Software Engineering SS18

Course Exercises and Material

April 16, 2018

1 Introduction

In the following sections, you will learn how to edit, run and test Java code. We will introduce several tools to help you with these tasks, namely Eclipse, git, and JUnit. These are tools commonly used by developers both in academia and in industry, so your familiarity with them will be a valuable skill.

Note, for future assignments it is not necesarry that you use these specific tools, or even any of them, however utilizing the tools introduced here should make the assigments easier.

2 Running eclipse

Eclipse is an open source integrated development environment (IDE) for Java and other programming languages. A version of eclipse should already be found on the computers on RTK. Simply run `eclipse` as a command from the terminal to start the program. If you want to use your personal computer and you do not have eclipse installed, the newest version can be downloaded from: `https://www.eclipse.org/downloads/`. Be aware that the newest version of eclipse requires java 8, while only java 7 is installed on the comuters in RTK.

3 Git

Git is a version control system for tracking changes of files and coordinating work between multiple people. For a quick tutorial for how to use git from the command line see: `https://git-scm.com/docs/gittutorial`.

In the final project of this course it will be expected of you to work together in groups of 2-3 people. To this end, multiple people will work on the same code project. To avoid people modifying the same files, and easier manage and keep track of your work, you might benefit from using something like git in this course.

To this end, you should have a server that all of you can access that contains your code base, or repository. One way to get accesss to such a server is vie

https://github.com/ which is widely used among students and developers to store their repositories. Be aware, that if you create a free public account all repositories will be pubilicly available for everyone to see, which you might want to avoid if you use github for your git repository for your final project. As a student, it is however easy to gain access to unlimited private repositories by signing up for a student account following the instructions specified at: https://help.github.com/articles/applying-for-a-student-developer-pack/.

For tutorials how to use git(hub) with eclipse see: http://www.vogella.com/tutorials/EclipseGit/article.html and http://www.vogella.com/tutorials/EclipseGit/article.html#using-eclipse-git-with-github. For now however, we will simply be working with no version control.

4 Working with java in eclipse

If everything works as expected, download the tar.gz file at https://math-inf.uni-greifswald.de/institut/ueber-uns/mitarbeitende/hellmuth/teaching/softwaretechnik/, containing a directory with java files that will help you get started with learning how to use Java in eclipse. After downloading and unpacking the file, we need to import it as a project in eclipse. To do this, first create a new empty java project (File » New » Java Project). Then, right click the source folder and choose Import » General » File System, and choose the directory with your unpacked source files. If done correctly, the project and source files should show up in your Package Explorer.

Notice that the project does not compile. Specifically, there are compilation errors in HejWorld.java that you will need to correct. To help you fix the program, observe that eclipse marks the problematic lines by red squigly underlines, and the specific linenumber is marked with a red X on the left side.

Use these visual aides to help find the compilation errors in your code. (You can also find a list of all of the compilation errors in your project by looking at the Problems tab at the bottom of the window. If the Problems tab is not visible, then you can open it again by doing Window » Show View » Problems.)

Once you have found a compilation error, you can move the mouse over it and a description of the error will pop up as a tooltip. You can also mouse over either of the red icons in the margin of the Java editor to get this information.

Once you have fixed the broken HejWorld.java file, rightclick on the file and select Run As » Java Application. If you ran HejWorld successfully, you will see both English and Danish greetings in the Console window.

Autocompletion: Using autocompletion will reduce the amount of typing that you have to do as well as the number of spelling mistakes, thereby increasing your efficiency. In Eclipse, CTRL+Space can be used to autocomplete most things inside the Java editor. For example, place the cursor at the right of danishGree and then press CTRL+Space. You should see danishGree expand to danishGreeting. Then press CTRL+S to save the file. You will see that the compilation error disappears upon saving your changes. CTRL+Space can

also help you autocomplete method names. Again, place the cursor to the right of world on line 26. Eclipse knows that world is of type HejWorld, so when you press CTRL+Space after the period, it pops up a list of methods that HelloWorld has. Since we want to use the method sayHello(), type **s** to start writing the method name as you normally would. Now the list of available methods has been reduced to one item because sayHello() is the only method in HelloWorld that starts with s. Now with sayHello() as the only option in the list, you can press Enter to complete the method name.

