Eidgenössische
Technische Hochschule
Zürich

Ecole polytechnique fédérale de Zurich
Politecnico federale di Zurigo
Federal Institute of Technology at Zurich

Departement of Computer Science
Markus Püschel, David Steurer

Technical Guide

Algorithms und Data Structures

D-INFK

This technical guides illustrates the process of setting up the Eclipse environment for code development required for the practical programming exercises and their submission to the Judge, the automatic online evaluation system. It provides step by step instructions using a sample *Hello Judge* exercise.

1 The Hello Judge Exercise

Exercise *Hello Judge*

Write a program such that reads from the standard input, and writes to the standard output, such that:

- If the input is "`1`", then it outputs "`Hello World!`"
- If the input is "`2`", then it outputs "`Hello ETH!`"
- If the input is "`3`", then it outputs "`Hello Judge!`"

Input The input consists of a single number of the set $\{1, 2, 3\}$

Output The output consists of a single line, followed by an end-line character.

Grading This is not a graded exercise, and delivers no points. Submit your solution as `Main.java` at `https://judge.inf.ethz.ch/team/websubmit.php?cid=28784&problem=AD18TUT`. The enrolment password is "`asymptotic`" (excluding the quotation marks).

Example

Input:

```
1
```

Output:

```
Hello World!
```

Notes For this exercise we provide a zip bundle available at `https://www.cadmo.ethz.ch/education/lectures/HS19/DA/uebungen/AD18TUT.Hello.Judge.zip` that defines the program template that will load the input and write the output for you. The archive also contains additional test cases (which might differ from the ones used for grading). Importing other classes is **not allowed** (with the exception of the already imported `java.util.Scanner` class).

2 Setting up Eclipse

Step 1. Download the programming template, and unzip the code. Then launch Eclipse.

Figure 1: Eclipse logo, shown at startup of the IDE. We assume the latest Eclipse version i.e. 2019–09 at the time of writing this guide, however other Eclipse version might be compatible too.

Step 2. Once Eclipse is started, import the programming template into the IDE, using **File** → **Import**, as illustrated on Figure 2.

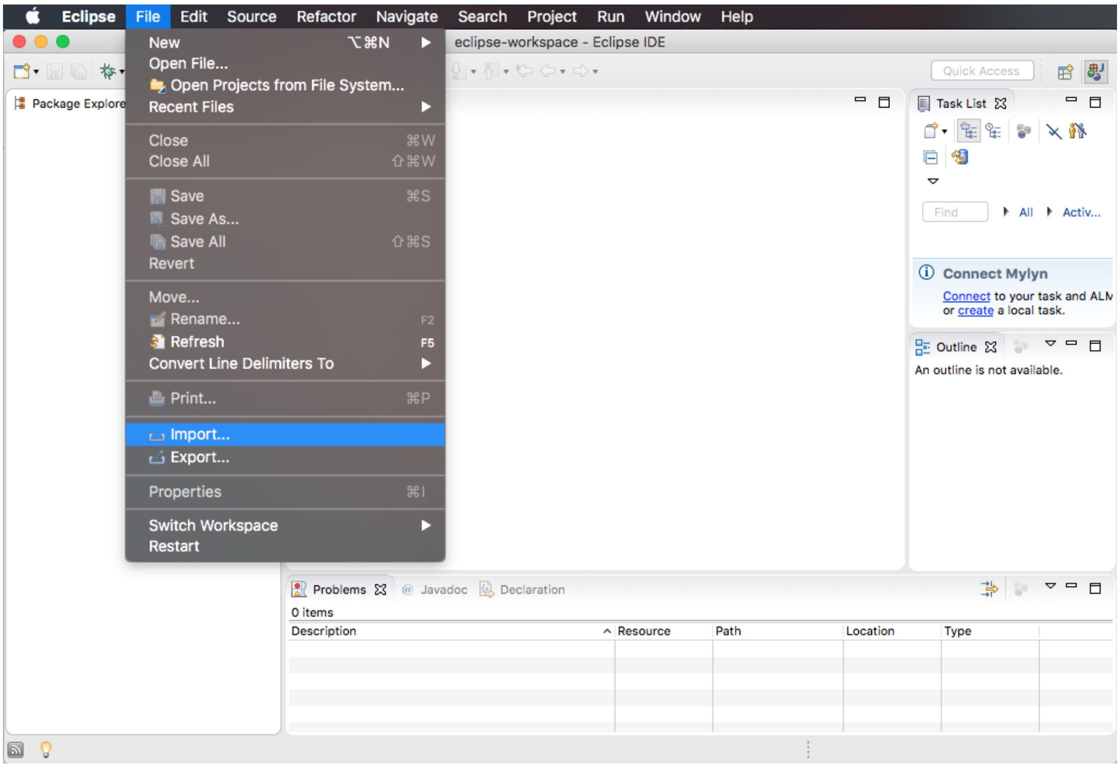
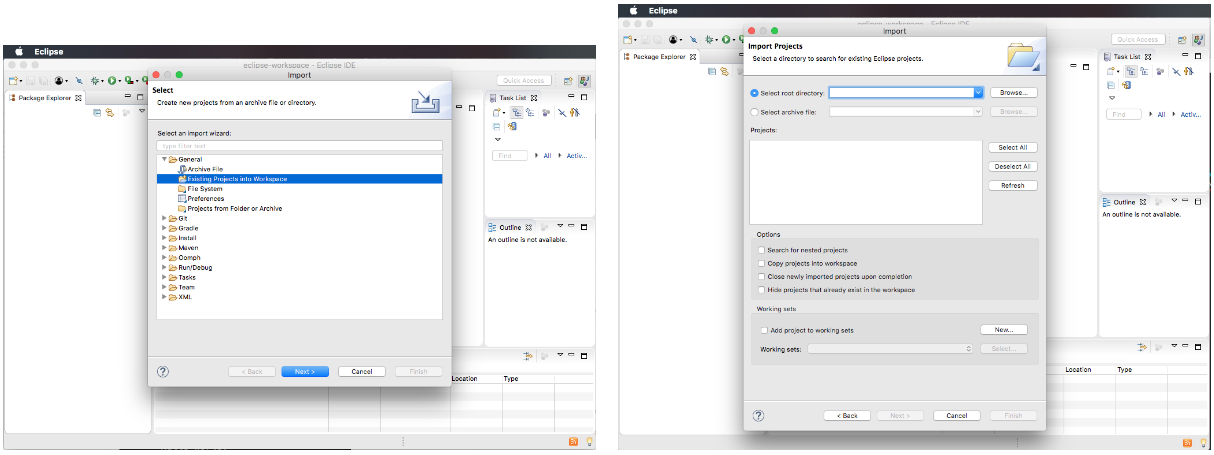


