Универзитет „Св. Кирил и Методиј“ - Скопје
ФАКУЛТЕТ ЗА ИНФОРМАТИЧКИ НАУКИ И КОМПЈУТЕРСКО ИНЖЕНЕРСТВО

Напредно програмирање

Аудиториски вежби 1

Верзија 1.0, 20 Септември, 2016

Содржина

1. Што е Eclipse?

Eclipse претставува интегрирана околина за развој (IDE) за програмскиот јазик Java. Денес претставува водечка околина за развој за Java со опфатен дел од пазарот од приближно 65%.

Eclipse е создаден од Open Source заедницата и се користи во повеќе различни области, пр. како развојна околина за Java или Android апликации. Развојот на Eclipse датира од 2001.

Eclipse Open Source заедницата има преку 200 Open Source проекти во повеќе различни аспекти од развојот на софтвер.

Сите Eclipse проекти ги води *Eclipse Foundation*. Тоа е не профитна организација, поддржана од своите членови со цел да хостира Eclipse Open Source проекти и да помага во созревањето на Open Source заедницата и како комплементарен екосистем на производи и сервиси.

Eclipse IDE може да се прошири со додатни софтверски компоненти кои се нарекуваат *plug-ins*. Притоа постојат повеќе Open Source проекти од различни компании кои го имаат проширено Eclipse IDE.

Eclipse може да се користи и како основа за креирање на апликации со помош на Eclipse Rich Client Platform (*Eclipse RCP*) за апликации.

2. Eclipse Public License

Eclipse Public License (EPL) е Open Source софтверска лиценца која ја користи *Eclipse Foundation* за нејзиниот софтвер. EPL е дизајнирана да биде соодветна за бизнисите со тоа што EPL лиценцираните програми може да се користат, модификуваат, копираат и дистрибуираат слободно и без да се плаќа.

3. Инсталација на Eclipse

3.1. Java побарување на Eclipse

Eclipse има потреба од инсталирана Java Runtime околина, односно минмум Java 5 за да се извршува. Притоа препорачливо е користење на Java верзија 6 или повисока.

Eclipse IDE содржи сопствен Java компајлер. За компајлирање изворен код надвор од Eclipse потребни се **Java Development Tools**.

3.2. Инсталација на Java

Java можеби е веќе инсталирана на вашата машина. Ова може да се провери со отворање на конзола (ако сте на Windows: `Win+R`, внесете `cmd` и притиснете `Enter`) и впишување на следната команда:

```
java -version
```

Ако Java е соодветно инсталирана, треба да видите информации за тоа. Ако командната линија врати резултат дека програмата не може да се најде, треба да инсталирате Java.

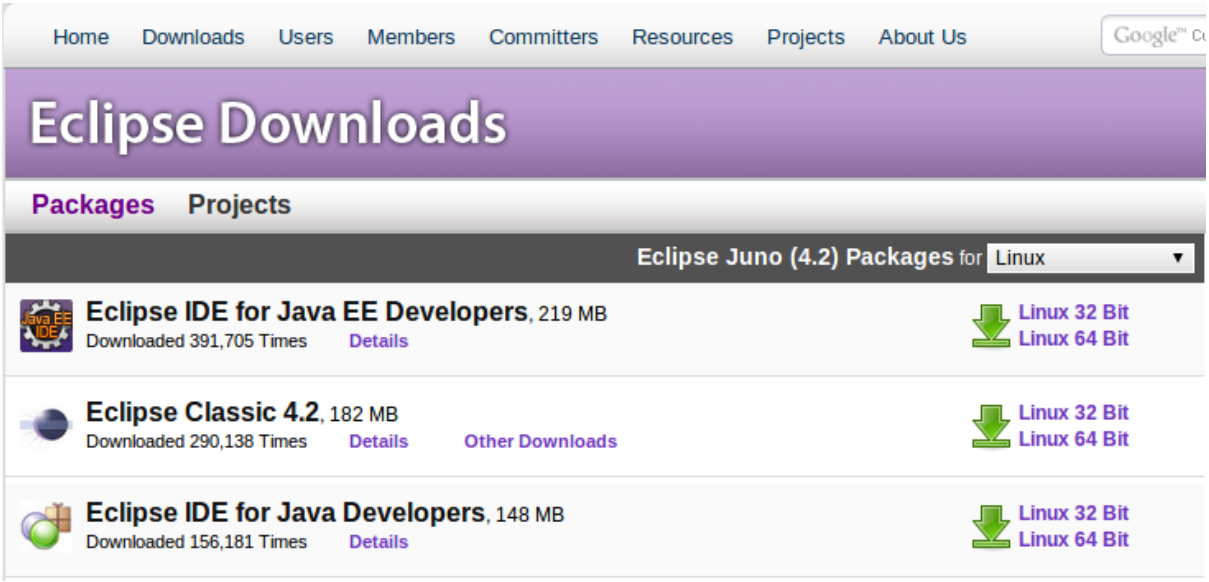
Google пребарување за "How to install JDK on YOUR_OS" треба да врати резултати со линкови со помош. Заменете го `YOUR_OS` со вашиот оперативен систем, пр. Windows, Ubuntu, Mac OS X, итн.

3.3. Download Eclipse

www.eclipse.org веб сајтот содржи запакувани инсталации на Eclipse дистрибуции.

Симнете го **Eclipse IDE for Java Developers** пакетот од следното URL: `http://www.eclipse.org/downloads`

На следните слики е прикажан сајтот на Eclipse за симнување на за на Linux систем.



Слика 1. Eclipse Download страница

Содржината на симнувањето е `.zip` датотека.

3.4. Инсталација на Eclipse

Откако ќе ја симнете `.zip` датотеката која ја содржи Eclipse дистрибуцијата единствено треба да ја отпакувате во посакуваниот локален директориум.

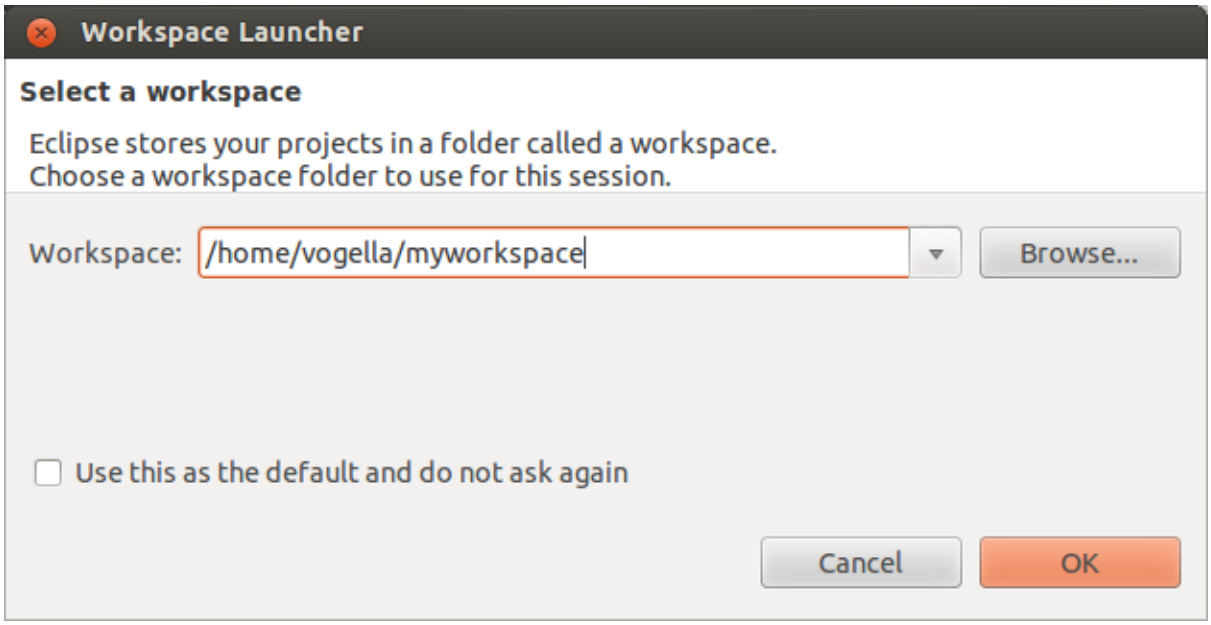
Употребете директориум во чија патека нема празни места, затоа што Eclipse понекогаш има проблем со тоа.

По отпакувањето може да го користите Eclipse. Нема потреба од дополнителни инсталации.

4. Работа со Eclipse

4.1. Стартување Eclipse

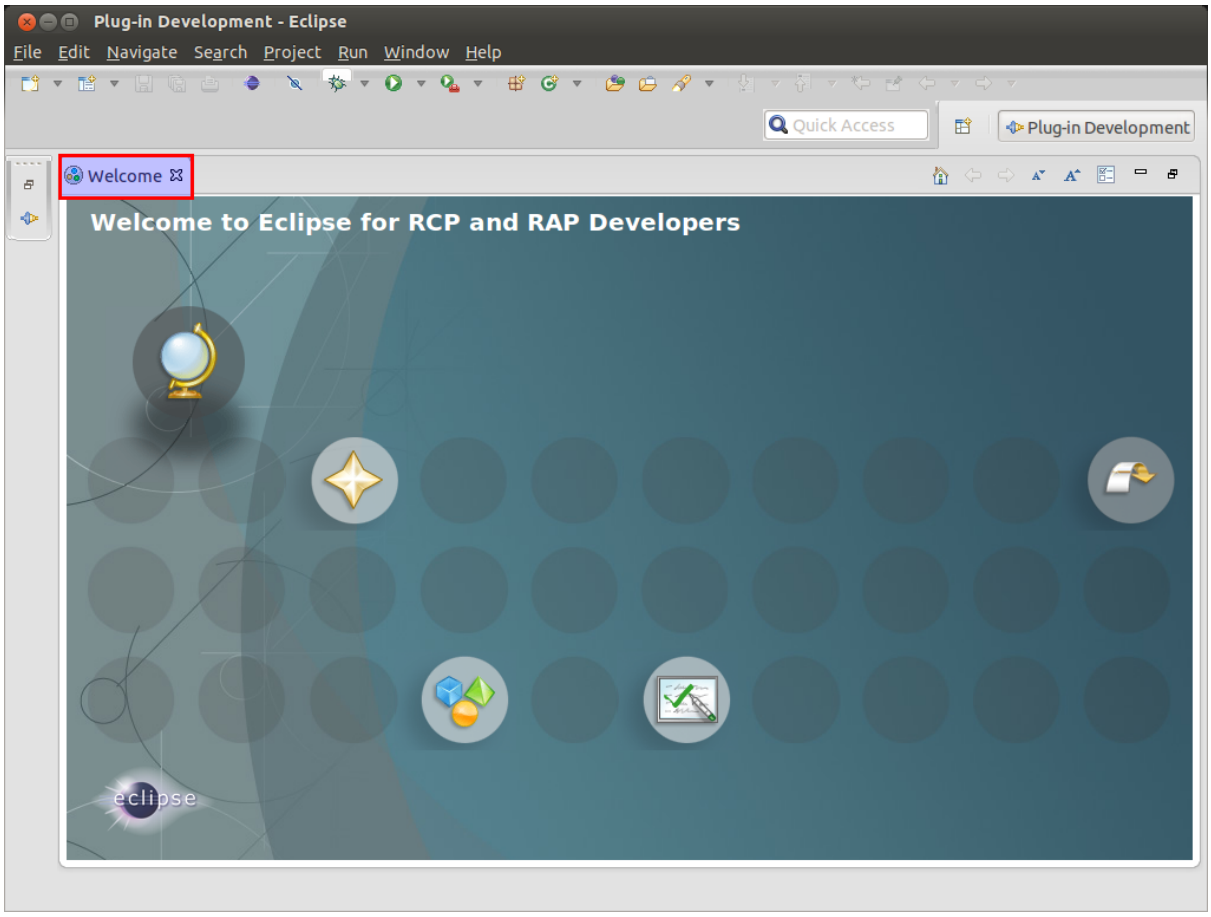
Стартувајте го Eclipse со двоен-клик на датотеката `eclipse.exe` (Microsoft Windows) или `eclipse` (Linux / Mac) во директориумот кој го отпакувавте Eclipse.



