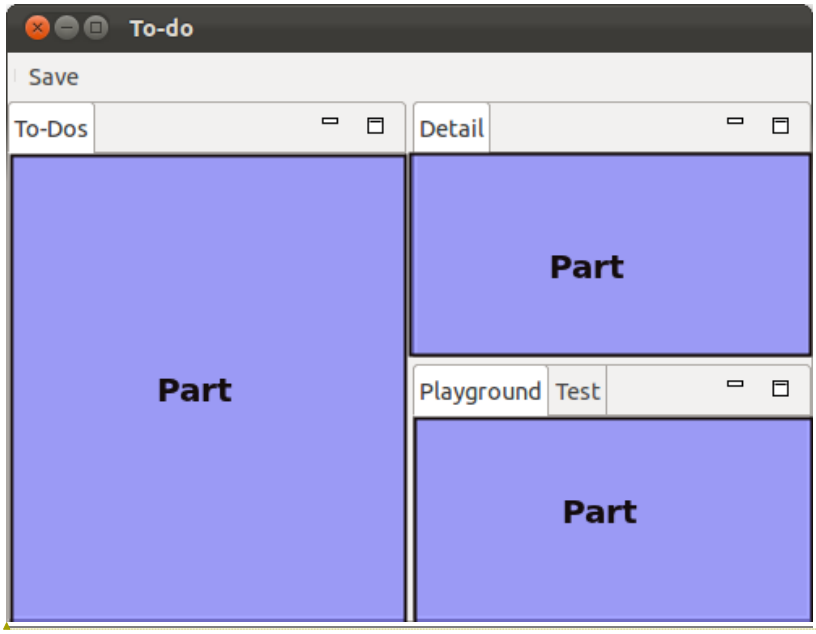
5.2. Eclipse projects

An Eclipse project contains source, configuration and binary files related to a certain task and groups them into buildable and reusable components. An Eclipse project can have *natures* assigned to it which describes the purpose of this project. For example the Java *nature* defines a project as Java project.

Natures for a project are defined via the `.project` file in the project directory.

5.3. Parts

Parts are user interface components which allow you to navigate and modify data. *Parts* are typically divided into *Views* and *Editors*.



Formatted: Font: (Default) Arial, 12 pt

The distinction into *Views* and *Editors* is primarily not based on technical differences, but on a different concept of using and arranging these *Parts*.

A *View* is typically used to work on a set of data, which might be a hierarchical structure. If data is changed via the *View*, this change is typically directly applied to the underlying data structure. A *View* sometimes allows us to open an *Editor* for a selected set of the data.

An example for a *View* is the *Java Package Explorer*, which allow you browse the files of Eclipse Projects. If you choose to change data in the Package Explorer, e.g. if you rename a file, the file name is directly changed on the file system.

Editors are typically used to modify a single data element, e.g. a file or a data object. To apply the changes made in an editor to the data structure, the user has to explicitly save the editor content.

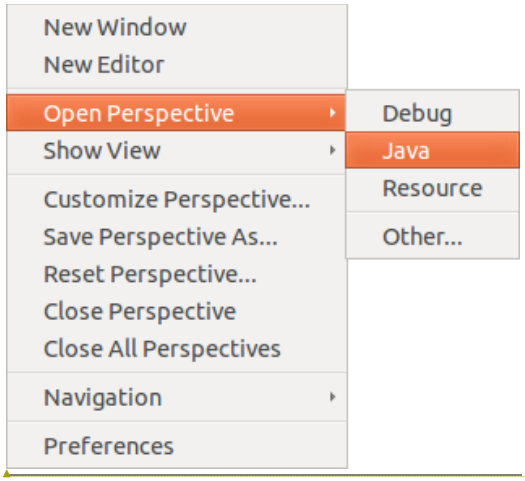
Editors were traditionally placed in a certain area, called the *editor area*. Until Eclipse 4 this was a hard limitation. Eclipse 4 applies no technical restrictions on *Editors*, they can be freely positioned.

For example the *Java Editor* is used to modify Java source files. Changes to the source file are applied once the user selects the *Save* command.

