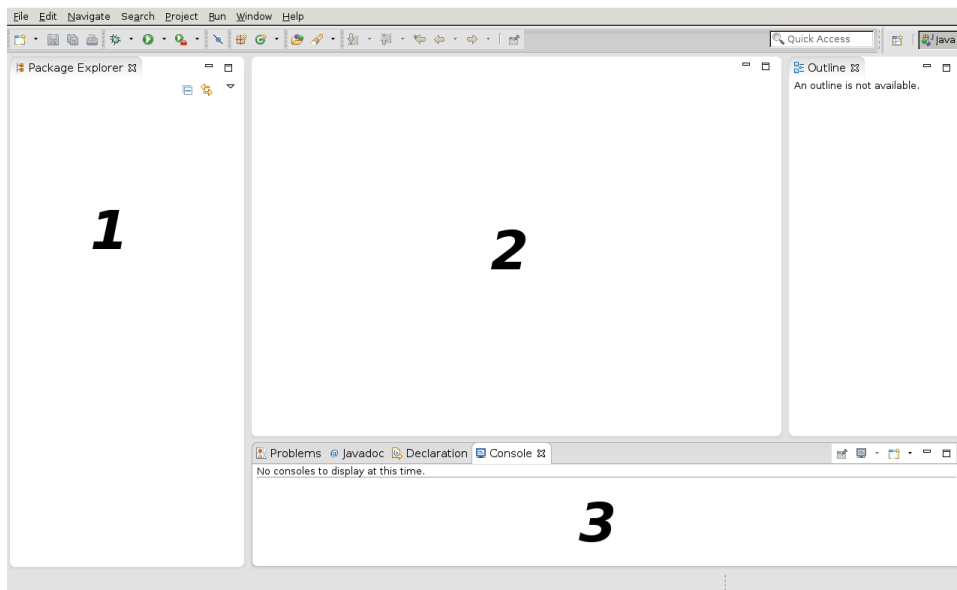


## Eclipse Übersicht



- 1 Hier werden die Projekte angezeigt und die jeweiligen Dateien (z.B. Klassen) der Projekte.
- 2 Hier kann man Dateien editieren.
- 3 Die Konsole. Hier wird der Input/Output des Programms angezeigt sobald es ausgeführt wird.

### Ich sehe eine der Ansichten nicht, wie kann ich sie einblenden?

Im Menu unter **Window > Show View > ...** können alle verschiedenen Fenster eingeblendet werden, falls sie geschlossen wurden.

## Projekte

In Java arbeitet man immer mit *Projekten*. Ein Projekt entspricht einem Programm, d.h. solange Sie an *einem* Programm arbeiten, arbeiten Sie *immer im gleichen Projekt*.


Wenn Sie mit anderen Leuten zusammen arbeiten, so muss das Projekt nur *einmal* erstellt werden. Der Arbeitsablauf um ein Projekt zu erschaffen, und dass dann alle Gruppenmitglieder das Projekt in ihrem Eclipse haben ist wie folgt (**Achtung: Die Schritte müssen in dieser Reihenfolge ausgeführt werden!**):

1. Jemand erstellt das Projekt bei sich in Eclipse. Nur diese Person *erstellt* ein Projekt! (Siehe “Ein neues Projekt anlegen (im Git Repository)”) )
2. Die gleiche Person lädt das Projekt auf den Server. (Siehe “Das erstellte Projekt pushen”)
3. Jetzt sind die anderen Gruppenmitglieder dran: Sie updaten ihr repository (**git pull**) und importieren das Projekt in Eclipse. (Siehe “Das Projekt importieren”)

## Ein neues Projekt anlegen (im Git Repository)

Wenn Sie ein neues Projekt anlegen wollen können sie das über das Menu **File > New > Java Project** machen.