Figure 2: Importing the template into Eclipse (Part 1)

Step 3. Click on the **General** section, and choose **Existing Project into Workspace**. Then click **Next** (Figure 3a). In the next window, click on **Browse**, in order to **Select the root directory** (Figure 3b).



Import

Select

Create new projects from an archive file or directory.

Select an import wizard:

- General
  - Archive File
  - Existing Projects into Workspace
  - File System
  - Preferences
  - Projects from Folder or Archive
- Git
- Gradle
- Install
- Maven
- Oomph
- Run/Debug
- Tasks
- Team
- XML

Import Projects

Select a directory to search for existing Eclipse projects.

● Select root directory:

○ Select archive file:

Projects:

Options

□ Search for nested projects

□ Copy projects into workspace

□ Close newly imported projects upon completion

□ Hide projects that already exist in the workspace

Working sets

□ Add project to working sets

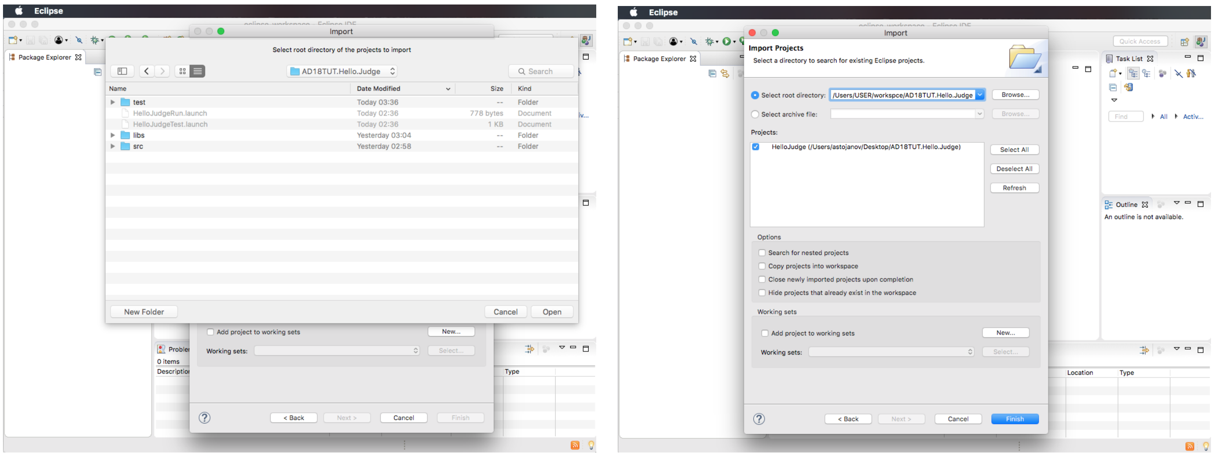
Working sets:

(a) Importing an Existing Project into Eclipse

(b) Select root directory

Figure 3: Importing the template into Eclipse (Part 2)

Then navigate to the folder of the unzipped Hello Judge template, and then click **Open** (Figure 3a). Once the file system address is defined in the "Select toot directory" field, click on **Finish**.