5.4. Perspective

A *Perspective* is a visual container for a set of *Parts*. The Eclipse IDE uses *Perspectives* to arrange *Parts* for different development tasks. You can switch *Perspectives* in your Eclipse IDE via the *Window → Open Perspective → Other*menu entry.

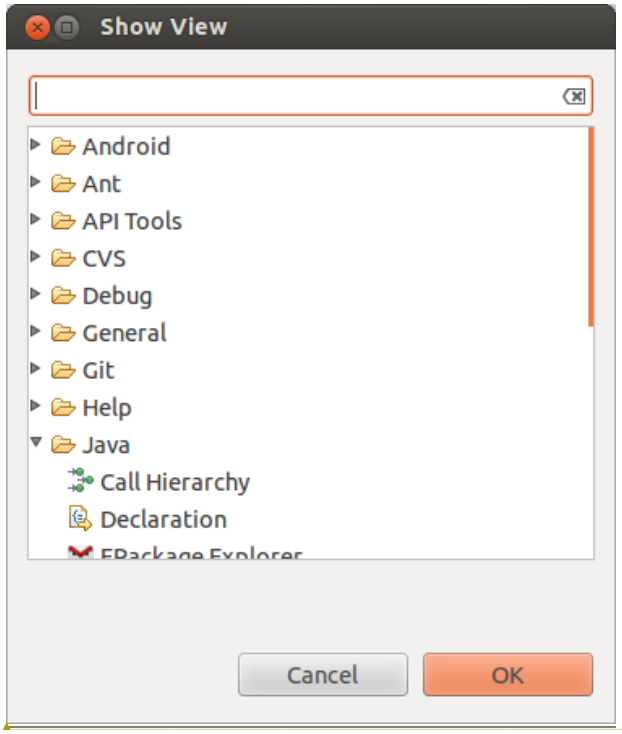
The main perspectives used in the Eclipse IDE are the *Java* perspective for Java development and the *Debug*perspective for debugging Java applications.



Formatted: Font: (Default) Arial, 12 pt

You can change the layout and content within a *Perspective* by opening or closing *Parts* and by re-arranging them.

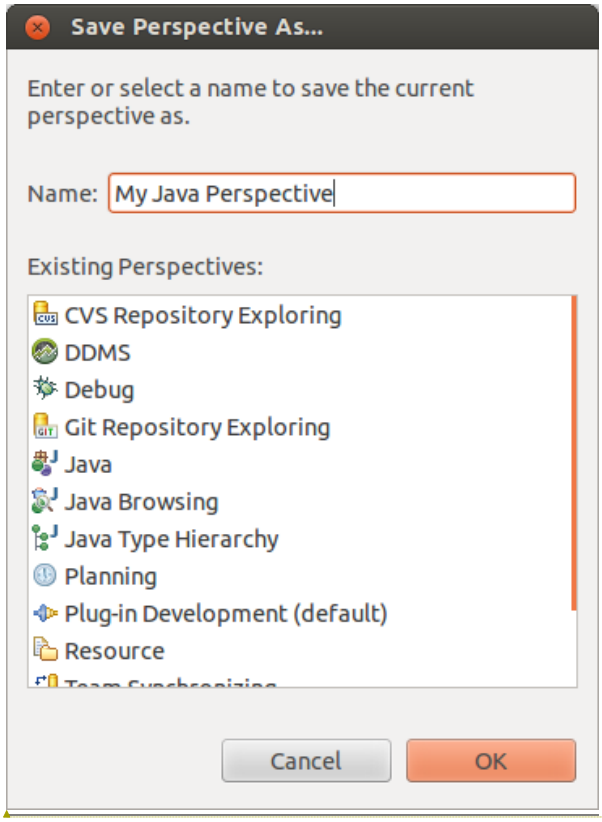
To open a new *Part* in your current *Perspective* use the *Window → Show View → Other* menu entry. The following *Show View* dialog allows you to search for certain *Parts*.



Formatted: Font: (Default) Arial, 12 pt

In cases you want to reset your current perspective to its default, you can use the *Window → Reset Perspective* menu entry.