**Create a Java Project**  
Enter a project name.

Project name:  **1**

☐ Use default location

Location:  **2**

JRE

☒ Use an execution environment JRE:

☐ Use a project specific JRE:

☐ Use default JRE (currently 'java-7-openjdk-amd64') [Configure JREs...](#)

Project layout

☐ Use project folder as root for sources and class files

☒ Create separate folders for sources and class files [Configure default...](#)

Working sets

☐ Add project to working sets

Working sets:

- 1** Geben Sie dem Projekt einen Namen. Wenn Sie z.B. das Spiel "Space Invaders" programmieren, würden Sie das Projekt "SpaceInvaders" nennen (wenn Sie auf Leerzeichen (white-spaces) verzichten macht das Ihr Leben deutlich einfacher!).
- 2** Weil wir mit Git arbeiten erstellen wir das Projekt *nicht* in der **default location**, sondern wählen das Verzeichnis auf der Festplatte aus, wo wir unser Git Repository erstellt haben.

### Ich möchte mehrere Projekte im gleichen Git Repository haben

Wenn Sie mehrere Projekte im gleichen Git Repository haben wollen, dann müssen Sie einen Unterordner im Repository erstellen und dann das Projekt darin anlegen.

## Das erstellte Projekt pushen

Sie haben nun das Projekt auf Ihrem Dateisystem im Git Repository erstellt – doch jetzt wollen Sie es auch noch uploaden, damit Ihr Projekt Partner das erstellte Projekt auch bei sich hat.

```
--|bash|~|-----
|-|david|-|x201|-$ cd lee/lee/

--|bash|~/lee/lee|-----
|-|david|-|x201|-$ git status
On branch master
Your branch is up-to-date with 'origin/master'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        .classpath
        .project

nothing added to commit but untracked files present (use "git add" to track)

--|bash|~/lee/lee|-----
|-|david|-|x201|-$ git add .classpath .project

--|bash|~/lee/lee|-----
|-|david|-|x201|-$ git status
On branch master
Your branch is up-to-date with 'origin/master'.

Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        new file:   .classpath
        new file:   .project

--|bash|~/lee/lee|-----
|-|david|-|x201|-$ git commit -m "Projekt angelegt."
[master aecab9b] Projekt angelegt.
2 files changed, 23 insertions(+)
create mode 100644 .classpath
create mode 100644 .project

--|bash|~/lee/lee|-----
|-|david|-|x201|-$ git push
Username for 'https://github.com': stolzda
Password for 'https://stolzda@github.com':
Counting objects: 4, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 654 bytes | 0 bytes/s, done.
Total 4 (delta 0), reused 0 (delta 0)
To https://github.com/stolzda/lee.git
246a6ec..aecab9b master -> master
```

**cd REPOSITORY LOCATION** Sie gehen ins Verzeichnis in dem sie das Repository geklont haben, hier wurde es nach “lee/lee” geklont.

**git status** Sie sehen, welche Dateien sich seit dem letzten **commit** geändert haben. Da wir das Projekt neu angelegt haben, gibt es zwei neue Dateien: **.classpath** und **.project**. Diese müssen beide committed werden damit jemand anders das Projekt direkt bei Eclipse importieren kann.

**git add .classpath .project** Wir fügen die beiden Dateien zum nächsten commit hinzu.

**git status** Wir schauen wir der Zustand unserer Kopie des Repositories jetzt ist. Wir sehen dass die beiden Dateien für den nächsten commit vorgemerkt sind.