Слика 2. Избор на Workspace

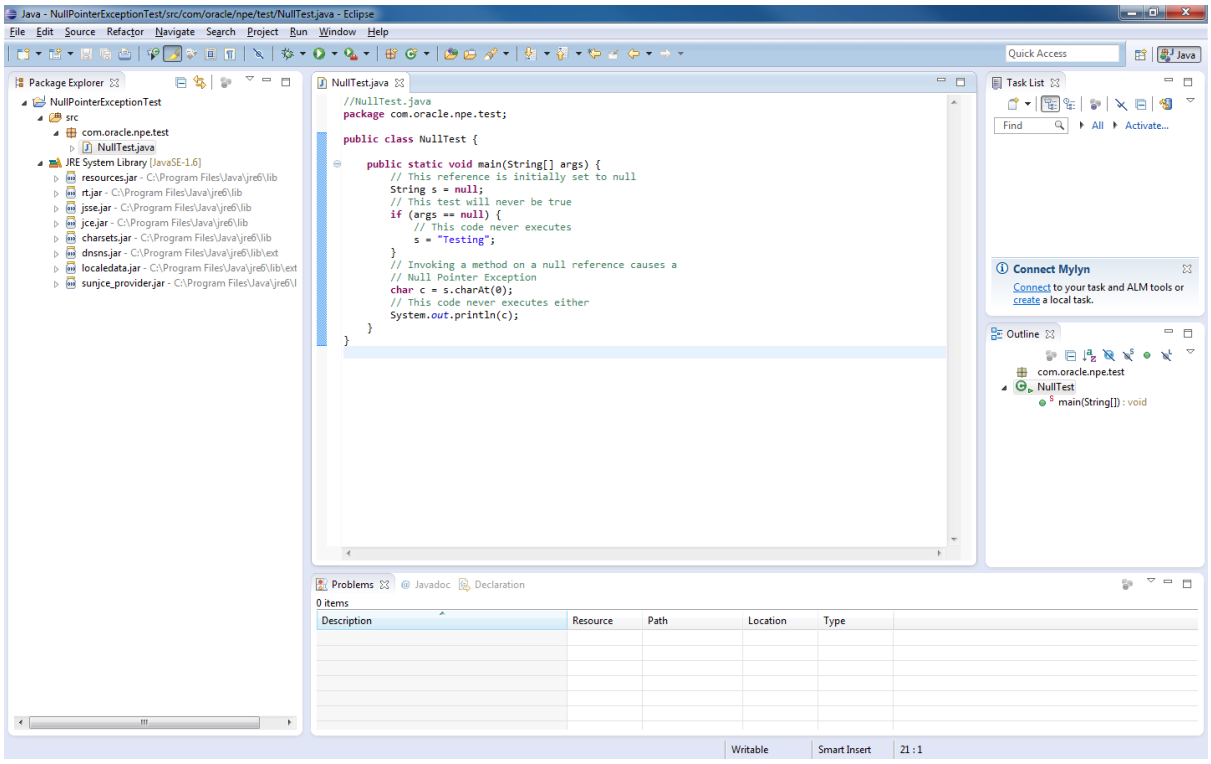
Системот ќе побара да изберете `workspace` (Избор на Workspace). `Workspace` е местото каде што ќе работите. Изберете празен директориум и притиснете на копчето OK.

Eclipse ќе се стартува и ќе прикаже Welcome страница. Затворете ја оваа страница.



Слика 3. Затворање на Eclipse welcome screen

Откако ќе го затворите почетниот екран треба да видите екран сличен на следниот:



Слика 4. Почетен Eclipse поглед

5. Преглед на Eclipse корисничкиот интерфејс

Eclipse е составен од **Perspectives**, **Views** и **Editors**. **Views** и **Editors** се групирани во **Perspectives**.

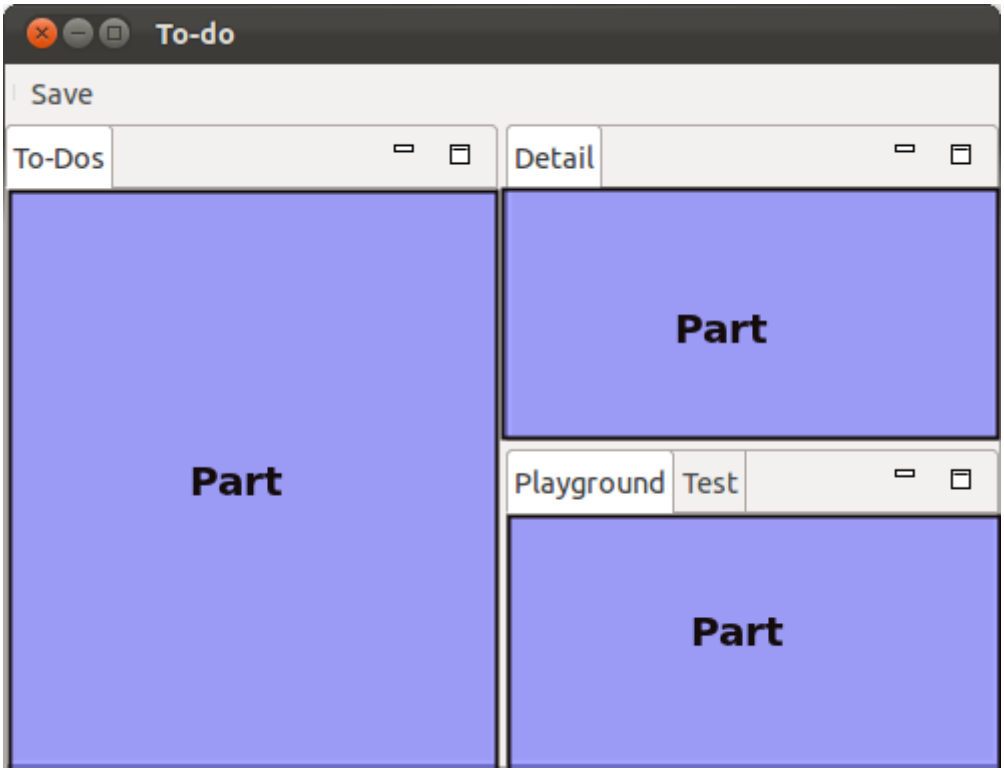
5.1. Workspace

Workspace е физичката локација (патеката на датотеките) со кои работите. Вашите проекти, изворни датотеки, слики и други артефакти може да се чуваат во вашиот работен простор, но исто така може да референцирате и надворешни ресурси (пр. проекти).

Може да изберете работен простор на стартување на Eclipse или преку мени (`File → Switch Workspace → Others`).

5.2. Parts

Parts се компоненти од корисничкиот интерфејс кои овозможуваат да навигирате и модификувате податоци. Вообичаено поделени во **Views** и **Editors**.



Слика 5. Eclipse апликација со неколку делови

Раздвојувањето на **Views** и **Editors** примарно не е базирано на технички разлики, туку на различни концепти на користење и нивно уредување.

View вообичаено се користи за работа со податоци, кои може да се во хиерархиска структура. Ако податоците се променат преку **View**, оваа промена вообичаено директно се применува на податочната структура под неа. **View** понекогаш овозможува да се отвори **Editor** за избрано множество податоци.

Пример за **View** е **Java Package Explorer**, кој овозможува да се прелистуваат датотеките во Eclipse проектите. Ако промените податоци во Package Explorer, на пр. промените име на датотека, ова име директно се менува и во податочниот систем.

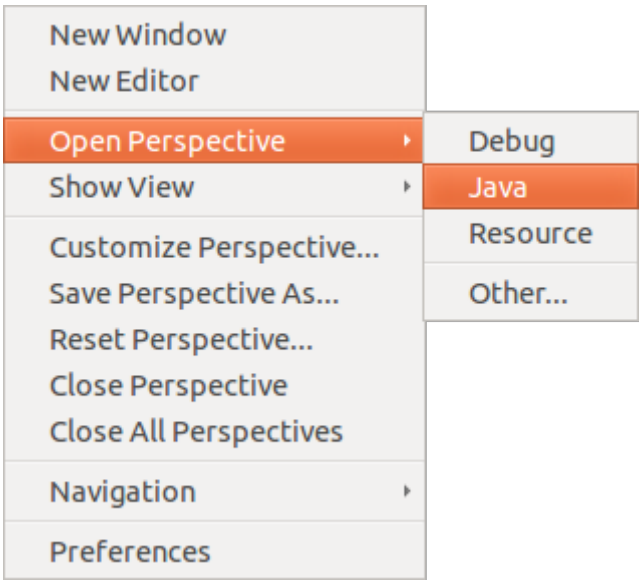
Editors вообичаено се употребуваат за менување единечен податочен елемент, пр. датотека или податочен објект. За да се применат овие промени, потребно е корисникот експлицитно да ја зачуваа содржината од едиторот.

Editors традиционално се позиционирани во одредена област, наречена **editor area**.

5.3. Perspective

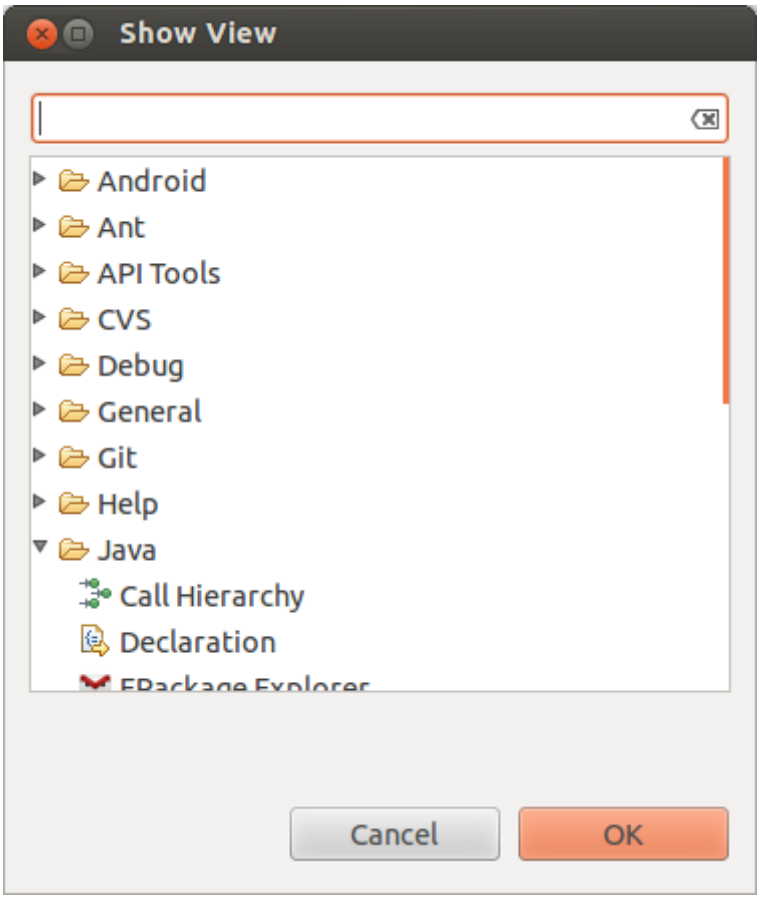
Perspective е визуелен контејнер на множество од делови **Parts**. Eclipse IDE користи **Perspectives** за да ги уреди **Parts** за различни задачи при развој. **Perspectives** се менуваат преку менито `Window → Open Perspective → Other` (Perspective).