You can save your *Perspective* via *Window → Save Perspective As....*

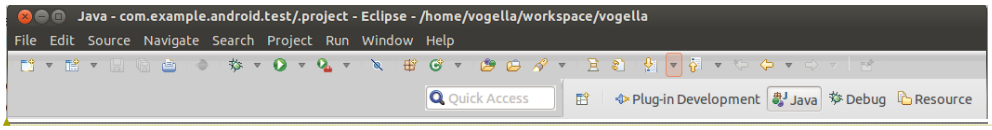


Formatted: Font: (Default) Arial, 12 pt

6. Eclipse Java perspective

6.1. Toolbar

The application toolbar contains actions which you typically perform, e.g. creating Java resources or running Java projects. It also allows to switch between opened perspectives.

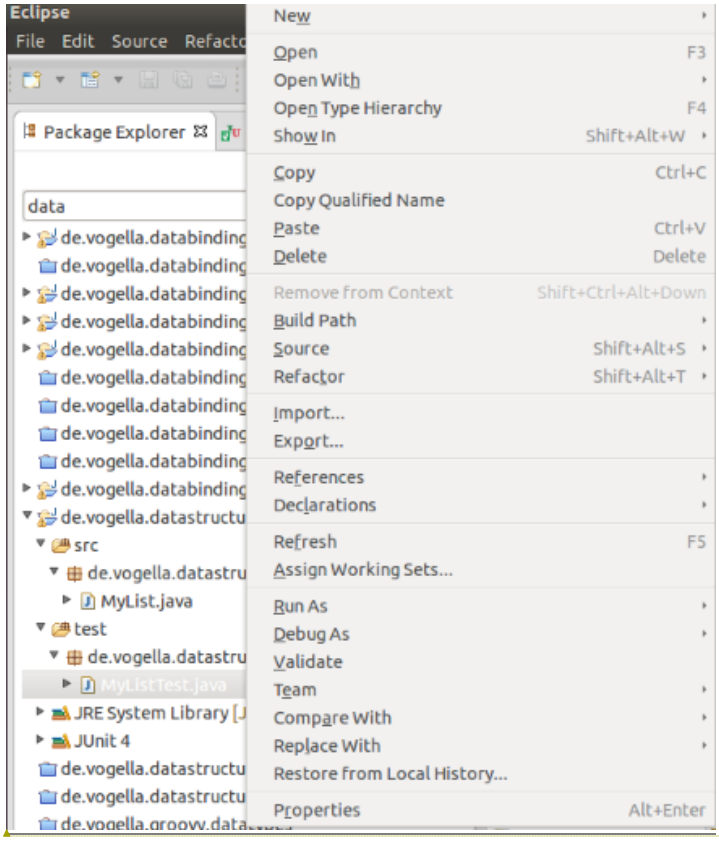


Formatted: Font: (Default) Arial, 12 pt

6.2. Useful Views

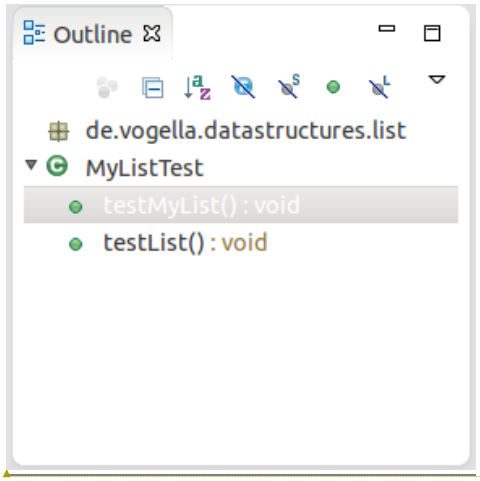
The Eclipse perspective contains useful *Views* which allow you to work with your Java project.

The *Package Explorer View* allows you to browser the structure your projects and to open files via double-click. It is also used to change the structure of your project. For example you can rename files or move files and folders via drag and drop. A right mouse click on a file or folder shows you the available options.