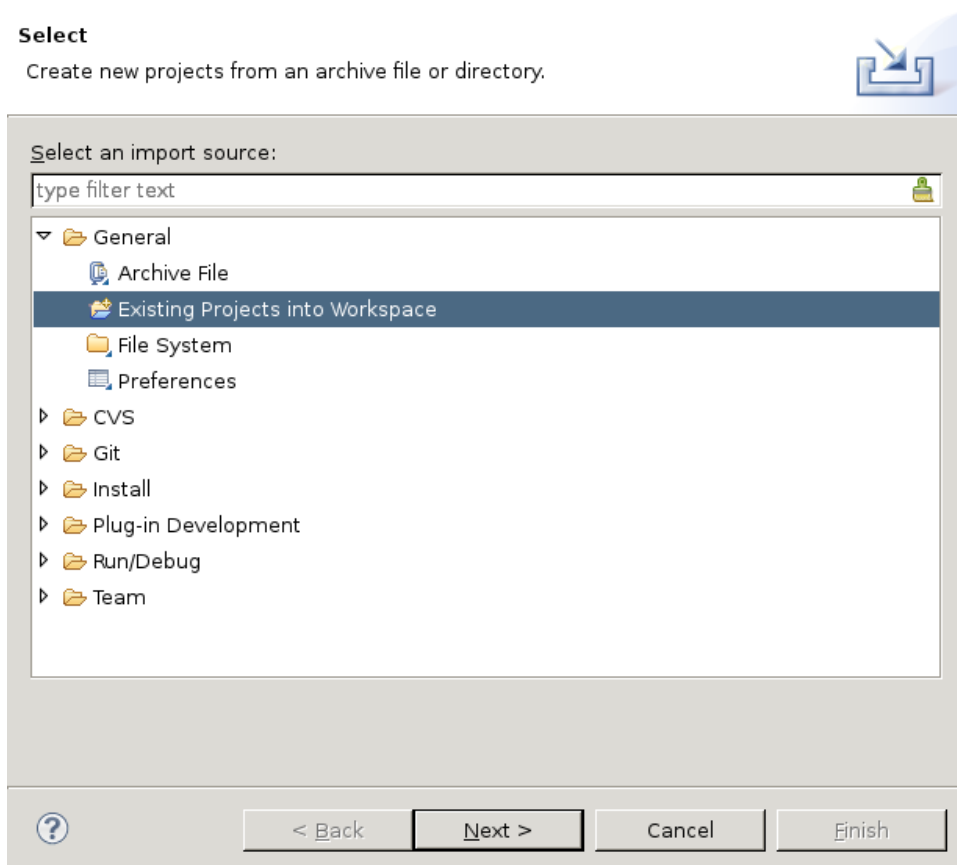
**git commit -m “Projekt angelegt.”** Wir erstellen einen commit mit der Nachricht “Projekt angelegt.”, damit wir später noch wissen was wir in diesem commit gemacht haben.

**git push** Da der commit bis jetzt nur lokal war, laden wir den commit auch auf den Server. Dazu pushen wir den commit.