Основните перспективи во Eclipse IDE се Java перспективата за развој и перспективата Debug за дебагирање на Java апликации.



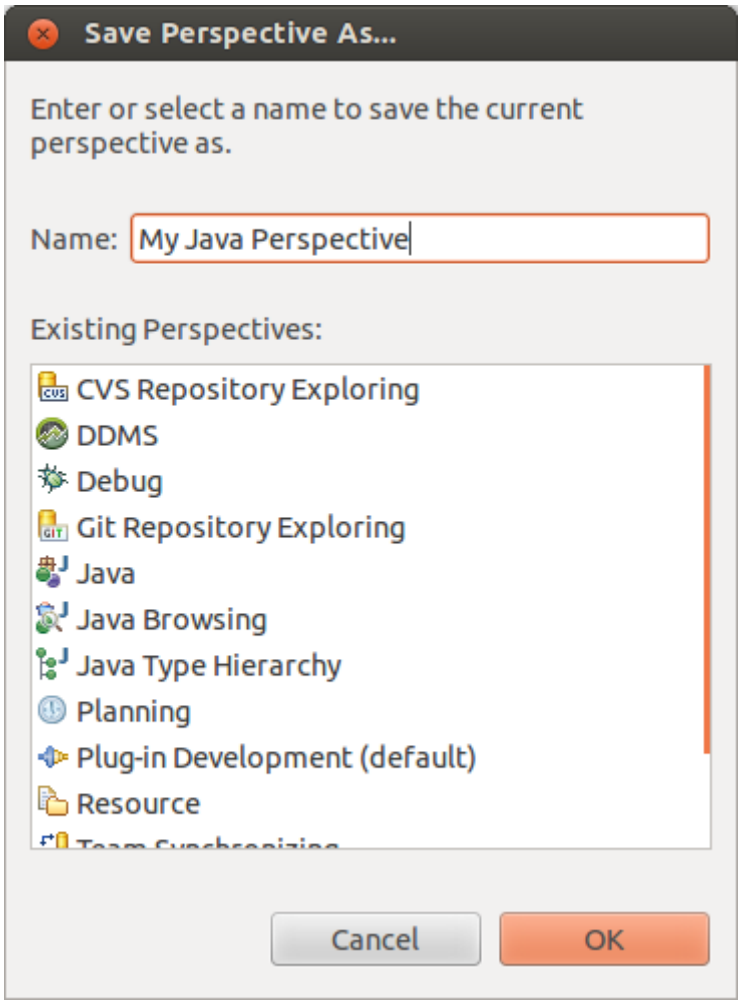
Слика 6. Менување перспективи во Eclipse IDE

Може да ги менувате позициите и содржината на деловите во **Perspective** со отварање и затворање или со едноставно уредување со влечење.



Слика 7. Show View дијалог

За да отворите нов **Part** во вашата тековна **Perspective** користете го менито `Window → Show View → Other`. Следниот `Show View` (Show View дијалог) дијалог ви овозможува да пребарувате одредени Parts.



Слика 8. Снимање на вашата перспектива конфигурација

Во случаи кога сакате да ја ресетирате вашата тековна перспектива на нејзината стандардна, можете преку менито `Window → Reset Perspective`.

Може да ја снимите вашата **Perspective** преку `Window → Save Perspective As...` (Снимање на вашата перспектива конфигурација).

6. Креирање на првата Java програма

Во следните неколку чекори ќе го опишеме процесот на креирање едноставна и минимална Java програма со користење Eclipse. Вообичаено во светот на програмирањето оваа програма испишува `Hello World` во конзолата, но ние ќе ја адаптираме да отпечати `Hello Eclipse!` стандардниот излез.

6.1. Креирање проект

1. Изберете од менито `File → New → Java project`.
2. Внесете `edu.finki.np.hello` како име на проектот.
    a. Изберете "Create separate folders for sources and class files".

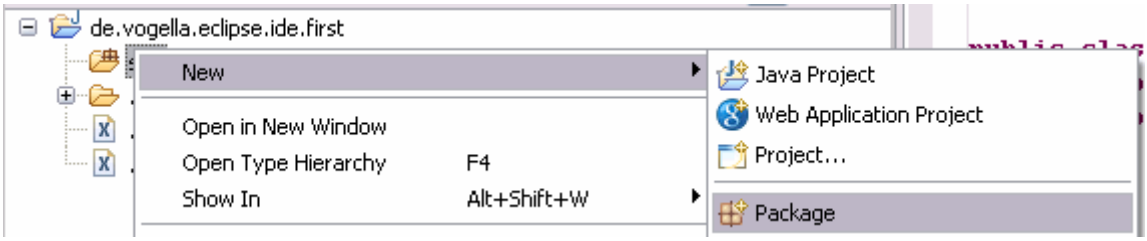
Слика 9. Волшебникот за нов Java Project

Притиснете на копчето Finish за да го креирате проектот. Креиран е нов проект и е прикажан како директориум. Отворете го `edu.finki.np.hello` и прегледајте ја неговата содржина.

6.2. Креирање пакети

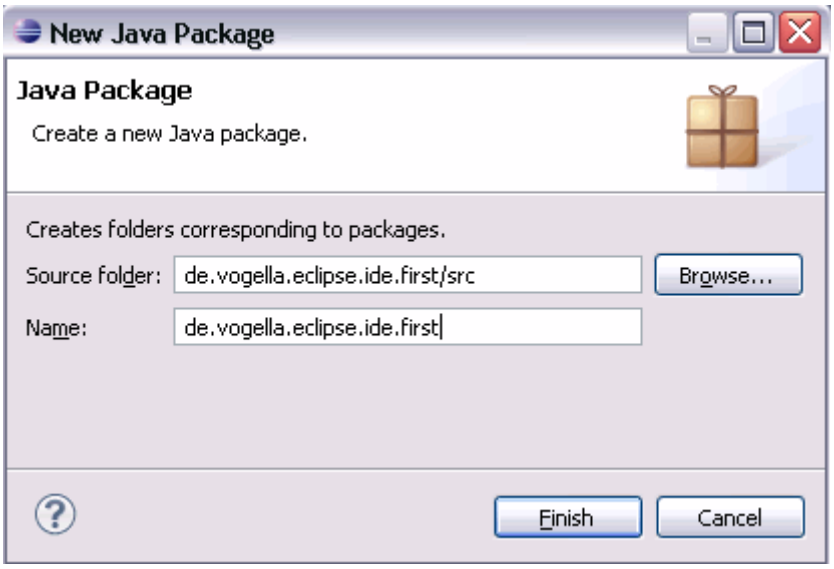
Во следниот чекор ќе креирате нов `package`. Добра конвенција е да користите исто име за проектот и пакетот на највисоко ниво.

Да креирате пакет `edu.finki.np.hello`, изберете го фолдерот `src` и со десен клик на него изберете `New → Package`.



Слика 10. Десен клик за креирање пакет

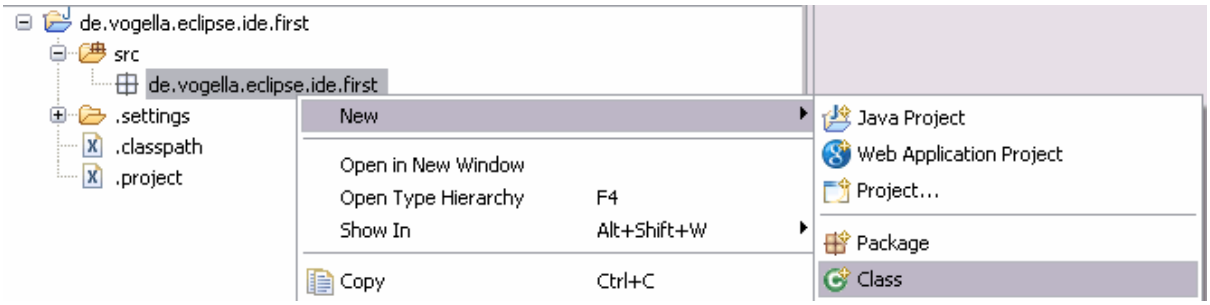
Внесете го името на новиот пакет во дијалогот.



Слика 11. Дијалог за креирање пакет

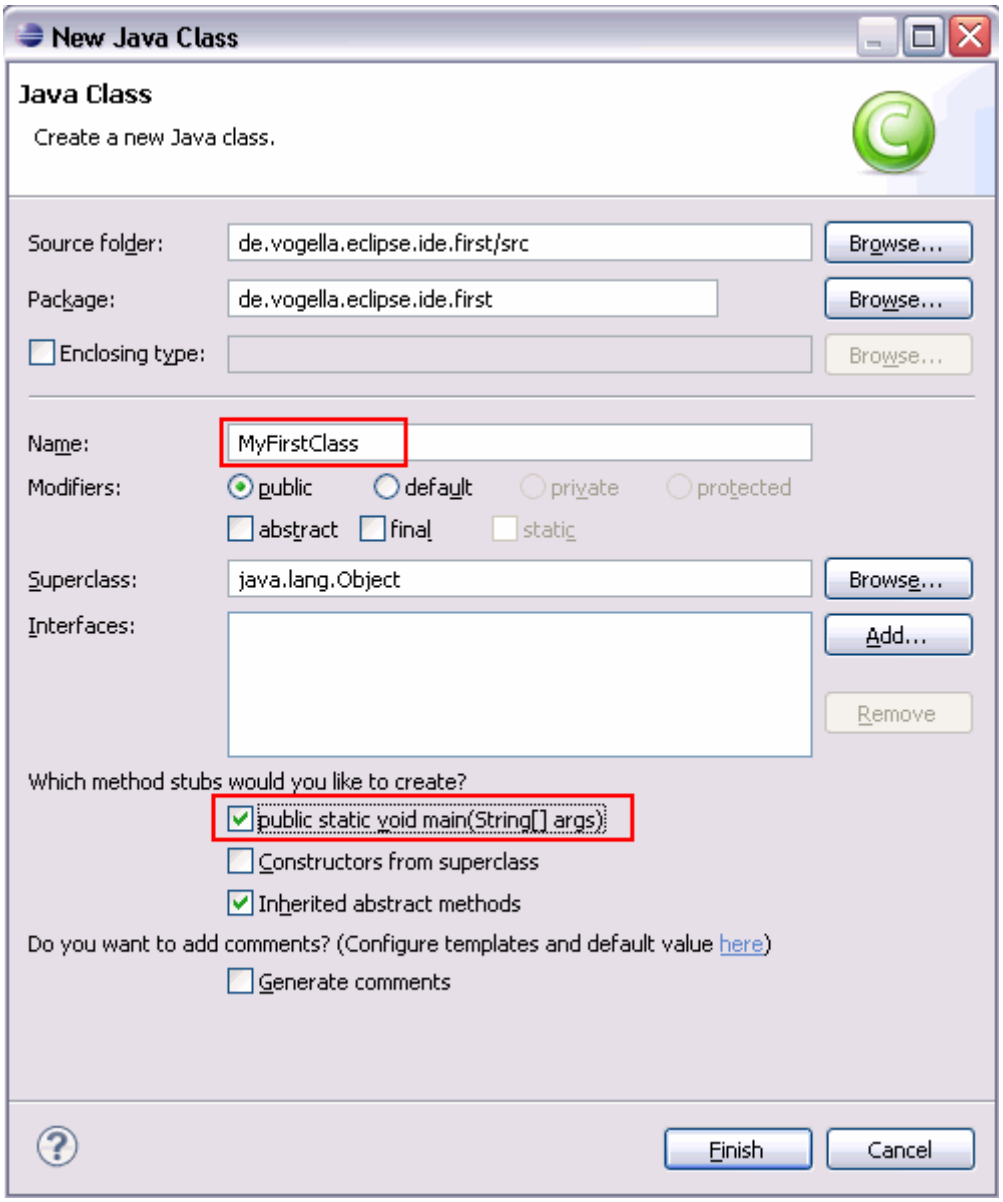
6.3. Креирање Java класа

Десен клик на пакетот и изберете `New → Class`.