5 Warm-Up Exercise – RandomHello

For this exercise we will create a java class that when run, will randomly choose one of five possible greetings to print to the console. The content of the greetings are up to you.

First create a new class. Select from the top menu File » New » Class. A window will pop up, asking you details about the class. Type the name RandomHello in the "Name" field and click Finish.

In order to run RandomHello, you need to give it a main method whose signature is public static void `main(String[] argv)` (which is identical to the signature of the main method in HelloWorld and HejWorld). When creating RandomHello, you can check the appropriate checkbox and have Eclipse create a stub of the main method for you.

Alternatively, you can use eclipse autocompletetion to make a main method, or use the following skeleton code:

```
/**
 * RandomHello selects a random greeting to display to
 * the user.
 */
public class RandomHello {

 /**
 * @effects uses a RandomHello object to print
 * a random greeting to the console
 */
 public static void main(String[] argv) {
   RandomHello randomHello = new RandomHello();
   System.out.println(randomHello.sayHello());
 }

 /**
 * @return a random greeting from a list of
 * five different greetings.
 */
 public String sayHello() {
 // YOUR CODE GOES HERE
```

```
 }
}
```

Random Number Generator and importing classes: To generate a random number, you should take adavantage of the Java class Random.

Type the following into the body of your `sayHello()` method:

```
Random randomGenerator = new Random();
```

This line creates a random number generator. In Eclipse, your code may be marked with a red underline, indicating an error. This is because the Random class lies in a package that has not yet been imported

Java libraries are organized as packages and you can only access Java classes in packages that are imported (this is similar to #include statements in C++). To explicitly import `java.util.Random` and avoid the compilation error, you can add the following line at the very top of your file:

```
import java.util.Random;
```

The above line will import the class Random into your file. If you are using Eclipse, you can also hit CTRL-SHIFT-O to organize your imports. Because there is only one class named Random, Eclipse can figure out that you mean to import java.util.Random and will add the above line of code automatically.

Viewing documentation for source code: You can also use Eclipse to view the documentation for available Java classes, which is automatically generated from comments in Java source code using a tool called javadoc. You will learn about how to generate your own javadoc files later in the course. For now, we will show you how to view them in Eclipse. The first time you do this however, you need to tell Eclipse where to look for the external Java documentation files (You will only have to do this once). To accomplish this do the following:

- Switch to the Java Browsing perspective by choosing Window » Perspective » Open Perspective » Other... » Java Browsing.
- In the Projects window, expand Assignment0 to find Assignment0/rt.jar. Right-click on rt.jar, and choose Properties.
- For the Javadoc location, enter `https://docs.oracle.com/javase/8/docs/api/` (or `https://docs.oracle.com/javase/7/docs/api/` if your are using the RTK computers), and press OK. (This information may already appear, if so, just click OK.)
- Now return to the Java Perspective (Window » Open Perspective » Java), and go back to editing RandomHello. Place your cursor on Random, and press SHIFT-F2. A web browser window will appear, showing documentation for the Random class.

To configure Eclipse to recognize the Java documentation files for source code in our current project, right click on the project name ("Assignment0") in the Package Explorer pane and click "Properties". Select "Javadoc Location" in the left pane. You can either type in a location or click "Browse..." to browse to the desired directory you want to put your documentation. Here, we recommend you create a folder named "doc" in the current project directory, and use this folder as your Javadoc location path. After setting the Javadoc location path, click OK.

It is now possible to generate Javadoc documentation for your source code by clicking on the Project pull-down menu at the top and choosing Generate Javadoc. This opens a window that allows you to customize what kind of documentation is generated. You can simply click on Finish.