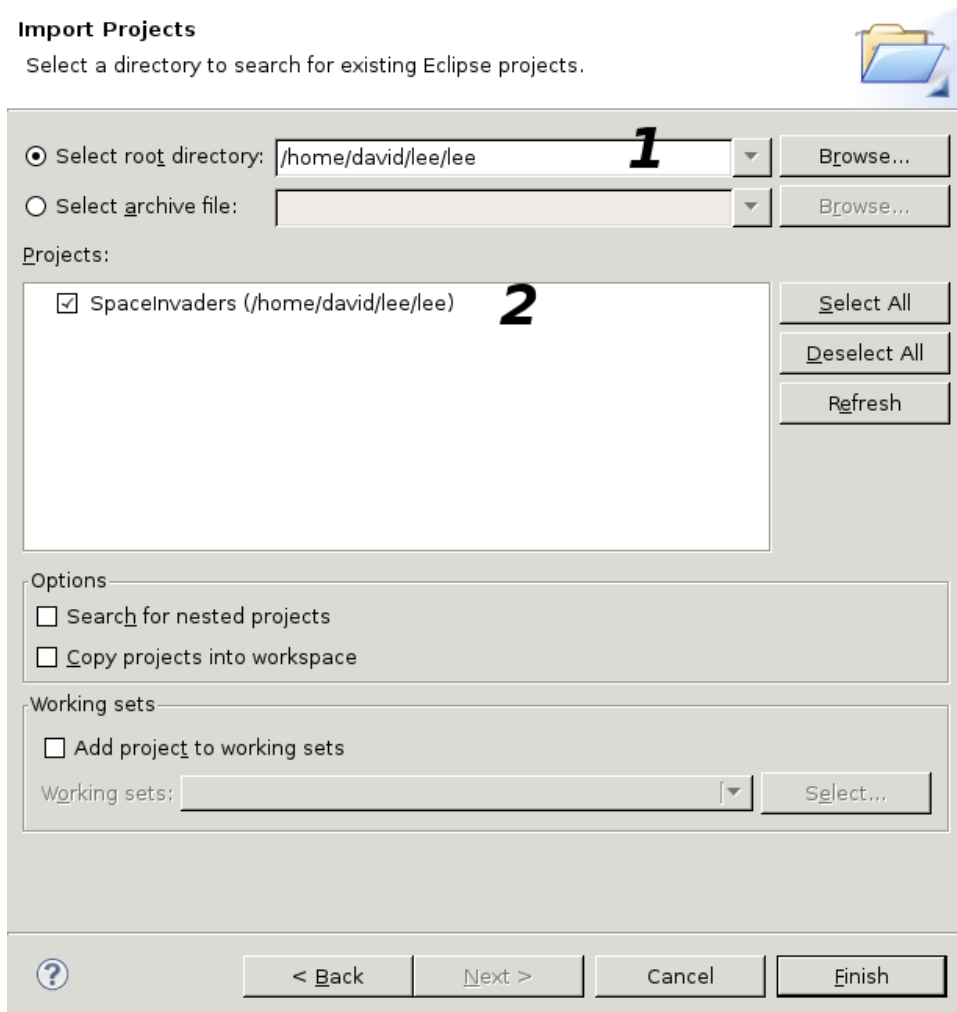
## Das Projekt importieren

Wenn Ihr Projektpartner das Projekt in Eclipse angelegt hat und auf das Repository gepushed hat, können Sie das Projekt runterladen, in dem Sie normal ein **git pull** ausführen. Danach können Sie das Projekt in Eclipse *importieren*. Das kann über **File > Import...** gemacht werden.

**Achtung:** Erstellen Sie nicht ein neues Projekt, sonst haben Sie zwei verschiedene Projekte, anstatt dass Sie am gleichen Projekt arbeiten!



Im Wizard wählen Sie **Existing Projects into Workspace** aus.

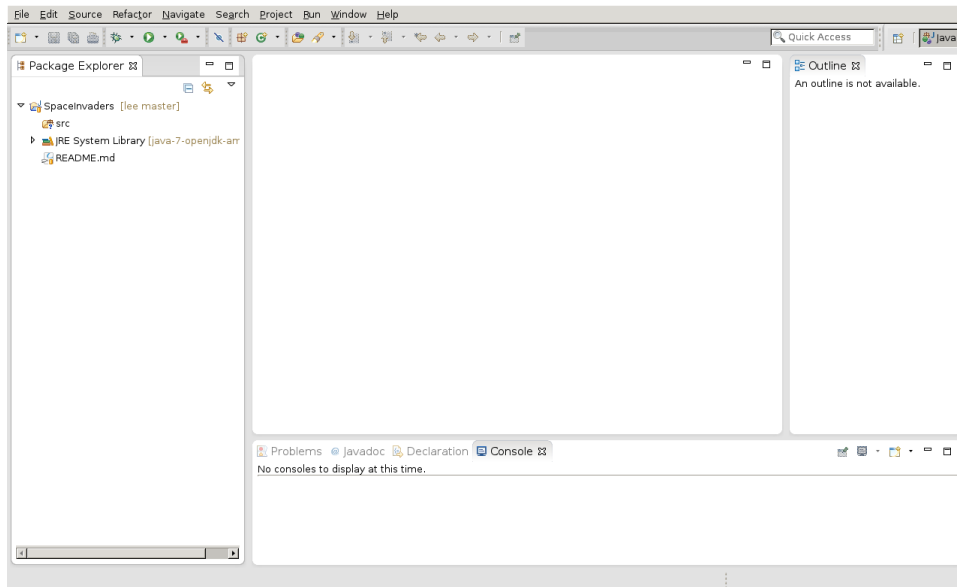


- 1 Wählen Sie das Verzeichnis aus, in welches Sie Ihr Repository geklont haben. (Wenn Sie ein Ordner innerhalb Ihres Repository gemacht haben, um mehrere Projekte im gleichen Repository zu haben, so wählen Sie dieses Verzeichnis aus.)
- 2 Stellen Sie sicher, dass das Projekt hier angezeigt wird und dass Sie es ausgewählt haben.