Слика 12. Избор за креирање нова класа

Внесете `MyFirstClass1` како име на класата и изберете го `public static void main (String{[]{]] args)`.



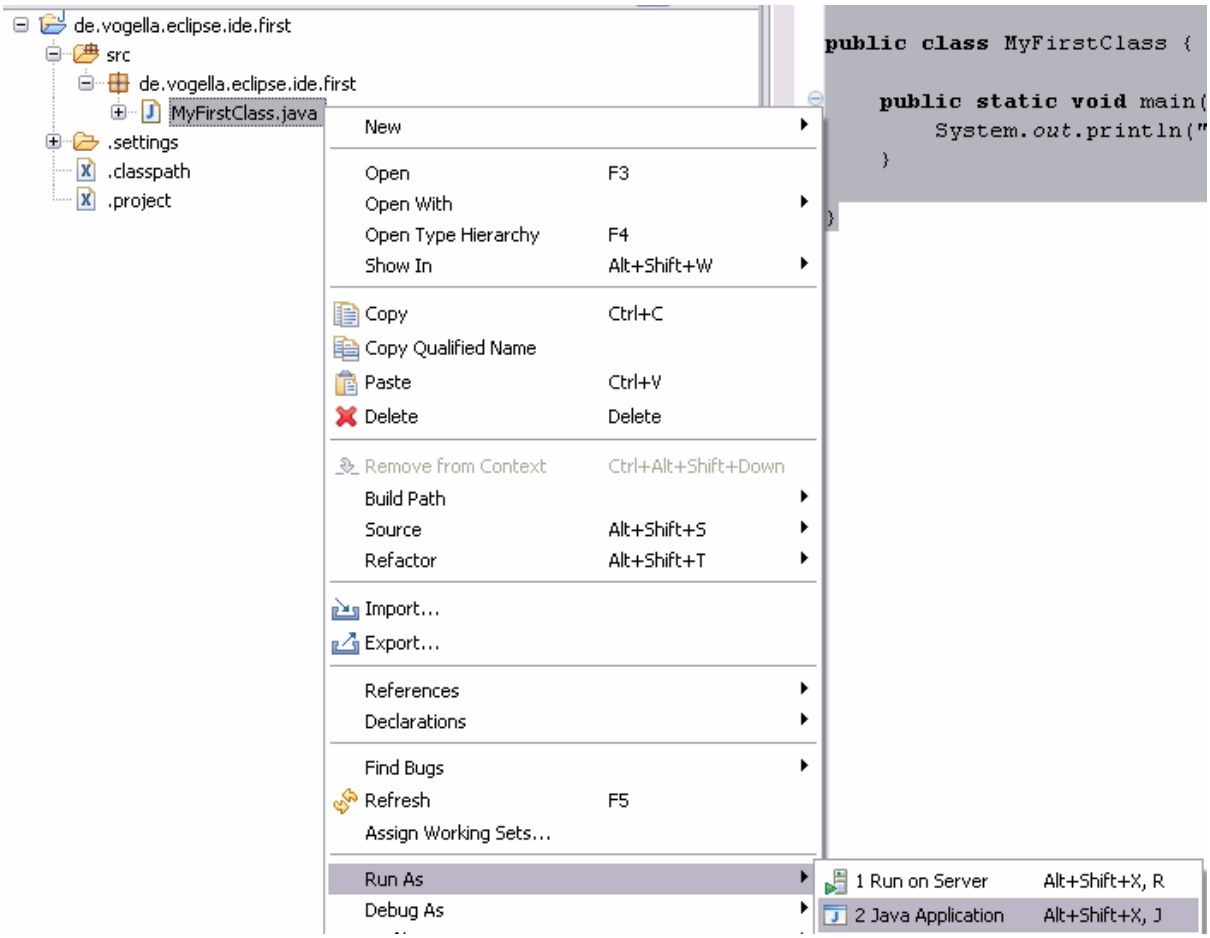
Слика 13. Избор за креирање нова класа

Ова создава нова датотека и ја отвара во **Editor** за Java изворни датотеки.

```
package edu.finki.np.first;

public class MyFirstClass {

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

}
```

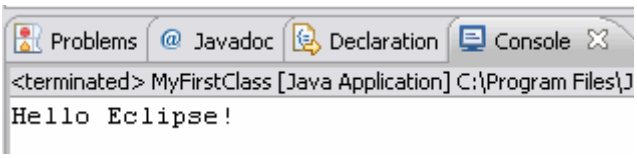
6.4. Извршување на проект во Eclipse

За да го извршите кодот, со десен клик на вашата Java класа изберете `Run-as → Java application`.



Слика 14. Ивршување проект

Eclipse ќе ја изврши вашата Java програма. Треба да го видите следниот излез во конзола **View**.



Резултат од извршување на апликацијата

Честитки! Го креиравте вашиот прв Java проект, пакет и Java класа и ја извршивте оваа програма во Eclipse.

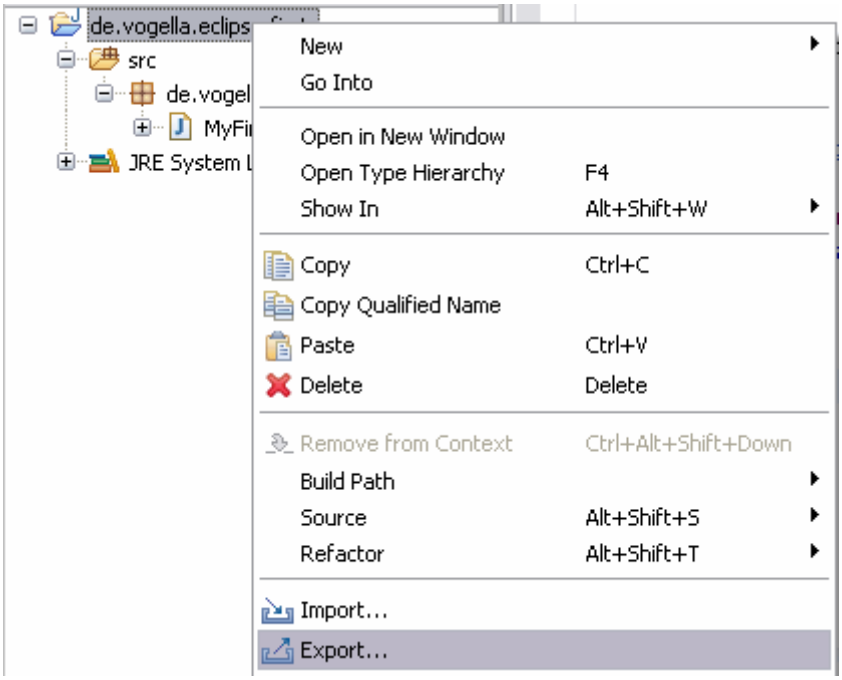
7. Извршување Java програма надвор од Eclipse

7.1. Креирање на jar датотека

За да извршите Java програма надвор од Eclipse треба да ја експортирате како

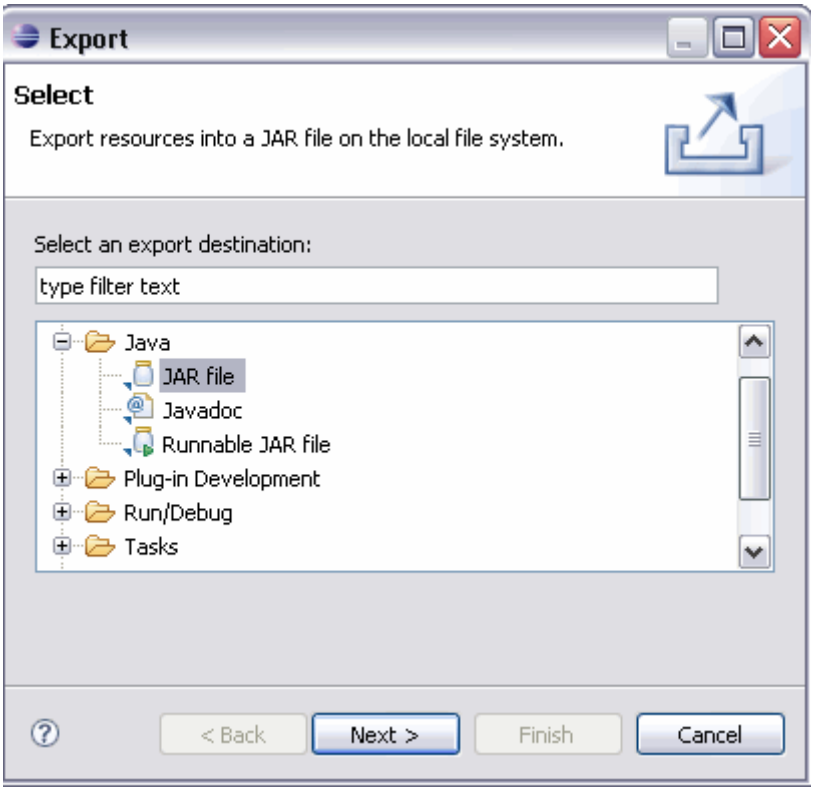
`jar` датотека. `jar` датотека е стандарден формат за дистрибуција на Java апликации.

