1 Lab A1: Java and Eclipse

1.1 Hello World

Task A1.1: Go through the steps to create a Hello World application in Eclipse[1], as we did in class. Run the application.

1.2 Java Keyword Piles

Task A1.2: Print the cards with the Java keywords[2], and cut them out. Shuffle the cards and pile them up. Draw cards from the pile and assign them to one of the keyword groups. Discuss for each keyword group what it is used for.

Take a picture of the piles and include it in your report.

1.3 Looping Around

The following method contains a simple for-loop.

```
public static void loop1() {
    for(int i=0; i<5; i=i+1) {
        System.out.println(i);
    }
}
```

Hint: The statement `i=i+1;` increments the counter `i` by one. This statement can also be written as `i++`. So if you see it elsewhere, you know what it means.

Task A1.3: Before running the main method and calling `loop1()` in it, try to predict what you will see in the console window. Write down you prediction. After that, run the program. Compare.

Exchange the method call `System.out.println(i)` with `System.out.print(i)`. The method `println(i)` is documented here[3], and `print(i)` is documented here[4].

[1] eclipse.html
[2] https://dl.dropboxusercontent.com/u/830148/ttm4175/keyword-piles.pdf
[3] http://docs.oracle.com/javase/8/docs/api/java/io/PrintStream.html#println-java.lang.String-
[4] http://docs.oracle.com/javase/8/docs/api/java/io/PrintStream.html#print-java.lang.String-

Task A1.4: Read the documentation of these two methods, and try to predict what will happen. Then, run the program. Describe.

Have a look at the loop in method `loop2()`. Something very important is missing here. To run it, place a comment before `loop1()`and uncomment `loop2()`. But beware - the program will probably behave strangely and you have to cancel it.

```
public static void loop2() {
    int i=0;
    while(i<5) {
        System.out.println(i);
    }
}
```

Task A1.5: What is happening here? Which statement is missing? Make a screenshot of the debug view and include it in the report.

Hint: To abort a running program, you can click the red stop button in the console view. The grey x-buttons remove the output of a program after it terminated.

Figure 1: Alt text

1.4 Fibonacci Messed it up

The Fibonacci numbers[5] form the following series of numbers:

`0, 1, 1, 2, 3, 5, 8, 13, 21, 34, ...`

Starting with the numbers 0, 1, the following numbers in this series are calculated by adding together its two predecessors. (The series starts either with 0,1 or 1,1. In the following, we start with 0,1.) So, when the first two numbers are 0 and 1, the third is 0+1=1, and the fourth is 1+1=2, and so on.

[5] http://en.wikipedia.org/wiki/Fibonacci_number

Task A1.6: The sequence above shos the first 10 Fibonacci numbers. Which are the subsequent two numbers?

Write a function in Java that returns the n-th Fibonacci number. The method should have the following signature:

```
public static long fib(int n) {
    ...
}
```

In the template for this exercise we have included a main method. It calls the fib method with some pre-calculated values. It works like a test suite, by checking if your method calculates the correct numbers at least for some selected examples. Use it while you develop your implementation of `fib(int n)`.

Task A1.7 (Advanced) Write the contents of the method yourself. Within the method, use a for-loop and an integer variables to calculate the result. It may be necessary to treat the first two values separately. For that, you can use if-statements. If you manage to do this task, ignore task A1.8.

Task A1.8: Find the solution for task A1.7, by re-ordering given code lines. Simply assume Mr. Fibonacci has messed up his algorithm and you should sort it again. We have placed the code lines on this extra page[6].

Task A1.9 (Advanced): Given that you are using integers as variables, until which n can you calculate the numbers? What happens if you go above it?

Task A1.10 (Advanced): Another way to calculate the Fibonacci numbers is the following:

```
public static long fibRec(int n) {
    if (n <= 1) return n;
    else return fibRec(n-1) + fibRec(n-2);
}
```

This function `fibRec()` calls itself. Can you explain how this works?

1.5 Multiplication Table

Task A1.11: Write a program that has the following output:

[6] fibonacci.html

```
 * |  1  2  3  4  5  6  7  8  9
-------------------------------
 1 |  1  2  3  4  5  6  7  8  9
 2 |  2  4  6  8 10 12 14 16 18
 3 |  3  6  9 12 15 18 21 24 27
 4 |  4  8 12 16 20 24 28 32 36
 5 |  5 10 15 20 25 30 35 40 45
 6 |  6 12 18 24 30 36 42 48 54
 7 |  7 14 21 28 35 42 49 56 63
 8 |  8 16 24 32 40 48 56 64 72
 9 |  9 18 27 36 45 54 63 72 81
```

Obviously, the program should calculate the results of the multiplication, not just printing out a line like this:

```
...
System.out.println("3 |  3  6  9 12 15 18 21 24 27");
...
```