Select root directory of the projects to import

AD18TUT.Hello.Judge

Name	Date Modified	Size	Kind
test	Today 03:36	--	Folder
HelloJudgeRun.launch	Today 02:36	778 bytes	Document
HelloJudgeTest.launch	Today 02:36	1 KB	Document
libs	Yesterday 03:04	--	Folder
src	Yesterday 02:58	--	Folder

Import Projects

Select a directory to search for existing Eclipse projects.

● Select root directory: /Users/USER/workspce/AD18TUT.Hello.Judge

○ Select archive file:

Projects:

☑ HelloJudge (/Users/astojanov/Desktop/AD18TUT.Hello.Judge)

Options

□ Search for nested projects

□ Copy projects into workspace

□ Close newly imported projects upon completion

□ Hide projects that already exist in the workspace

Working sets

□ Add project to working sets

Working sets:

(a) Importing an Existing Project into Eclipse

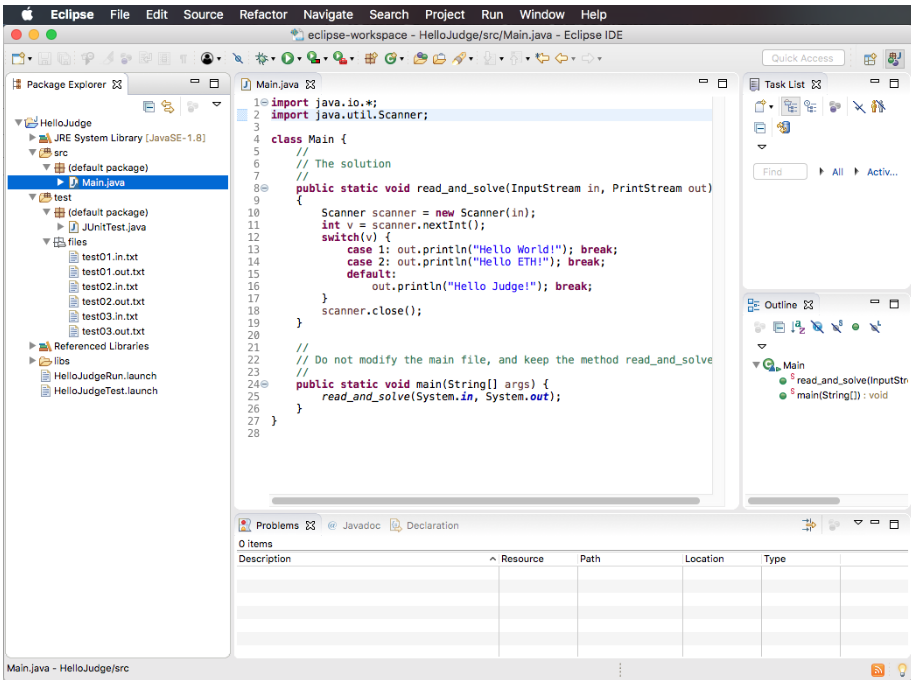
(b) Select root directory

Figure 4: Importing the template into Eclipse (Part 3)

At this point in time, we have successfully imported the template into Eclipse. Looking at the **Project Explorer** tab, we can observe that the code is organized such that we have a `Main.java` file located into the `src` folder. To illustrate the usage of the template, we also provide the solution (Figure 5).

Note that each exercise solution must be completed by writing code into the `Main.java` file only, such that no other file is created. The other files in the template skeleton should not be modified, and are provided for your convenience.