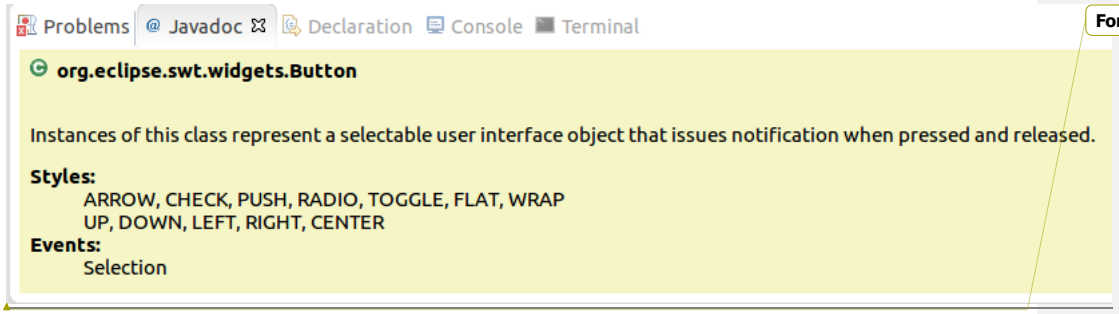
The *Outline View* shows the structure of the currently selected Java source file.

Formatted: Font: (Default) Arial, 12 pt



Formatted: Font: (Default) Arial, 12 pt

The *Javadoc View* shows the documentation of the selected element in the Java editor.

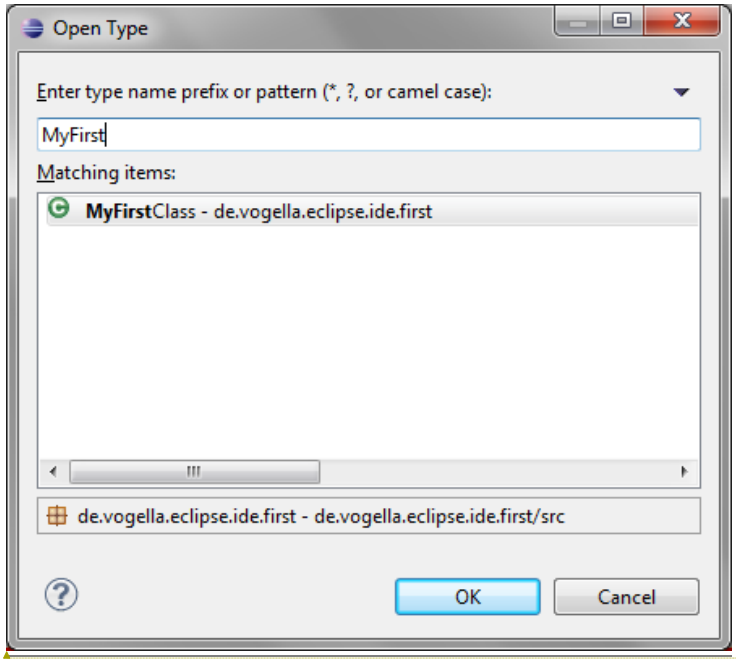


Formatted: Font: (Default) Arial, 12 pt

6.3. Opening a class

You can navigate between the classes in your project via the *Package Explorer View*.

In addition you can open any class via positioning the cursor on the class in an editor and pressing F3. Alternatively, you can press Ctrl+Shift+T. This will show a dialog in which you can enter the class name to open it.



Formatted: Font: (Default) Arial, 12 pt

7. Create your first Java program

The following describes how to create a minimal Java program using Eclipse. It is tradition in the programming world to create a small program which writes "Hello World" to the console. We will adapt this tradition and will write "Hello Eclipse!" to the console.

7.1. Create project

Select from the menu *File → New → Java project*.
Enter `de.vogella.eclipse.ide.first` as the project name. Select the *Create separate folders for sources and class files* flag.

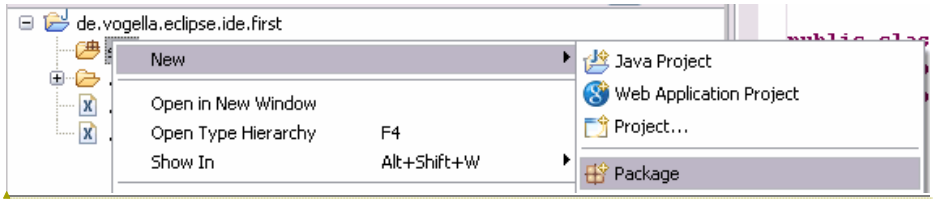
Formatted: Font: (Default) Arial, 12 pt

Press the *Finish* button to create the project. A new project is created and displayed as a folder. Open the`de.vogella.eclipse.ide.first` folder and explore the content of this folder.

