

Exercise 1 – Suggested Solution

Written by: Idan Felix (intro2cs.group1@idc.ac.il)

1.1 - Experimenting with errors

PrintSomeNumbers.java

```
public class PrintSomeNumbers {
    public static void main(String[] args) {
        int i = 0;
        while (i < 6) {
            System.out.println(i);
            i = i + 1;
        }
        System.out.println("Done");
    }
}
```

Changes - <http://www.youtube.com/watch?v=FeMKM-eQPb4>

For each type of change, it is important to classify the error, as well as understanding it – These errors are the first step in debugging a program, and understanding what is written and why will improve this skill a great deal.

Changing the class name, without changing the filename

This is a compile time error:

```
PrintSomeNumbers.java:2: class printSomeNumbers is public,
should be declared in a file named printSomeNumbers.java
```

The compiler is unwilling to compile the code, because the filename does not match the public class that is written inside it. In the file that I used, the class is defined in line 2 in `PrintSomeNumbers.java` – and that is where the compiler has caught this error – according to the first line of its output.

In java, public classes names must match their filenames, and since Java is case-sensitive, "PrintSomeNumbers" and "printSomeNumbers" is considered a mismatch.

Changing the printed string, Done to done

The code compiles without any problem, and the program executes.

The output is changed, and the last line now says "done" and not "Done".

This is not considered an error, but can be considered as a logical error.

Both answers are acceptable.

Removing the first quotation mark

This change causes a Compile Time Error:

```
PrintSomeNumbers.java:9: ')' expected
                        System.out.println(Done");
                                                ^
PrintSomeNumbers.java:9: unclosed string literal
                        System.out.println(Done");
                                                ^
PrintSomeNumbers.java:9: ';' expected
                        System.out.println(Done");
                                                ^
PrintSomeNumbers.java:11: reached end of file while parsing
}
^
4 errors
```

The compiler complains about an **"Unclosed String Literal"** and then has 3 consequent errors: a missing `)`, a missing `;` and about reaching the end of the file while parsing.

These characters are missing because the compiler treats them as the content of the string that is opened after the letter `e`. What's important to learn here is that the compiler's output is sorted by the order in which the errors are found, and not by the order required for debugging.

Removing both quotation marks

Yet another compile time error:

```
PrintSomeNumbers.java:9: cannot find symbol
symbol   : variable Done
location: class PrintSomeNumbers
                        System.out.println(Done);
                                                ^
1 error
```

The compiler does not understand what `"Done"` means – because it is now treated as a variable, and not as a Literal.

The symbol that can't be found is of type variable, called `done` – this explains us what is looking for, and we know that the compiler looks for it in class `PrintSomeNumbers` – we learn these facts from the `"Symbol"` and `"Location"` lines in the compiler's output. In the following compile time errors of this type, note how these lines change.

Changing main to man

This is a runtime error, as the code compiles without a problem. However, when trying to execute the code, the following output is given:

```
Exception in thread "main" java.lang.NoSuchMethodError: main
```

The `main` is the starting point of the program – so the JVM tries to call the `main` method – but because its name was changed, the call fails – a `NoSuchMethod` error occurs.

Changing System.out.println(i) to System.out.println(i)

A compile time error occurs:

```
PrintSomeNumbers.java:6: cannot find symbol
symbol   : method println(int)
location: class java.io.PrintStream
        System.out.println(i);
                ^
```

The compiler can't find the symbol "println" in the PrintStream class. System.out is of type "PrintStream", and "println" is not one of this type's symbols.

Changing System.out.println(i) to println(i)

A compile time error occurs:

```
PrintSomeNumbers.java:6: cannot find symbol
symbol   : method println(int)
location: class PrintSomeNumbers
        println(i);
        ^
```

1 error

Here, the compiler can't find the symbol "println" in the class "PrintSomeNumbers". It looks in that class because the call's prefix is missing – The call looks like it is referring to this class. This touches the topic of breaking code to methods, that will be discussed in future lectures.

Removing the semicolon at the end of the System.out.println(i); statement

A compile time error occurs:

```
PrintSomeNumbers.java:6: ';' expected
        System.out.println(i)
                                ^
```

1 error

In java, statements end with a semi-colon (";"), very much like that a sentence ends with a period in English. Omitting this semi-colon causes a compile time error, as the compiler meet the "}" character when it expects the ";" character.

Removing the last brace ("}") in the program

A compile time error occurs:

```
PrintSomeNumbers.java:10: reached end of file while parsing
    }
    ^
1 error
```

In java, each opening bracket should match a closing bracket. The code between brackets is considered a block. In this case, this block is the whole class that is defined in the file. When compiling, the compiler was still expecting a "}" when it reached the end of the file, and is what the compiler outputs.

Changing $i = i + 1$ to $i = i - 1$

The code compiles without a problem, and seems to run without a problem – except that it counts from 0 down, and not up as expected. It also doesn't stop, because the condition " $i < 5$ " holds, even when $i = -100000$. This is a logical error, as neither the compiler nor the runtime don't raise any errors.

When i reaches -2147483648 it underflows and becomes some big number (Integer.MAX_VALUE, to be exact), which is greater than 6, so "Done" is printed, and the program exits.

Here's the first 3 and last 3 lines of the program's output:

```
0
-1
-2
.
.
.
-2147483646
-2147483647
-2147483648
Done
```

1.2 - Writing a Simple Program

This is a simple program to write. Here's the code.

PrintSomeNumbersBackward.java

```
public class PrintSomeNumbersBackward {
    public static void main(String[] args) {
        int i = 20;
        while (i >= 0){
            System.out.println(i);
            i = i - 1;
        }
        System.out.println("Done");
    }
}
```

1.3 - Playing With the Turtle

The submission block was omitted.

```
public class TurtleInitials {
    public static void main(String[] args) {
        // For more information about Vincent Van Gogh
        // Go to http://en.wikipedia.org/wiki/Vincent\_van\_Gogh

        Turtle vanGogh = new Turtle();

        // Moving to the top-left corner.
        vanGogh.moveForward(100);
        vanGogh.turnLeft(90);
        vanGogh.moveForward(80);
        vanGogh.turnLeft(180);

        // Drawing first line
        vanGogh.tailDown();
        vanGogh.moveForward(30);
        vanGogh.moveBackward(15);

        // Drawing second line
        vanGogh.turnRight(90); // Now facing downward
        vanGogh.moveForward(50);
        // Last line of "I"

        vanGogh.turnLeft(90);
        vanGogh.moveBackward(15);
        vanGogh.moveForward(30);

        // A small space between letters
        vanGogh.tailUp();
    }
}
```

```
// Continued from prev. page
vanGogh.moveForward(7);
vanGogh.tailDown();
vanGogh.turnLeft(90);

// The letter "F"
vanGogh.moveForward(35);
vanGogh.turnRight(90);
vanGogh.moveForward(27);
vanGogh.moveBackward(27);
vanGogh.turnLeft(90);
vanGogh.moveForward(15);
vanGogh.turnRight(90);
vanGogh.moveForward(27);

// Placing the turtle nicely
vanGogh.tailUp();
vanGogh.moveForward(15);
vanGogh.turnRight(90);
vanGogh.moveForward(50);

// Doing a little victory dance
int i = 0;
while (i < 300){
    int j = 0;
    vanGogh.turnRight(5);
    i++;
}

//... And hiding.
vanGogh.hide();
}
}
```