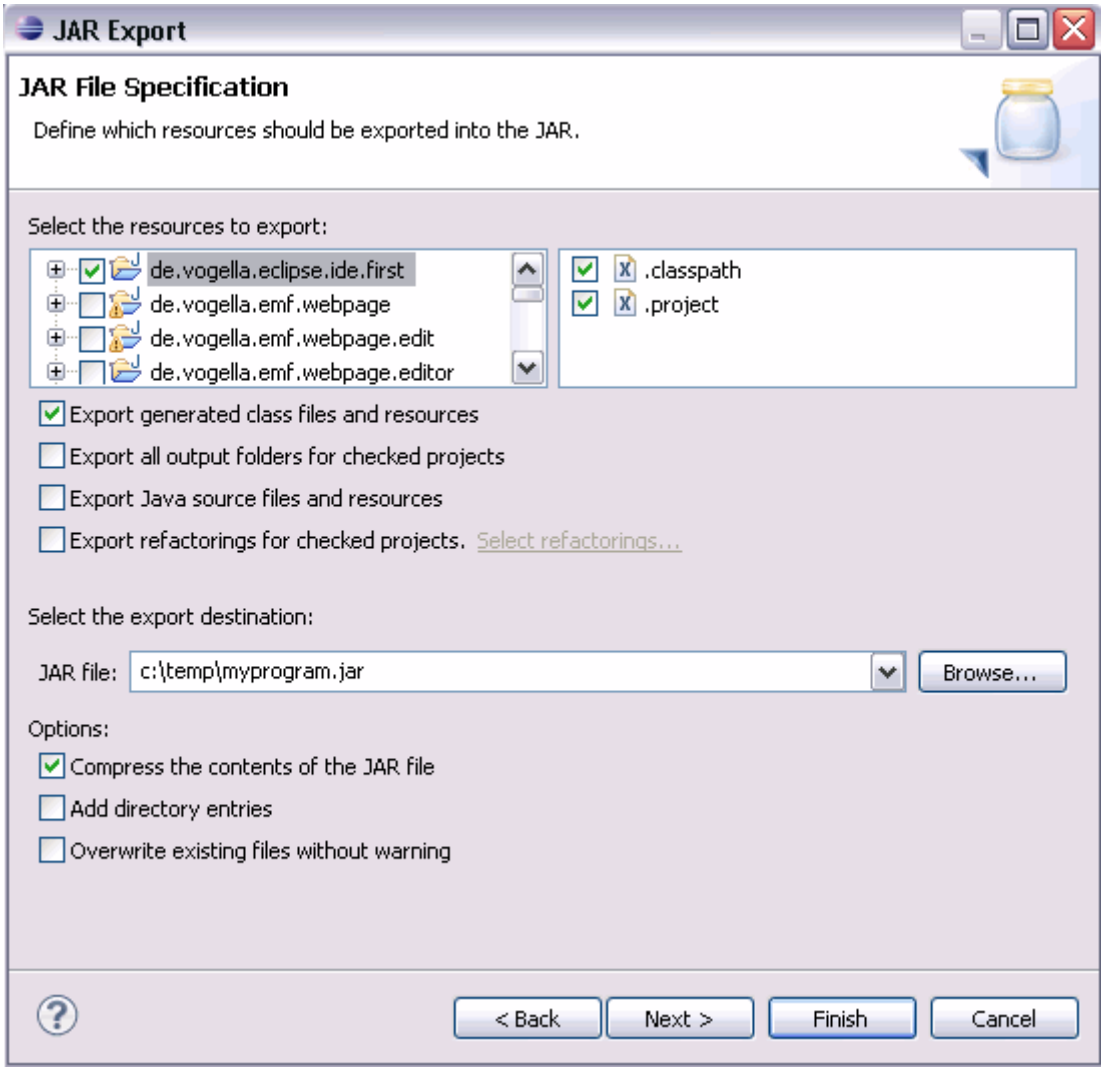
Изберете го вашиот проект, десен клик и изберете `Export`.



Волшебник за експортирање на Java проект

Изберете JAR датотека, изберете next. Изберете го вашиот проект и изберете си дестинација и име за jar датотеката. Пример `myprogram.jar`.





Слика 15. Волшебник за експортирање на Java проект, дел II

Волшебник за експортирање на Java проект, дел III

Честитки!

7.2. Извршете ја вашата програма надвор од Eclipse

Отворете командна линија.

Променете ја вашата работна патека со испишување `cd path`. На пример ако вашиот jar е лоциран во `c:\\temp` испишете `cd c:\\temp`.

За да ја извршите оваа програма треба да ја вклучите jar датотеката во вашиот `classpath`. Со `classpath` се дефинираат сите Java класи кои се овозможени во Java извршната околина. Може да додате `jar` датотека во classpath со опцијата `-jar`.

```
java -classpath myprogram.jar edu.finki.np.first.MyFirstClass
```

Ако ја испишите точно наведената команда и се наоѓате во соодветниот директориум, треба да видите порака `Hello Eclipse!` во конзолата.

Слика 16. Извршување апликација надвор од Eclipse

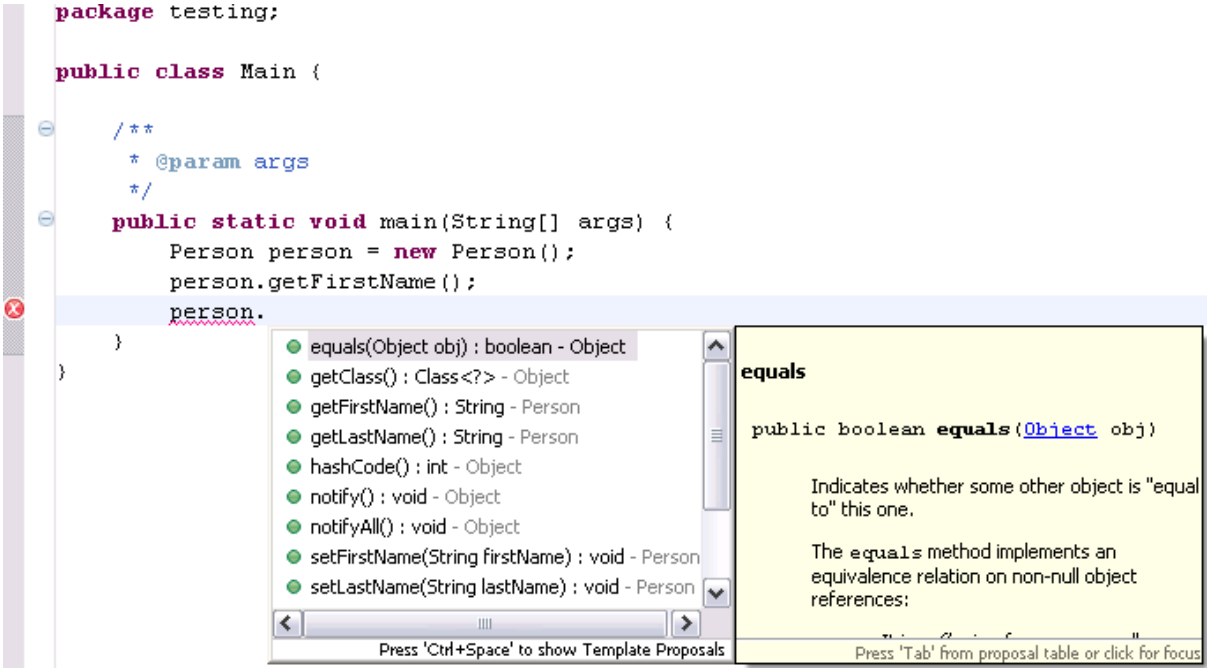
8. Content Assist, Quick Fix и Class Navigation

8.1. Content assist

Помошникот за содржина ви овозможува да добиете помош во самиот едитор. Може да се повика со `Ctrl+Space`

На пример испишете `syso` во едиторот на Java изворен код и притиснете `Ctrl+Space`. Ова ќе го замени `syso` со `System.out.println("").`

Ако имате референца кон објект, како на пример објектот `person` од типот `Person` и сакате да ги видите неговите методи, испишете `person.` и притиснете `Ctrl+Space`.



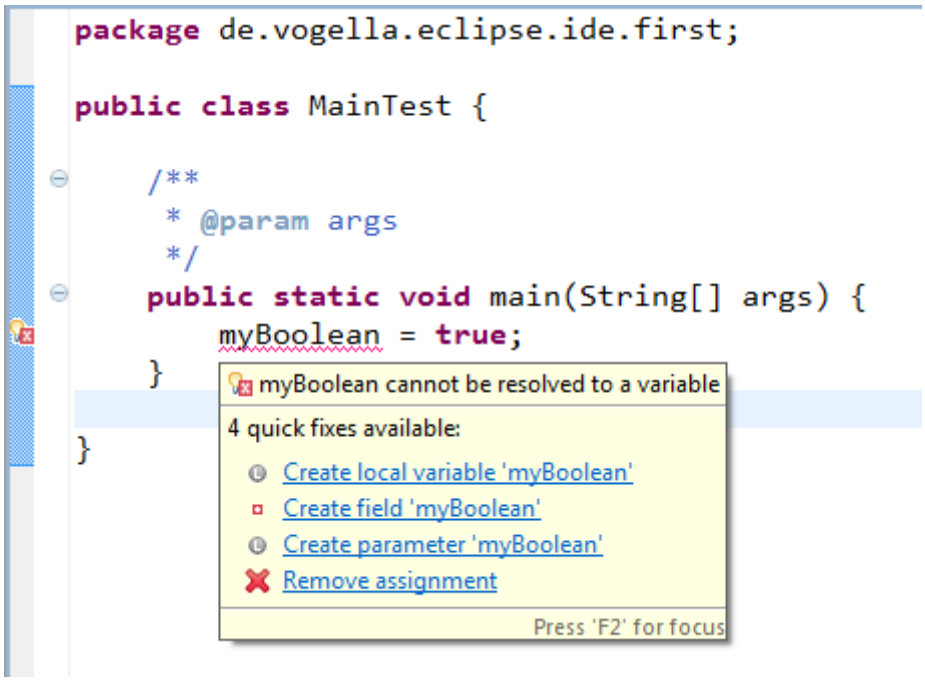
Слика 17. Помошник во содржината

8.2. Quick Fix

Секогаш кога Eclipse детектира некаков проблем, ќе ве го подцрта проблематичниот текст во едиторот. Изберете го овој текст и притиснете `Ctrl+1` за да видите можни начини за да го решите овој проблем.

На пример напишете `myBoolean = true;` Ако myBoolean не е сè уште дефинирана, Eclipse ќе ја означи како грешка. Изберете ја променливата и

притиснете `Ctrl+1`, Eclipse ќе ви предложи креирање на член или локална променлива.



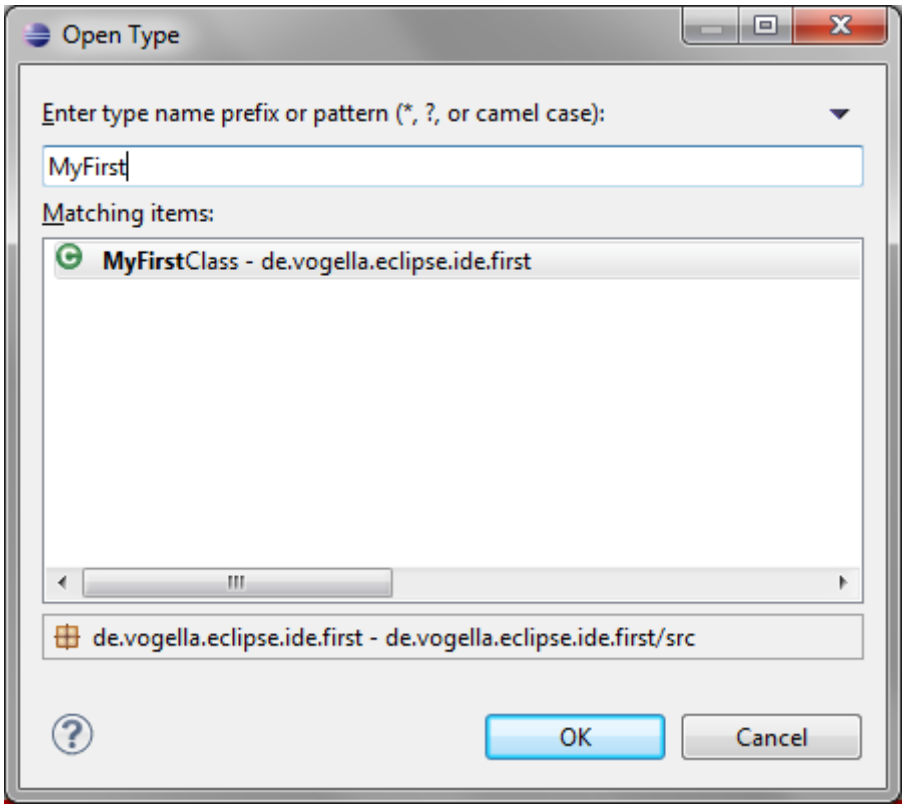
Слика 18. Пример со користење Quickfix

Quick Fix е многу моќна алатка. Ви овозможува да креирате нови локални променливи и членови, како и нови методи и класи. Исто така може да додава `try-catch` изрази околу исклучоците, а може да извршува и доделување на променливи, како и многу повеќе.