Using `java.util.Random` Find the documentation for `nextInt(int n)` on the Random class and read it. You do not have to understand all the details of its behavior specification, only that it returns a random number from 0 to $n-1$. You should use this method to choose your greeting. One way to choose a random greeting is using an array. This approach might look something like:

```
String[] greetings = new String[5];
greetings[0] = "Hello World";
greetings[1] = "Hola Mundo";
greetings[2] = "Bonjour Monde";
greetings[3] = "Hallo Welt";
greetings[4] = "Ciao Mondo";
```

Looking at the main method, the `System.out.println()` method prints a String to the console. In this case the String is provided by the method `sayHello()` which returns one of five Strings chosen randomly from the `greetings[]` array. See `HelloWorld.sayHello()` for a simple example of how to return a String. If you haven't used Java before, or are having trouble with the language or syntax, a good reference is the Java Tutorial from Sun's website. When you are finished writing your code and it compiles, run it several times to ensure that all 5 greetings can be displayed.

6 Unit Testing Java Code With JUnit

Part of the process of software engineering is to verify that the program you are implementing work according to its specification. One form of verification is unit testing. JUnit is a framework for creating unit tests in Java. A unit test is a test for verifying that a specific method in a class conforms to its specifications. You will learn more about unit testing later in the course. In this section, we will provide you with a quick overview of how JUnit works with a simple example.

Open both `Fibonacci.java` and `FibonacciTest.java`. From the comments, you can see that FibonacciTest is a test of the Fibonacci class. In the Package Explorer, right-click on FibonacciTest.java and select Run As » JUnit Test.

Upon running JUnit, you should be notified that not all tests were successful. The top pane displays the list of tests that failed, while the bottom pane shows the Failure Trace for the highlighted. The first line in the Failure Trace should display an error message that explains why the test failed (it is the responsibility of the author of the test code to produce this error message).

As shown in the figure above, if you click on the first failure `testThrowsIllegalArgumentException()`, the bottom pane will automatically switch to the appropriate error message. You can see from the first line of the failure trace that Fibonacci.java threw an IllegalArgumentException for the argument zero, when it shouldn't have thrown any exception (you will have to scroll the pane to the right to see this). Once you've figured out the problem in Fibonacci.java that caused this error and fixed it, you can rerun the JUnit test by clicking on the green icon with the arrow pointing to the right in the JUnit pane (just above "Result"). Use the information in the Failure Trace box to help you debug Fibonacci. After you have fixed all the problems in Fibonacci, you should see a bright green bar instead of a red one when you run FibonacciTest. Note: Now that you have a few files open, try holding down Ctrl and hitting F6 to bring up a dropdown box of open files. You can use the arrow keys to select the file whose editor you wish to bring into focus.

Now that you have seen an example how to create unit tests for a class, try and create unit tests for the HejWorld class, by creating a new class called HejWorldTest and following the example of `FibonacciTest.java`.

7 Balls and Boxes

Until now we have only introduced tools that will hopefully make your programming experience with Java more pleasant. In this section, we will focus on actual programming. Again, if you are not at all familiar with Java, it is recomended read Sun's Java Tutorial first.

The intention of this problem is to give you a sense of what Java programming entails and to demonstrate the use of Eclipse and the JUnit testing tool. If you have never programmed in Java. you may find this problem somewhat challenging. Do not be discouraged, if you spend effort getting this problem right now, you will likely have less trouble later in the course.

7.1 Creating a Ball

As a warm up exercise, take a look at `Ball.java`. A Ball is a simple object that has a capacity.

- Notice that eclipse produces warnings in the file. What is wrong with `Ball.java`?

If you pay attention to the warnings in Eclipse, you should be able to find at least one of the bugs without referring to the JUnit results.

7.2 Using Pre-Defined Data Structures

Next, we want to create a class called BallContainer. As before, skeleton code is provided (see `BallContainer.java`). A BallContainer is a container for Balls. BallContainer must support the following methods and your task is to fill in the code that will implement all these methods correctly:

1. `add(Ball)`
2. `remove(Ball)`
3. `getCapacity()`
4. `size()`
5. `clear()`
6. `contains(Ball)`

One of the nice things about Java is that it has many libraries and pre-defined data structures that you can simply use without have to write your own from scratch. One of the intentions of this problem is to expose you to the use of pre-defined Java datastructures. In BallContainer, we use a `java.util.LinkedList` to keep track of the balls. If you open `BallContainer.java` in Eclipse, you will notice that there is a warning message associated with the line:

```
contents = new LinkedList();
```

It should say something about an "Unsafe type operation". The Java Collections framework is strongly typed so instead of defining a generic LinkedList, you should define a typed LinkedList. Please modify this line to remove this warning for the constructor statement. Before you proceed to implement the required method, please take a moment to read through the documentation for LinkedList. Some of the methods that you are required to implement simply require you to call the appropriate predefined methods for LinkedList.