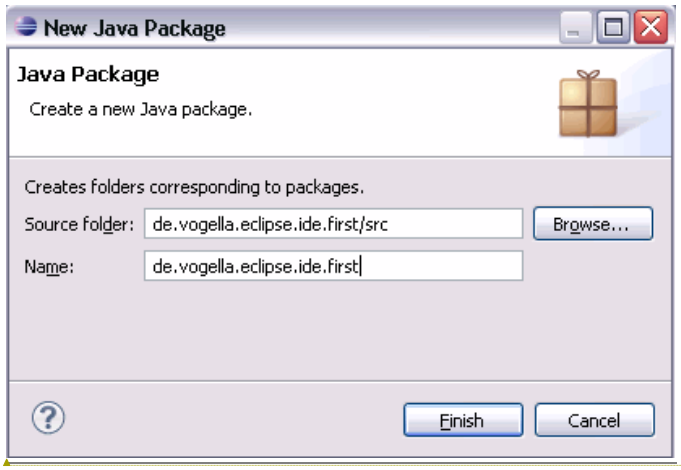
7.2. Create package

In the following step you will create a new `package`. A good convention is to use the same name for the top level package and the project.

To create the `de.vogella.eclipse.ide.first` package, select the folder `src`, right click on it and select *New →Package*.

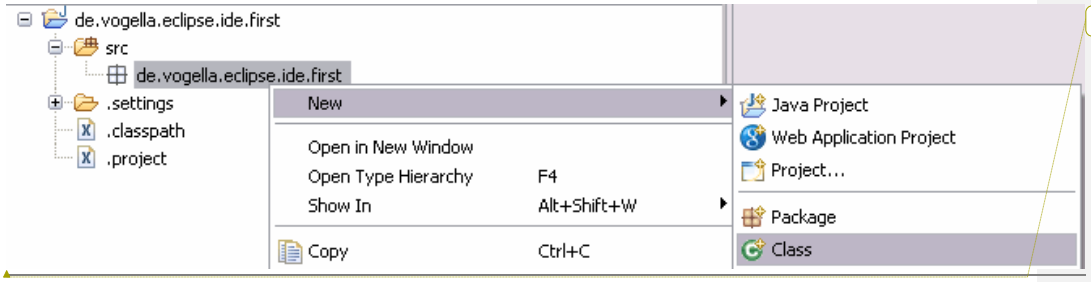


Enter the name of your new package in the dialog and press the *Finish* button.



7.3. Create Java class

Create a Java class. Right click on your package and select *New → Class*.