8.3. Отворање класа

Може да навигирате помеѓу класите во вашиот проект преку `Package Explorer` **View**.

Исто така може да ја отворите било која класа ако го позиционирате покажувачот врз името на класата и притиснете `F3`. Алтернативно но и многу моќно, може да притиснете `Ctrl+Shift+T`. Ова ќе ви отвори дијалог во кој може да ја пребарате класата по нејзиното име и да ја отворите.



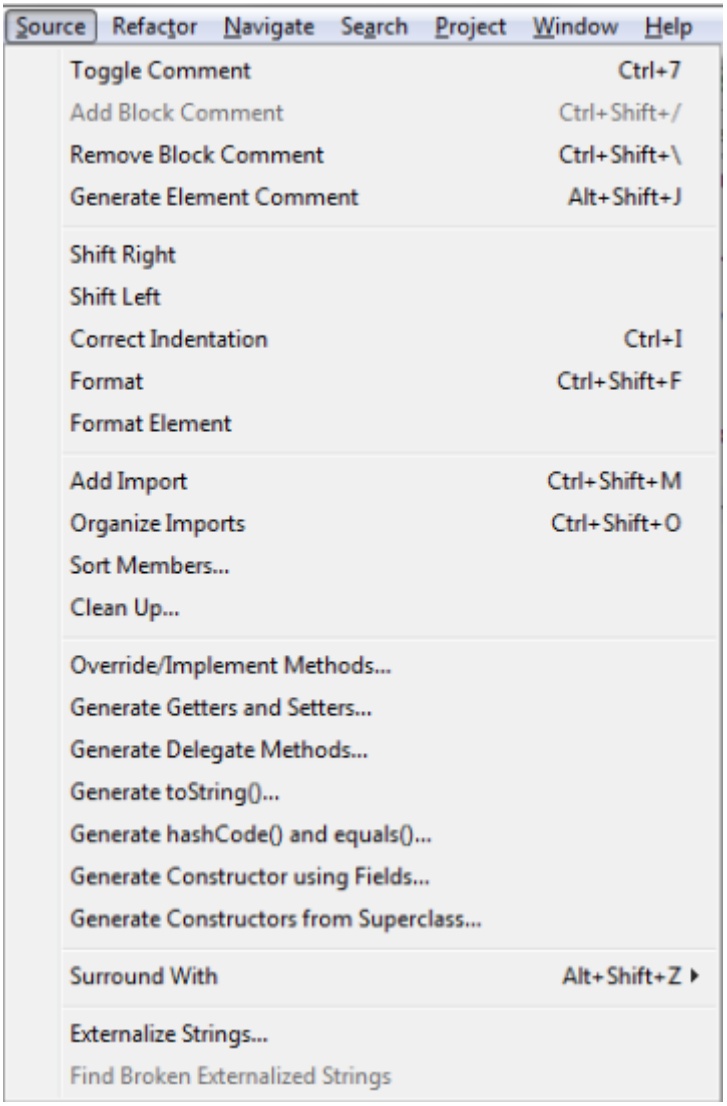
Слика 19. Отворање класа

8.4. Генерирање код

Eclipse има неколку можности за генерирање код за вас. Ова може да ви заштеди значително време при развој.

На пример Eclipse може да ги препокрие методите од суперкласите и генерира `toString()`, `hashcode()` и `equals()` методи. Исто така може да генерира и getter и setter методи за атрибутите во вашата Java класа.

Овие опции може да се најдата во менито Source.

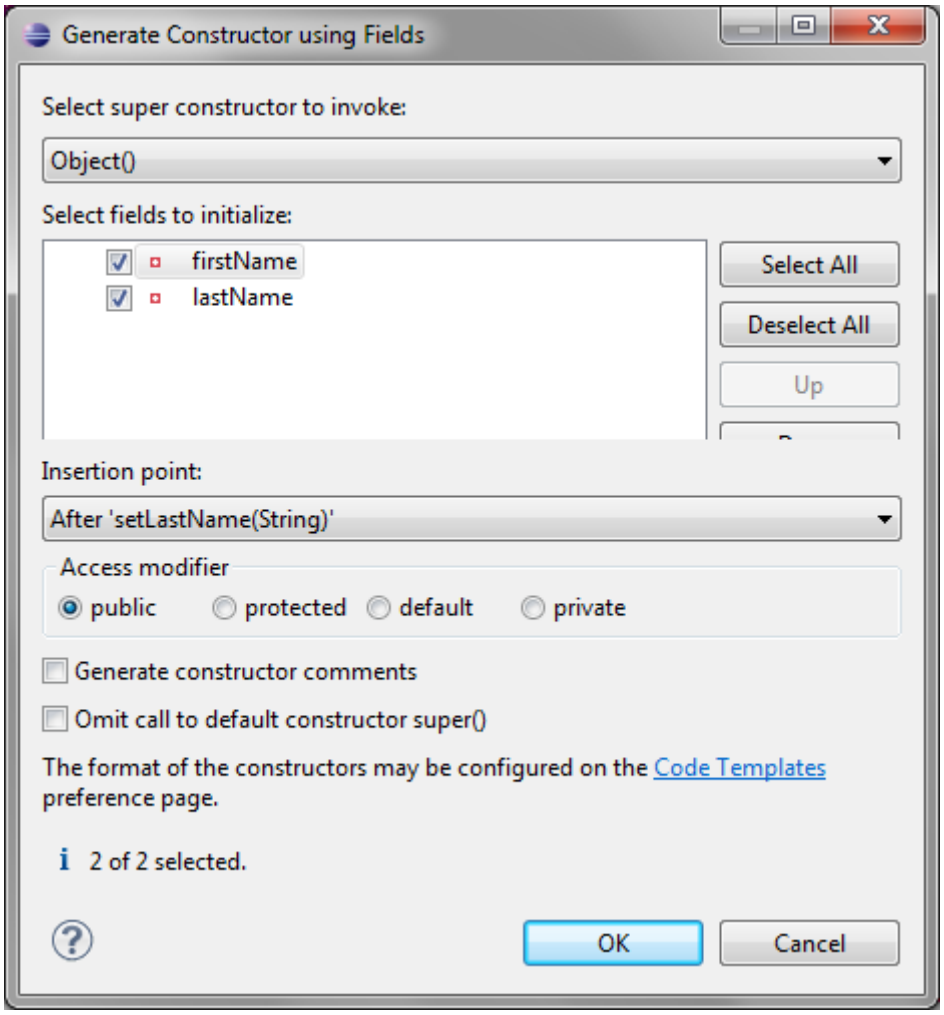


Слика 20. Генерирање код

За да го тестирање генерирањето на код, ќе ја креираме следната класа во `edu.finki.np.first` проектот.

```
package edu.finki.np.first;

public class Person {
  private String firstName;
  private String lastName;

}
```

Изберете `Source → Generate Constructor from Fields`, маркирајте ги двете полиња и притиснете `OK`.



Слика 21. Генерирање

Изберете `Source → Generate Getter and Setter`, изберете ги повторно двете полиња и притиснете го `OK` копчето.

Изберете `Source → Generate toString()`, маркирајте ги повторно двете полиња и притиснете `OK`.

Го генериравте следниот код:

```java
package edu.finki.np.first;

public class Person {
  private String firstName;
  private String lastName;

  public Person(String firstName, String lastName) {
    super();
    this.firstName = firstName;
    this.lastName = lastName;
  }

  public String getFirstName() {
    return firstName;
  }

  public void setFirstName(String firstName) {
    this.firstName = firstName;
  }

  public String getLastName() {
    return lastName;
  }

  public void setLastName(String lastName) {
    this.lastName = lastName;
  }

  @Override
  public String toString() {
    return "Person [firstName=" + firstName + ", lastName=" + lastName
        + "]";
  }

}
```

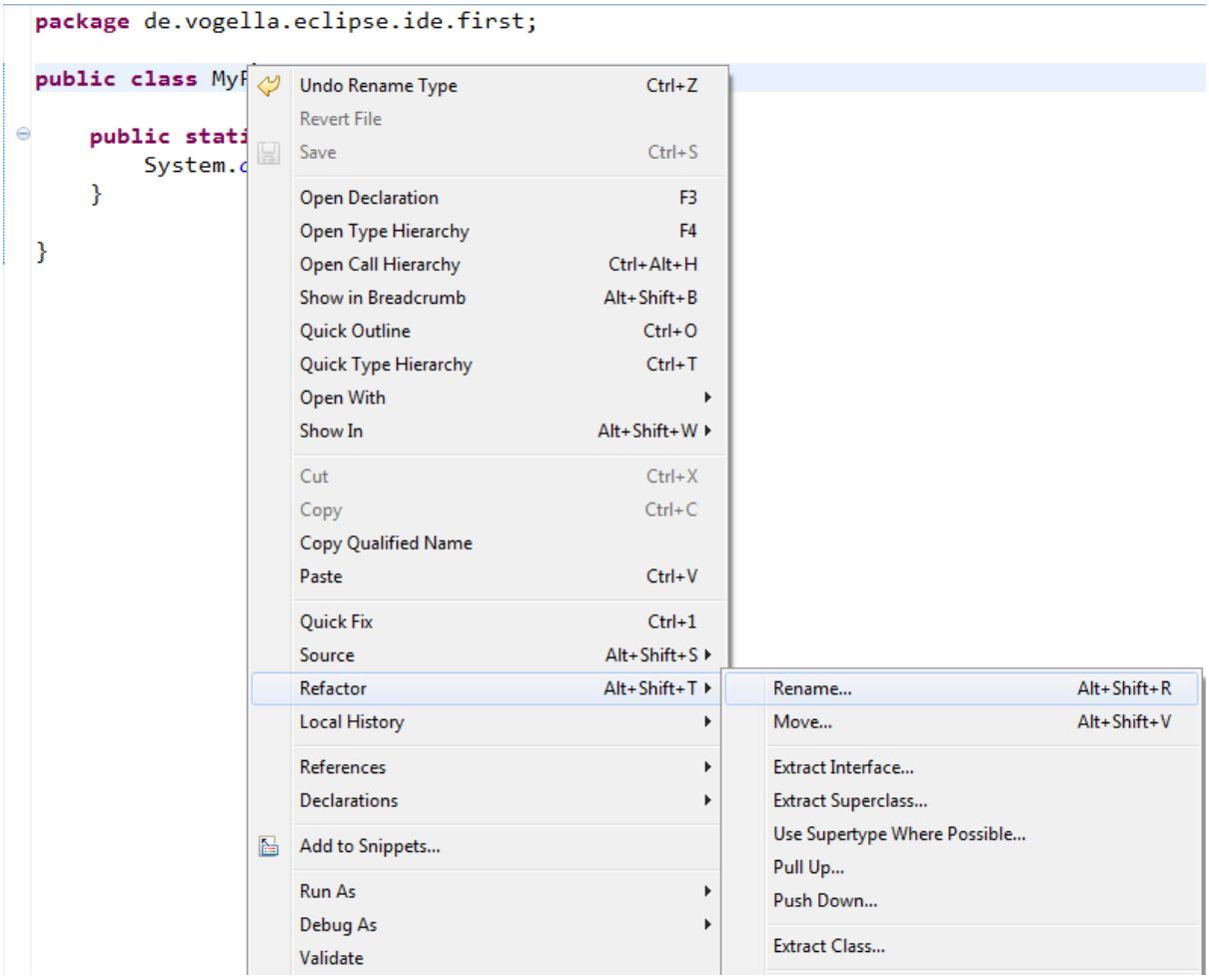
9. Refactoring

9.1. Refactoring во Eclipse

Refactoring is the process of restructuring the code without changing his behavior. For example renaming a Java class or method is a refactoring activity.