Hint Place your cursor on LinkedList, and press SHIFT-F2. A web browser window will appear, showing documentation for the LinkedList class.

Most of the methods that you are required to implement are quite simple. Before you start coding, please take time to think about the following questions:

- There are two obvious approaches for implementing `getCapacity()`:
  - Every time `getCapacity()` is called, go through all the Balls in the LinkedList and add up the capacities. (Hint: If you choose this approach, you will probably want to use an Iterator to extract Balls from the LinkedList. You can see an example of how Iterator is used in `BoxTest.java`.)
  - Keep track of the total capacity of the Balls in BallContainer whenever Balls are added and removed. This obviates the need to perform any computations
- Which approach do you think is the better one? Why?

7.3 Implementing Algorithms

By the time you are done with BallContainer, you should be somewhat comfortable with Java, so in this problem, we want you to do a little more design and thinking and a little less coding. Your final task in this problem is to create a class called Box. A Box is also a container for Balls. The key difference between a Box and a BallContainer is that a Box has only finite capacity. Once a box is full, we cannot put in more Balls. The size (capacity) of a Box is defined when the constructor is called:

```
public Box(double capacity);
```

Since a Box is really a BallContainer with some extra properties, it makes sense to say that in fact a Box is a type of BallContainer, which is why the Box is defined to extend BallContainer. This is called *Inheritance*. If you have never hear about this before, do not worry; you will learn more about this later in the course. For now, it suffices to know that what this means is that Box automatically has all the methods and properties of BallContainer. When you look in `Box.java`, you will realize that Box has only two methods defined in `Box.java`:

- `add(Ball)`
- `getBallsFromSmallest()`

Depending on your implementation of `getCapacity()` in our last problem, you may also need to implement a different `remove(Ball)` method for Box as well. If so, please do so. There is no need to change your implementation of BallContainer for this problem. You can also make other changes to Ball, BallContainer or Box. As before, your task is to implement these two methods.

Before you start working on `getBallsFromSmallest()` you may wish to check out the documentation on Iterator (Place your cursor on Iterator, and press SHIFT-F2.) Also, take some time to answer the following questions:

- There are many ways to implement `getBallsFromSmallest()`. Brainstorm with your buddy and come up with at least two ways. Briefly describe them.
- Which of the above ways do you think is the best? Why?

There is no single correct answer. The whole point of this exercise is help you fight that urge to code up the first thing that comes to mind, but instead spend a little more time thinking before you start coding.

8 Working with cards

By now you should hopefully be getting comfortable with using Java and eclipse. The next few exercises will give you more free reigns in how you solve the problem.

In the src folder there should be a package called deck. Inspect the classes `Card.java`, `CardSuit.java`, and `CardValue.java` and understand what each class is doing. When you feel like you you have a good grasp on what is going on, look at the file `MyHand.java`, and implement the methods specified there.

9 Primes

This next exercise will teach you one possible way to compare the peformance of several implementations by timing their runtime on certain inputs. Take a look at the file `Primes.java`, and understand how the method `findPrimes(int n)` find all primes from 2 to n. This method can easily be improved, and you should implement some improved approaches in the methods `findPrimesFaster(int n)` and `findPrimesEvenFaster(int n)`. When you believe you have implemented some faster methods, run the class and see if the methods you have implemented are indeed faster.

10 String Scrambler

This small exercise will teach you how to work with strings. In short, you need to implement the method `reverseWordOrder(String input)` found in the file `StringScrambler.java`. The method takes a string as input and should reverse the string.