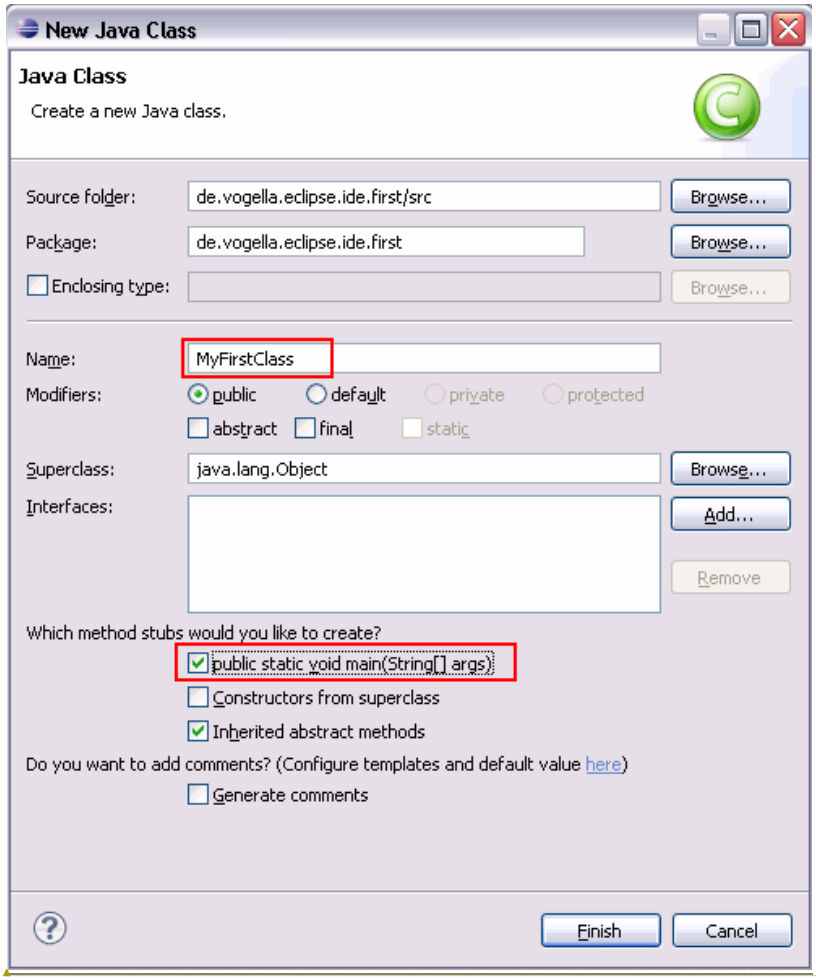


Enter `MyFirstClass` as the class name and select the *public static void main (String[] args)* flag.

Formatted: Font: (Default) Arial, 12 pt

Formatted: Font: (Default) Arial, 12 pt

Formatted: Font: (Default) Arial, 12 pt



Press the *Finish* button.

This creates a new file and opens the *Editor* for Java source files. Change the class to the following example.

```
package de.vogella.eclipse.ide.first;


public class MyFirstClass {
```

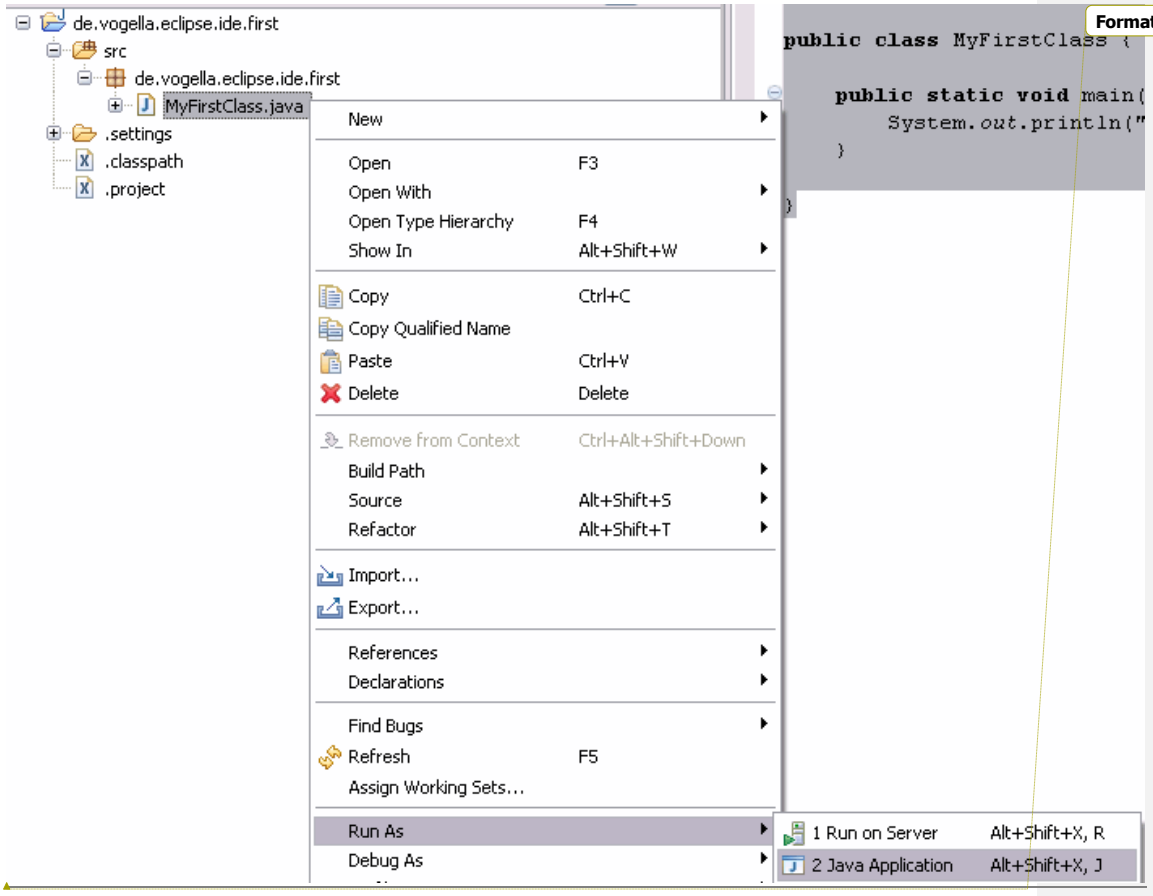
Formatted: Font: (Default) Arial, 12 pt

```
  public static void main(String[] args) {
    System.out.println("Hello Eclipse!");
  }

}
```