Eclipse supports simple refactoring activities, for example renaming or moving. For example you can select your class, right click on it and select `Refactor → Rename to rename your class or method`. Eclipse will make sure that all calls in your Workspace to your your class or method will also be renamed.

The following shows a screenshot for calling the Rename refactoring on a class.



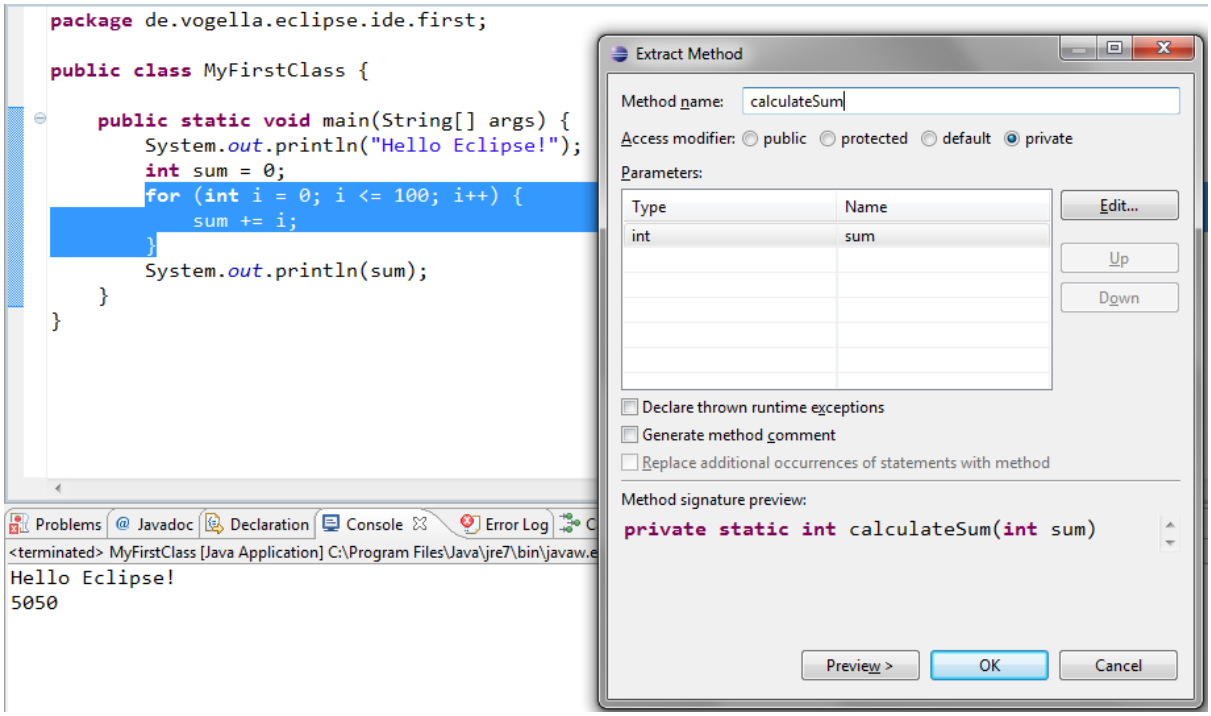
Слика 22. Renaming a class

9.2. Refactoring Examples

For the next examples change the code of your `MyFirstClass` class to the following.

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

public class MyFirstClass {

  public static void main(String[] args) {
    System.out.println("Hello Eclipse!");
    int sum = 0;
    for (int i = 0; i <= 100; i++) {
      sum += i;
    }
    System.out.println(sum);
  }
}
```

Another useful refactoring is to mark code and create a method from the selected code. For this mark the coding of the `for` loop, right click and select `Refactoring → Extract Method`. Use `calculateSum` as name of the new method.

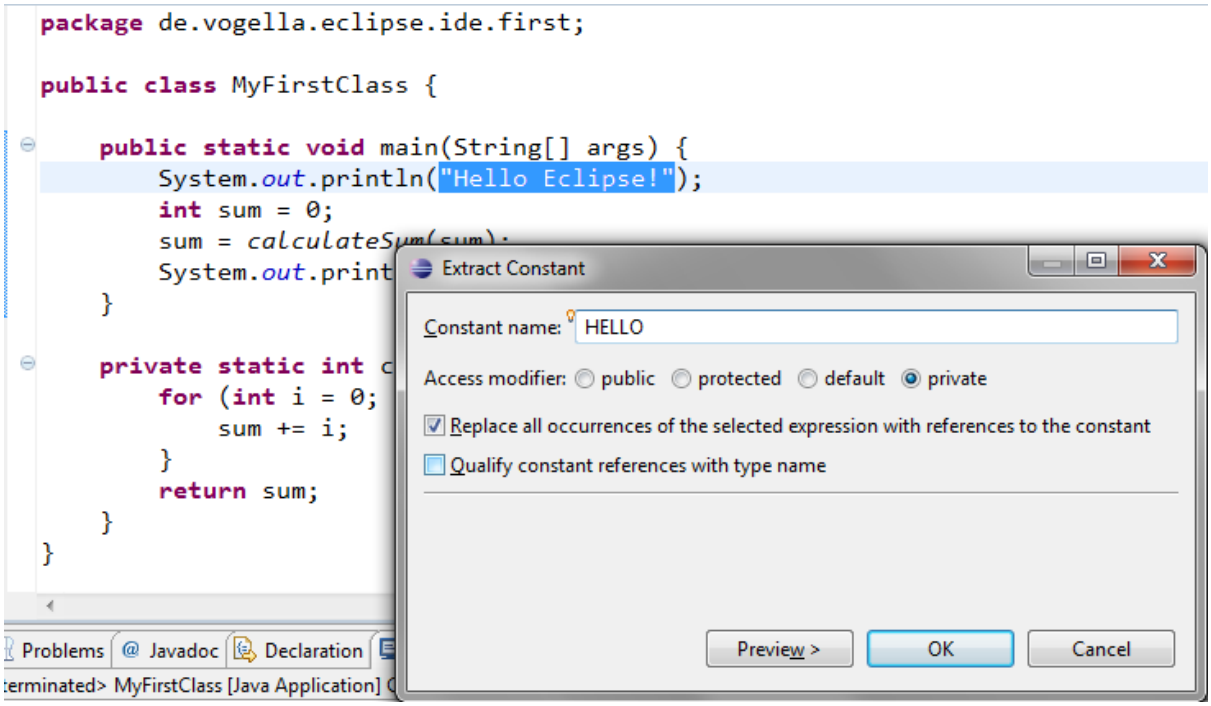


Слика 23. Extract Method refactoring

The resulting class should look like the following.

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

public class MyFirstClass {

  public static void main(String[] args) {
    System.out.println("Hello Eclipse!");
    int sum = 0;
    sum = calculateSum(sum);
    System.out.println(sum);
  }

  private static int calculateSum(int sum) {
    for (int i = 0; i <= 100; i++) {
      sum += i;
    }
    return sum;
  }
}
```

You can also extract strings and create constants from them. Mark for this example `Hello Eclipse!`, right click on it and select `Refactor → Extract Constant`. Name your new constant `HELLO`.



Extract Constants

The resulting class should look like the following.

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

public class MyFirstClass {

  private static final String HELLO = "Hello Eclipse!";

  public static void main(String[] args) {
    System.out.println(HELLO);
    int sum = 0;
    sum = calculateSum(sum);
    System.out.println(sum);
  }