Nun haben Sie das Projekt ebenfalls im Eclipse.

## Erstes Programmieren in Eclipse

Wenn Sie jetzt den **Package Explorer** in Eclipse anschauen, sehen Sie Ihr Projekt. Sie sehen ein Verzeichnis namens **src**. Hier hinein gehört der *Source Code*, d.h. diejenigen Dateien, welche Sie im Verlauf der Entwicklung erstellen und welche Code beinhalten.

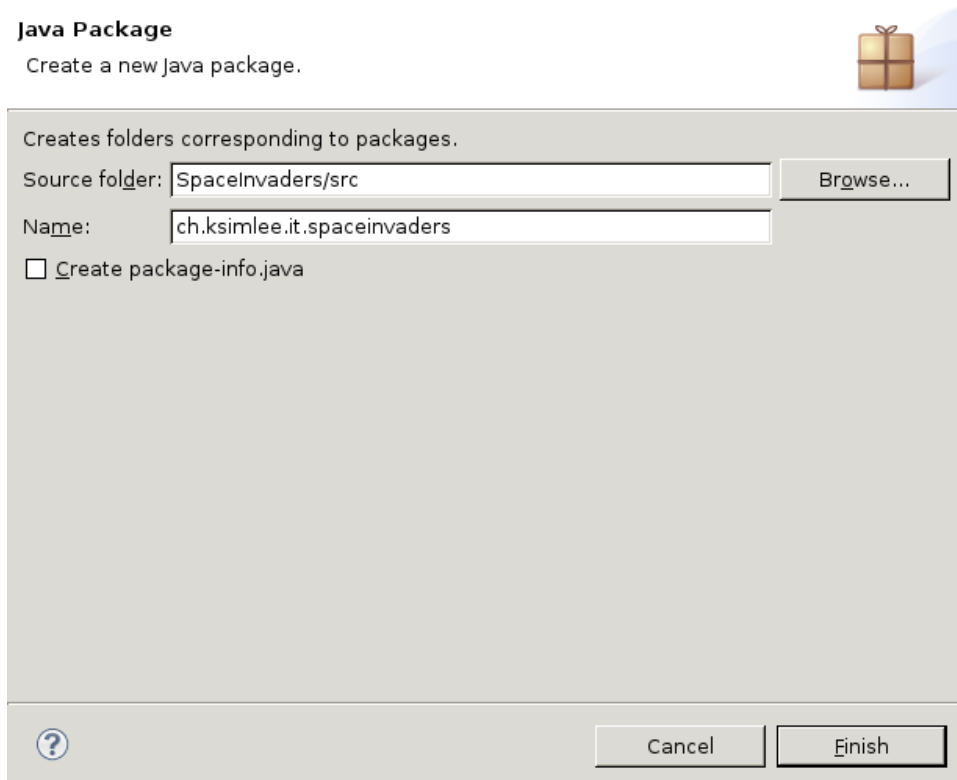


Damit der Source Code besser geordnet ist, und damit nicht alles in einem einzigen Verzeichnis gespeichert ist, gibt es bei Java sogenannte **packages**. Auf dem Dateisystem sind packages einfach Verzeichnisse, d.h. sie können auch verschachtelt werden.

Sie sollten immer innerhalb eines packages arbeiten. Deshalb erstellen wir nun ein erstes package. Bei Java gibt es eine Namenskonvention für packages:

1. Packages bestehen nur aus Kleinbuchstaben.
2. Packages identifizieren den Author und das Projekt, auf ähnliche Weise wie eine URL, aber rückwärts. Siehe nachfolgendes Beispiel.