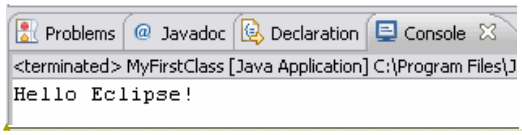
7.4. Run your project in Eclipse

Now run your code. Right click on your Java class and select *Run-as → Java application*.



Eclipse will run your Java program. You should see the output in the *Console View*.

Formatted: Font: (Default) Arial, 12 pt



Formatted: Font: (Default) Arial, 12 pt

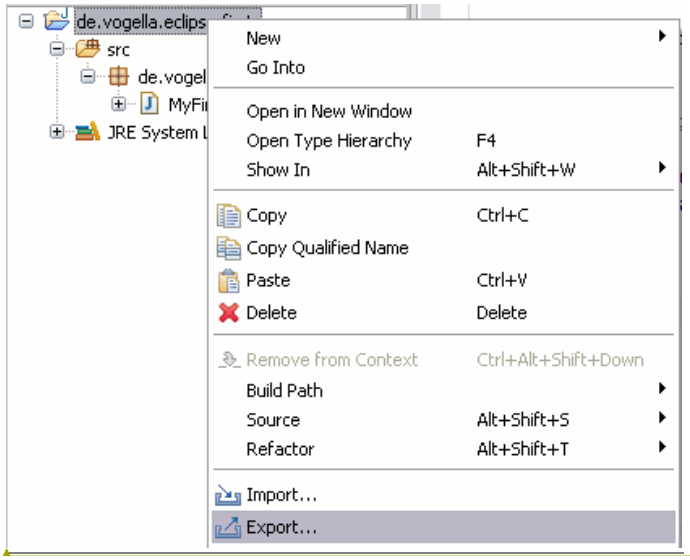
Congratulations! You created your first Java project, a package, a Java class and you ran this program inside Eclipse.

8. Run Java program outside Eclipse

8.1. Create jar file

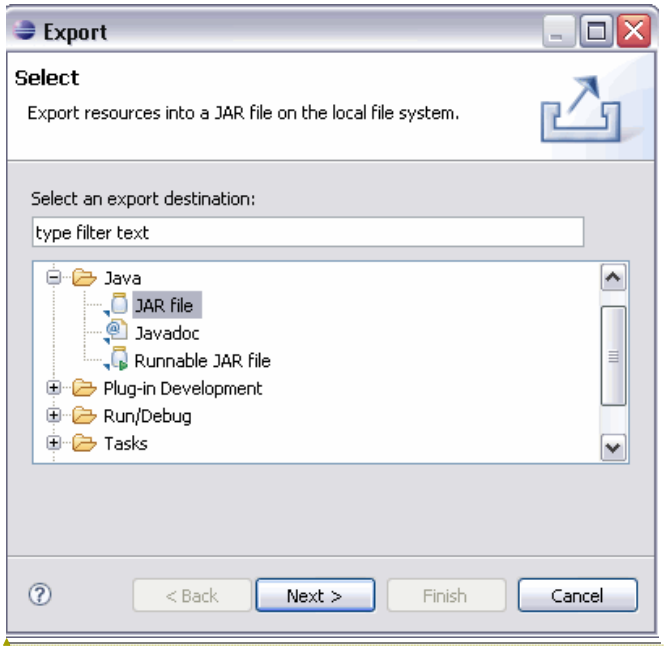
To run your Java program outside of Eclipse you need to export it as a `jar` file. A `jar` file is the standard distribution format for Java applications.