  private static int calculateSum(int sum) {
    for (int i = 0; i <= 100; i++) {
      sum += i;
    }
    return sum;
  }
}
```

Eclipse has much more refactorings, in most cases you should get an idea of the performed action by the naming of the refactoring operation.

10. Eclipse Shortcuts

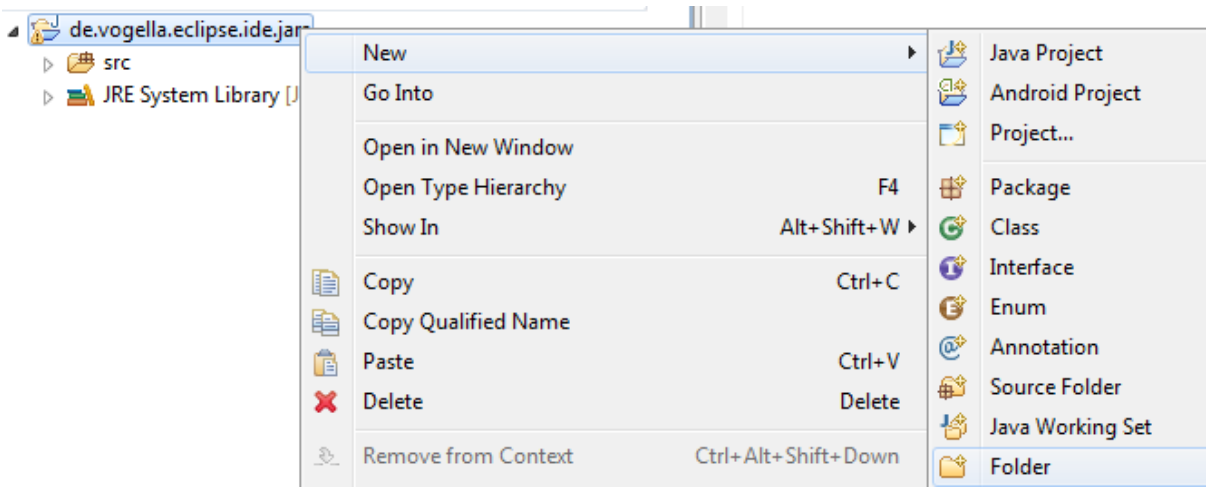
Eclipse provides a lot of shortcuts to work efficiently with the IDE. For a list of the most important Eclipse shortcuts please see Eclipse Shortcuts

11. Using jars (libraries)

11.1. Adding a library (.jar) to your project

The following describes how to add Java libraries to your project. Java libraries are distributed via `jar` files. It assumes that you have a jar file available; if not feel free to skip this step.

Create a new Java project `de.vogella.eclipse.ide.jars`. Then, create a new folder called `lib`, by right clicking on your project and selecting `New → Folder`.

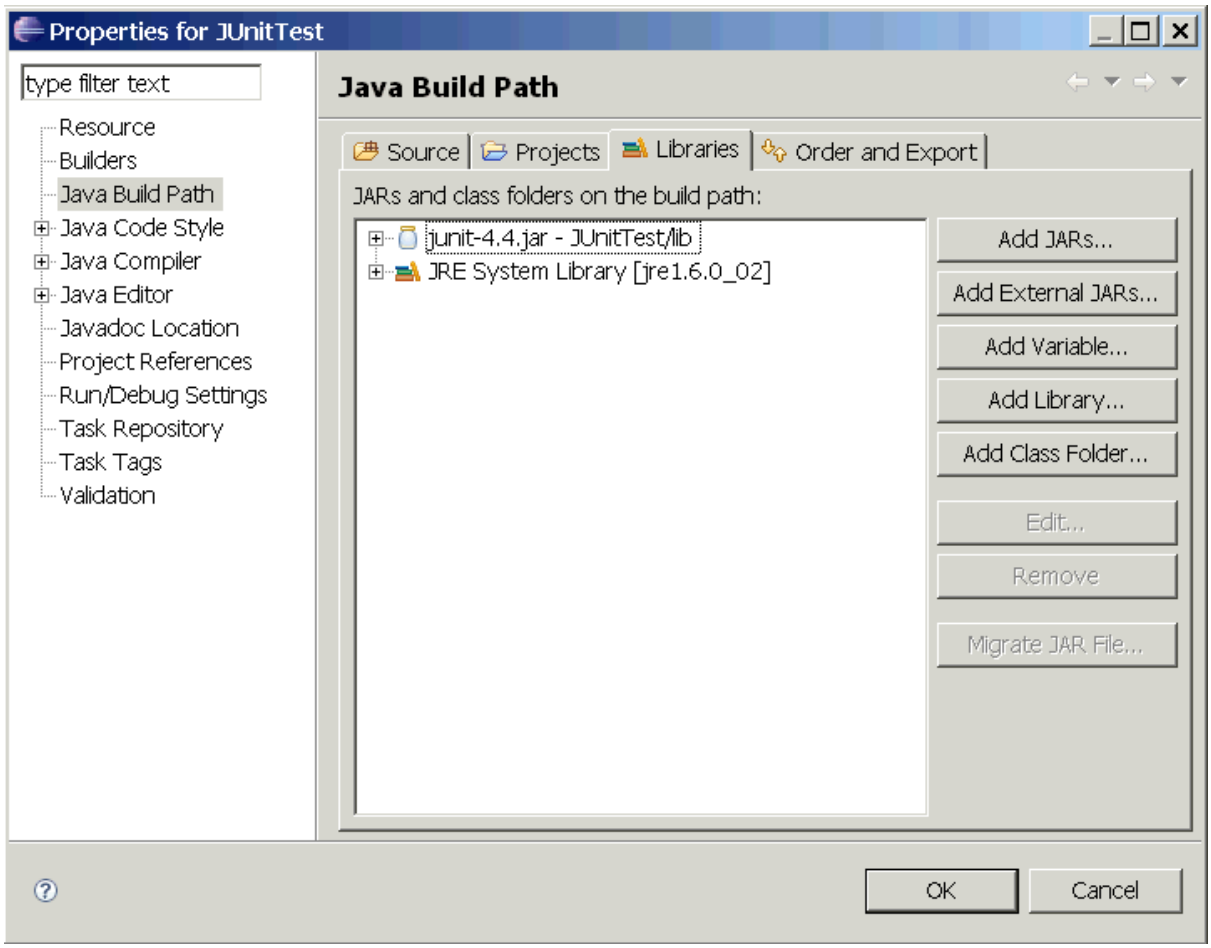


Creating a new folder

From the menu select `File → Import → General → File System`. Select your jar and select the `lib` folder as target. Alternatively, just copy and paste your `jar` file into the `lib` folder.

Right click on your project and select Properties. Under `Java Build Path → Libraries` select the **Add JARs button**.

The following example shows how the result would look like, if the `junit-4.4.jar` file had been added to the project.



Слика 24. Adding a jar to the current project

Afterwards you can use the classes contained in the `jar` file in your Java source code.

11.2. Attach source code to a Java library

As said earlier 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.

If the source code is not available, the editor will show the decompiled bytecode of that class.

This happens if you open a class from Java library and the source for this .jar file is not available. The same happens if you open a class from the standard Java library without attaching the source code to it.

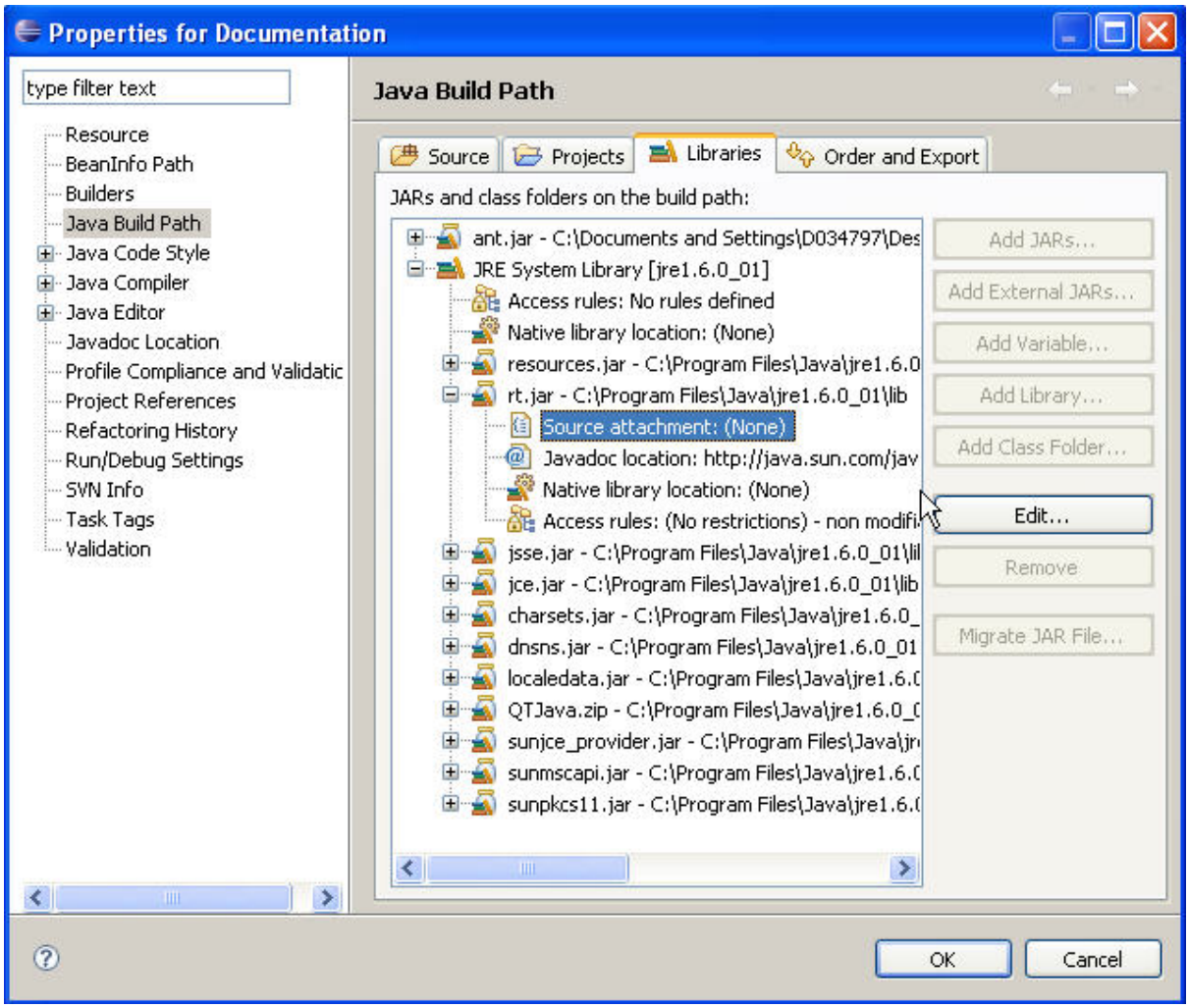
To browse the source of a type contained in a library (i.e. .jar file), you can attach a source archive or source folder to that library. Afterwards the editor will show the source instead of the bytecode.

In addition setting the source attachment allows debugging this source code.

The Source Attachment dialog can be reached in the Java Build Path page of a project. To open this page, right click on a project → `Properties → Java Build Path`. On the Libraries tab, expand the library's node, select the Source attachment attribute and press the Edit button.

In the Location path field, enter the path of an archive or a folder containing the source.

The following shows this for the standard Java library. If you have the Java Development Kit (JDK) installed, you should find the source in the JDK installation folder. The file is typically called `src.zip`.



Слика 25. Maintaining the location of the source attachment to an jar

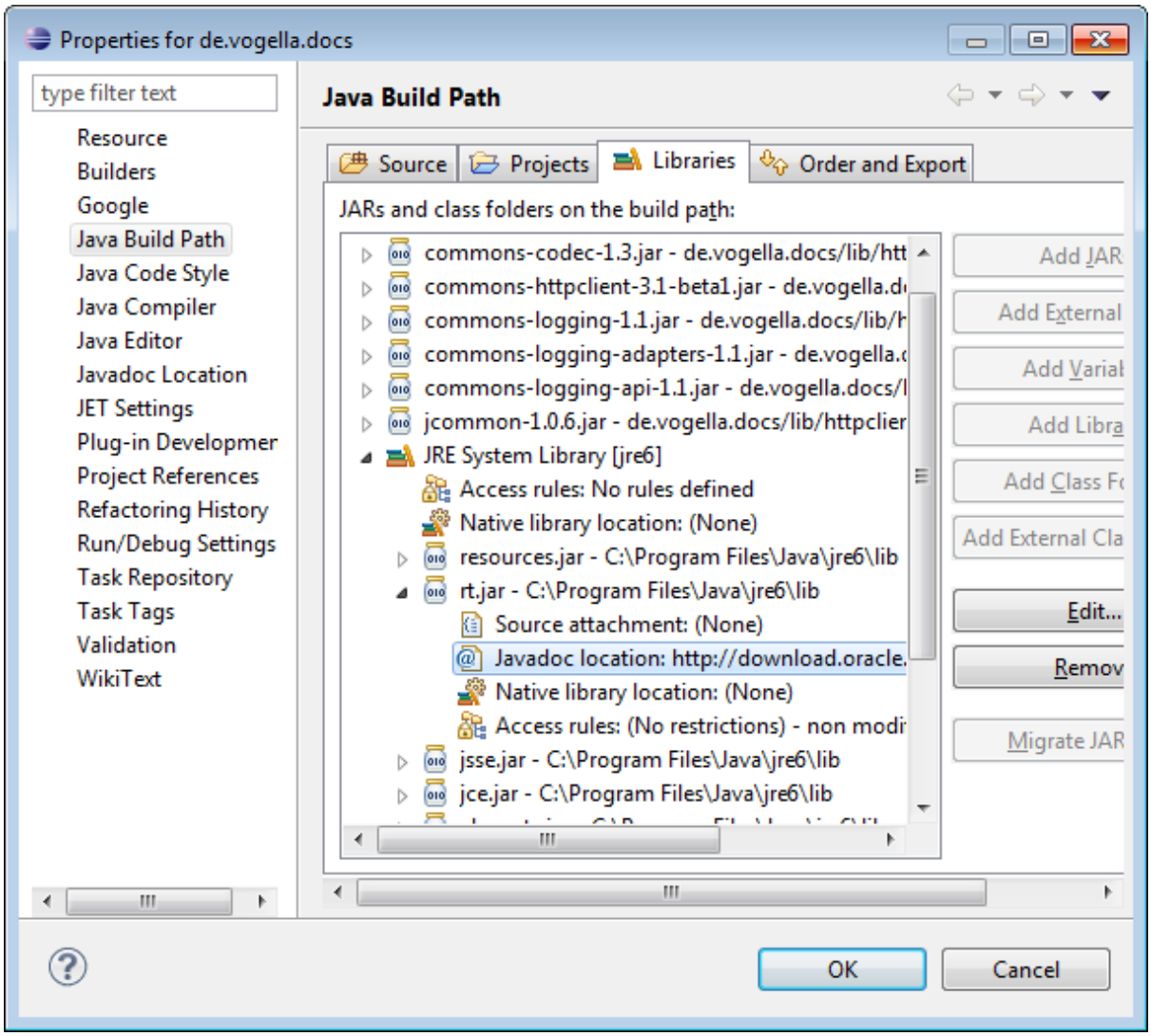
11.3. Add the Javadoc for a jar

It is also possible to add Javadoc to a library which you use.

Download the Javadoc of the jar and put it somewhere in your filesystem.

Open the Java Build Path page of a project via Right click on a project → `Properties` → `Java Build Path`. On the Libraries tab expand the library's node, select the **`Javadoc location`** attribute and press the Edit button.

Enter the location to the file which contains the Javadoc.



Слика 26. Maintain the location to the Javadoc file for a jar file