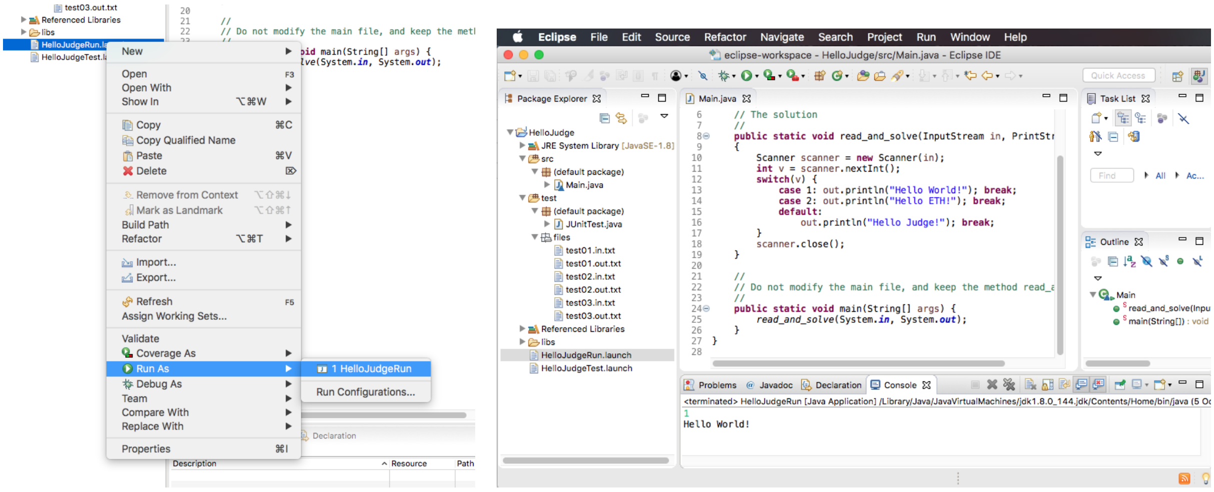
In `test/files` we have 3 input files, that end with the `.in.txt` extension, and 3 output files that end with the `.out.txt` extension. The input files, define sample test cases for your program, while the output files define the expected output. The `JUnitTest.java` uses the JUnit framework to simplify and automate testing on the given sample input / output test cases.



```java
import java.io.*;
import java.util.Scanner;

class Main {
    //
    // The solution
    //
    public static void read_and_solve(InputStream in, PrintStream out)
    {
        Scanner scanner = new Scanner(in);
        int v = scanner.nextInt();
        switch(v) {
            case 1: out.println("Hello World!"); break;
            case 2: out.println("Hello ETH!"); break;
            default:
                out.println("Hello Judge!"); break;
        }
        scanner.close();
    }

    //
    // Do not modify the main file, and keep the method read_and_solve
    //
    public static void main(String[] args) {
        read_and_solve(System.in, System.out);
    }
}
```

Figure 5: The solution to the *Hello Judge* exercise and file structure organization

Step 4. To run the code of the *Hello Judge* program, we right click on the `HelloJudge.launch` file → **Run as** → **HelloJudgeRun** (Figure 6a). We observe that once we input "1" (followed by a new line), the program will output "`Hello World!`" in the **Console** tab and the execution of the program will be completed (Figure 6b). In case the **Console** tab is not shown in Eclipse, click on **Window** → **Show View** → **Console** to enable it.



```
<terminated> HelloJudgeRun [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_144.jdk/Contents/Home/bin/java (5 Oc
1
Hello World!
```

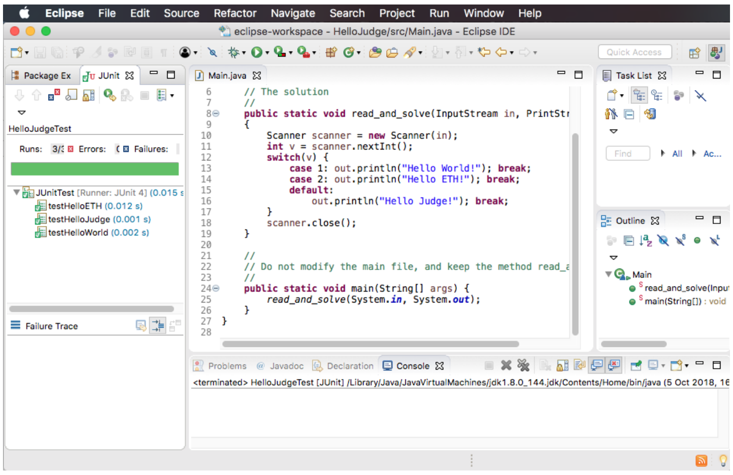
(a) Run *Hello Judge*

(b) Execution Completed

Figure 6: Running *Hello Judge* program

Note that you can manually test correctness and execution time of your program by copy pasting the contents of the input files, and manually comparing the output with the output files.