Um ein package zu erstellen machen wir einen Rechtsklick auf das **src** Verzeichnis und wählen **New > Package**.



Als Namen habe ich hier **ch.ksimlee.it.spaceinvaders** gewählt. Der Punkt erstellt die Verschachtelung, und die einzelnen Abschnitte sind wie folgt gewählt:

**ch** Das Projekt wurde in der Schweiz erstellt, hat die gleiche Bedeutung wie “.ch” bei Domain Namen.

**ksimlee** Das Projekt wurde an der KS im Lee erstellt.

**it** Das Projekt wurde innerhalb der KS im Lee im Bereich Informatik (“IT”) erstellt.

**spaceinvaders** Der Name des Projektes.

Die Absicht hinter dieser Namensgebung ist, dass der gewählte Name einzigartig ist, und nicht verschiedene Leute den gleichen Namen für verschiedene Projekte verwenden. Das könnte z.B. passieren, wenn ich nur “spaceinvaders” als Projektname gewählt hätte.

Das Package wurde jetzt erstellt, ist aber noch leer.

## Eine erste Klasse erstellen

Jetzt, da wir unser Projekt (inklusive package!) vorbereitet haben, können wir eine erste Klasse erstellen. Source Code ist in Java (fast) immer in Klassen (Englisch: Class), denn Klassen sind das zentrale Konzept in Java. Eine Klasse bündelt Funktionalität, welche zusammen gehört.

Wir machen einen Rechtsklick auf unser package, und wählen **New > Class**.

**Java Class**  
Create a new Java class.

Source folder:  **Browse...**

Package:  **1** **Browse...**

☐ Enclosing type:  **Browse...**

Name:  **2**

Modifiers: ☒ public ☐ default ☐ private ☐ protected  
☐ abstract ☐ final ☐ static

Superclass:  **Browse...**

Interfaces:  **Add...**  
**Remove**

Which method stubs would you like to create?

☒ `public static void main(String[] args)` **3**

☐ Constructors from superclass

☒ Inherited abstract methods

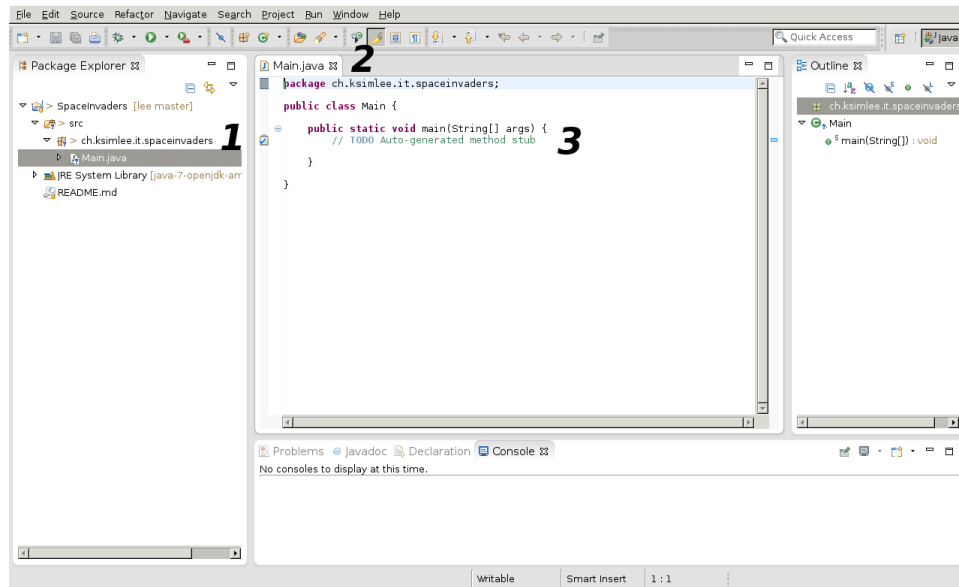
Do you want to add comments? (Configure templates and default value [here](#))