Select your project, right click on it and select `Export`.

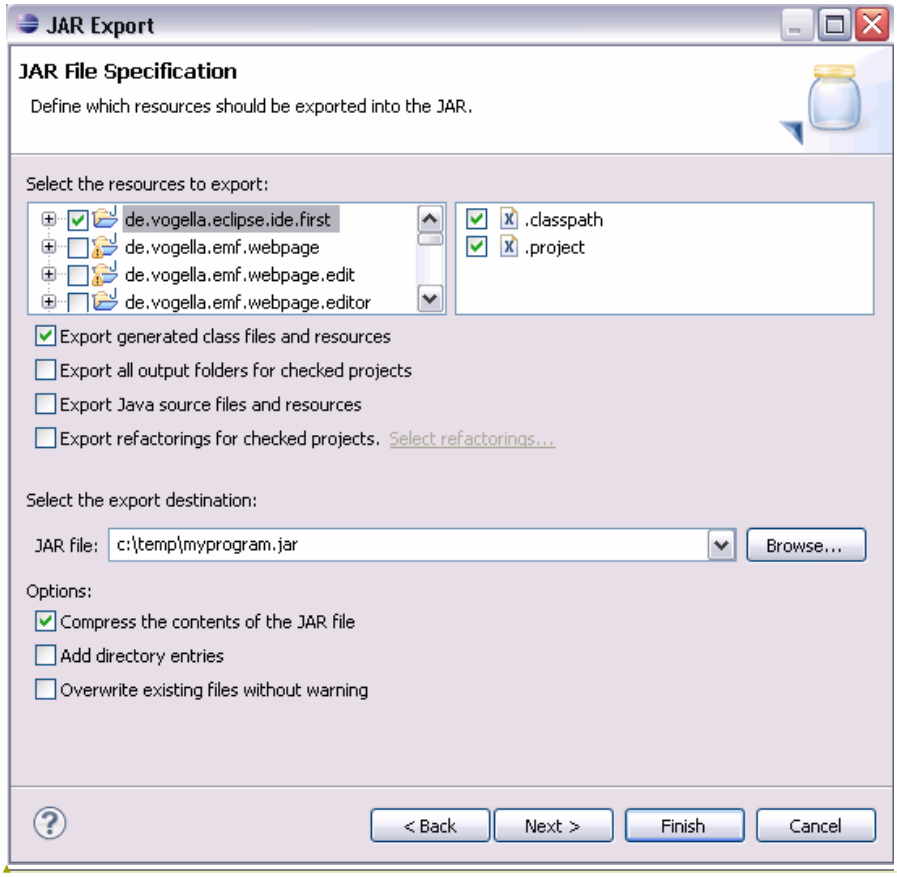


Formatted: Font: (Default) Arial, 12 pt

Select JAR file, select next. Select your project and maintain the export destination and a name for the jar file. I named it`myprogram.jar`.



Formatted: Font: (Default) Arial, 12 pt



Formatted: Font: (Default) Arial, 12 pt

Press finish. This creates a jar file in your selected output directory.

Congratulations! You created your first Java project, a package, a Java class and you ran this program inside Eclipse.

8.2. Run your program outside Eclipse

Open a command shell, e.g. under Microsoft Windows select *Start → Run* and type `cmd` and press enter. This should open a console.

Switch to your output directory, by typing `cd path`. For example if your jar is located in `c:\temp` type `cd c:\temp`.

firstjava_export To run this program you need to include the jar file in your `classpath`. The `classpath` defines what Java classes are available to the Java runtime. You can add a `jar` file to the classpath with the `-jar` option.

```
java -classpath myprogram.jar de.vogella.eclipse.ide.first.MyFirstClass
```

If you type the command from above and are in the correct directory you should see the "Hello Eclipse!" output on the console.

```
C:\temp>java -classpath myprogram.jar de.vogella.eclipse.ide.first.MyFirstClass
Hello Eclipse!
```

Formatted: Font: (Default) Arial, 12 pt