Step 5. In order to automatically test the correctness and execution time of your program, we right click on `HelloJudgeTest.launch` file → **Run as** → **HelloJudgeTest**. Once the program finishes, and all test are completed, and correct, we should observe the following:



```
HelloJudgeTest
Runs: 3/3  Errors: 0  Failures:
JUnitTest [Runner: JUnit 4] (0.015 s
    testHelloETH (0.012 s)
    testHelloJudge (0.001 s)
    testHelloWorld (0.002 s)
```

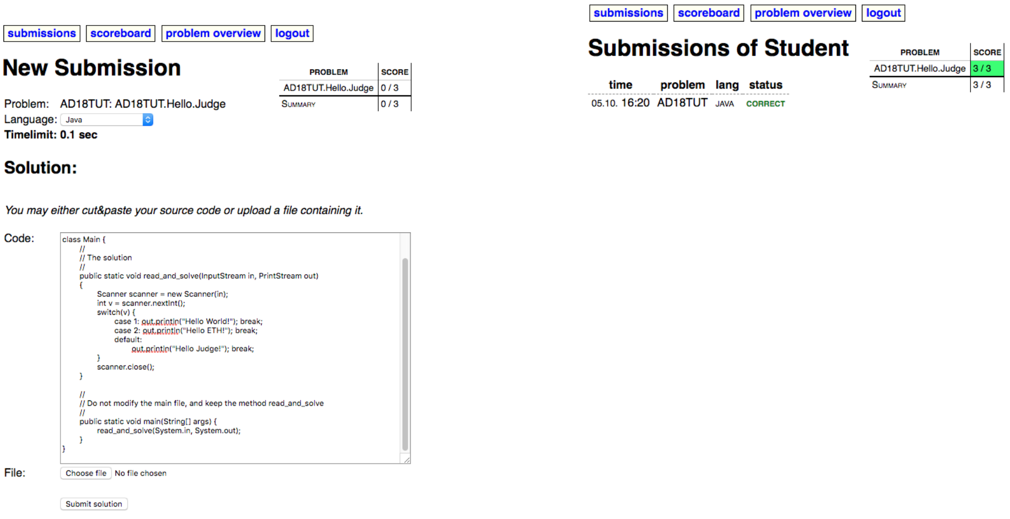
```
    // The solution
    //
    public static void read_and_solve(InputStream in, PrintStr
    {
        Scanner scanner = new Scanner(in);
        int v = scanner.nextInt();
        switch(v) {
            case 1: out.println("Hello World!"); break;
            case 2: out.println("Hello ETH!"); break;
            default:
                out.println("Hello Judge!"); break;
        }
        scanner.close();
    }

    //
    // Do not modify the main file, and keep the method read_a
    //
    public static void main(String[] args) {
        read_and_solve(System.in, System.out);
    }
}
```

```
<terminated> HelloJudgeTest [JUnit] /Library/Java/JavaVirtualMachines/jdk1.8.0_144.jdk/Contents/Home/bin/java (5 Oct 2018, 16
```

Figure 7: Testing *Hello Judge* using JUnit

3 Submitting to the Judge

Navigate to the exercise URL at `https://judge.inf.ethz.ch/team/websubmit.php?cid=28784&problem=AD18TUT` and login using your `netz` credentials. If necessary, enter the enrolment key "`asymptotic`" (excluding the quotation marks). Select the language to be `Java`, and then either copy paste the code into the text-area, or choose the `Main.java` file located in the `src` folder (Figure 8a).



New Submission

PROBLEM	SCORE
AD18TUT.Hello.Judge	0 / 3
Summary	0 / 3

Problem: AD18TUT: AD18TUT.Hello.Judge
Language: Java
Timelimit: 0.1 sec

Solution:

You may either cut&paste your source code or upload a file containing it.

Code:

```
class Main {
    //
    // The solution
    //
    public static void read_and_solve(InputStream in, PrintStream out)
    {
        Scanner scanner = new Scanner(in);
        int v = scanner.nextInt();
        switch(v) {
            case 1: out.println("Hello World!"); break;
            case 2: out.println("Hello ETH!"); break;
            default:
                out.println("Hello Judge!"); break;
        }
        scanner.close();
    }

    //
    // Do not modify the main file, and keep the method read_and_solve
    //
    public static void main(String[] args) {
        read_and_solve(System.in, System.out);
    }
}
```

File: No file chosen

(a) Submitting code *Hello Judge*

Submissions of Student

time	problem	lang	status
05.10. 16:20	AD18TUT	JAVA	CORRECT

PROBLEM	SCORE
AD18TUT.Hello.Judge	3 / 3
Summary	3 / 3

(b) Results

Figure 8: Using the Judge

Finally, click **Submit**. Once the program is evaluated on our servers, the results will be displayed (Figure 8b). Note that it might take few seconds up to few minutes until results are displayed.