☐ Generate comments

**?** **Cancel** **Finish**

- 1** Hier sehen Sie, in welchem package die Klasse erstellt wird. Das sollte das package sein, welches wir im letzten Schritt erstellt haben.
- 2** Hier können Sie den Namen der Klasse angeben. Da wir die “Haupt”-Klasse erstellen, also diejenige, welche später unser Programm startet, nenne ich sie **Main**.  
**Beachte:** Klassen Namen sollten mit einem Grossbuchstaben anfangen.
- 3** Jedes Programm hat eine Startfunktion, welche beim Start des Programms ausgeführt wird. Wenn wir diesen Haken setzen, wir die entsprechende Funktionssignatur in der neuen Klasse erstellt.  
**Beachte:** Jedes Programm, d.h. jedes Projekt, braucht nur eine Startfunktion!





1 Die erstellte Klasse namens **Main** ist im Package Explorer zu finden. Sie kann jederzeit durch einen Doppelklick geöffnet werden.

2 Die Klasse **Main** ist momentan geöffnet. Hier kann der source code geändert werden.

3 Diese Funktion (mit dem Namen **main**) wurde von Eclipse für uns erstellt.

**Beachte:** Die Startfunktion muss immer **main** heissen, unabhängig vom Namen der Klasse!

, wenn wir es ausführen!

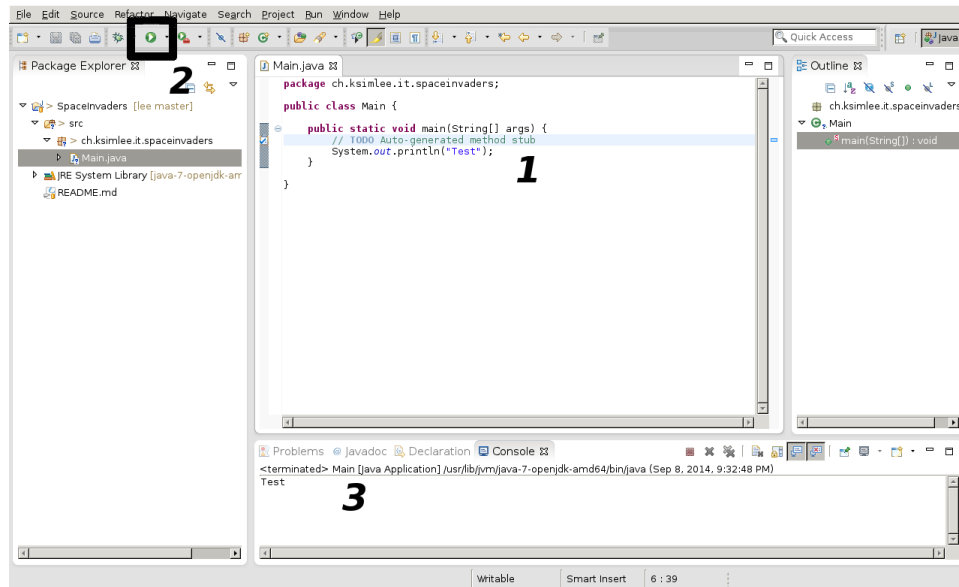
Die erste Klasse wurde nun angelegt, doch da die Funktion noch nichts macht, macht auch unser Programm nichts, wenn wir es ausführen!

## Der erste Befehl

Damit unser Programm etwas macht, können wir eine einfache Ausgabe auf der Konsole hinzufügen. Um in Java etwas auf der Konsole auszugeben, gibt es den folgenden “Befehl” (Sie werden später noch verstehen wie der “Befehl” zu Stande kommt):

Listing 1: Dieser Befehl gibt den String “Test” auf der Konsole aus.

```
System.out.println("Test");
```



- 1 Hier haben wir den Befehl zur Ausgabe des Strings “Test” zur **main** Funktion hinzugefügt.
- 2 Mit diesem Button können Sie Ihr Programm ausführen.
- 3 Wenn Sie das Programm ausführen, sehen Sie hier in der Konsole die Ausgabe. Hier sehen Sie den String “Test”.