



Problem Solving and Programming

Assignment 1

The purpose of this first assignment is to get you started with coding with Java in your preferred IDE. The assignment will focus on *sequence* and using data types as well as simple usage of some predefined classes in the Java Class Library. In addition to that it will also introduce and test you on *selection* as one of the control structures you are going to use in Java.

Carefully read the assignment rules as defined on Moodle. In brief, do the assignments yourself, hand them in at the deadline and don't cheat.

Problems? Do not hesitate to ask your teaching assistant at the practical meetings (or the teacher at the lectures) if you have any problems. You can also post a question in the assignment forum in Moodle.

Submission We are only interested in your .java files. Hence, zip the directory named YourLnuUserName_assign1 (inside directory named src) and submit it using the Moodle submission system.



1 Getting Started

The first thing you need to do is to download and install Java and your preferred IDE (*either* IntelliJ IDEA or Eclipse). Instructions for this were given at the lecture and is available in the lecture notes. When everything is installed, you need to set up the environment for the course, again discussed in the lecture notes, but briefly do the following:

- Create an Eclipse workspace/IntelliJ project (a folder) with the name `java_courses` on some location in your home directory.
- Create a new Java project with the name `1DV506` inside the project folder or workspace.
- Create a package with the name `YourLnuUserName_assign1` inside the project. For example, it might look something like `w0222ab_assign1`.
- Save all program files from the exercises in this assignment inside the package `YourLnuUserName_assign1`.
- In the future: always create a new package on the same structure (`YourLnuUserName_assignX`) for each assignment and a new project (with the course code as name) for each new course using Java.

1.1 First program

Compile and run the following program in your IDE of choice to make sure everything is working as expected.

```
/* The classical "Hello World!" program. */  
public class Hello {  
  
    public static void main(String[] args) {  
        System.out.println("Hello World!"); // Print  
    }  
}
```



Sequence

The following excersises are about printing text and numbers on screen with Java. They will also make you do some simple calculations.

1.2 The Three Laws of Robotics

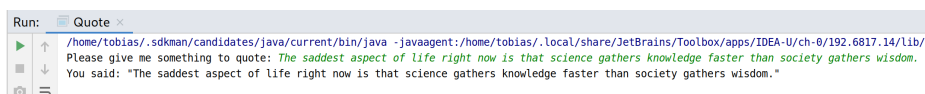
In the short story “Runaround” from 1942, Isaac Asimov introduced the three laws of robotics, later used in several novels and short stories by the author (and several others). The laws are:

1. A robot may not injure a human being or, through inaction, allow a human being to come to harm.
2. A robot must obey the orders given it by human beings except where such orders would conflict with the First Law.
3. A robot must protect its own existence as long as such protection does not conflict with the First or Second Laws.

Write a program called `ThreeLaws.java` that displays the laws on screen using Java, each on its own separate line.

1.3 A Quote

Write a program called `Quote.java` that asks for a sentence (using `Scanner`) and prints it in quotes (`"`). To do this, you have to look at *escape sequences* in chapter 4.3.2 in the latest edition of the book. An example of running the program, and entering a quote by Isaac Asimov, can be seen below:



```
Run: Quote
/home/tobias/.sdkman/candidates/java/current/bin/java -javaagent:/home/tobias/.local/share/JetBrains/Toolbox/apps/IDEA-U/ch-0/192.6817.14/lib/
Please give me something to quote: The saddest aspect of life right now is that science gathers knowledge faster than society gathers wisdom.
You said: "The saddest aspect of life right now is that science gathers knowledge faster than society gathers wisdom."
```



1.4 Fahrenheit

Write a program `Fahrenheit.java` using the class `Scanner` to read a real number (The Fahrenheit temperatur F) and then print the corresponding Celsius temperetaure C . The relationship between C and F is $F = (9/5) \times C + 32$. See below for a sample run of the program:

```
Provide a temperature in Fahrenheit: 100
Corresponding temperature in Celsius is 37.7777777777778
```

1.5 Time

Write a program `Time.java`, which reads a number of seconds (an integer) and then prints the same amount of time given in hours, minutes and seconds. Hint: Use integer division and the modulus (remainder) operator. An example of an execution:

```
Give a number of seconds: 9999
This corresponds to: 2 hours, 46 minutes and 39 seconds.
```

1.6 Sum of Three

Write a program `SumOfThree.java` which asks the user to provide a three digit number. The program should then compute the sum of the three digits. For example:

```
Provide a three digit number: 483
Sum of digits: 15
```



1.7 Change – VG-task

Exercise 1.7 is marked as VG task → only mandatory for those of you that aspire for a higher mark (A or B).

Write a program `Change.java` that computes the change a customer should receive when he has paid a certain sum. The program should exactly describe the minimum number of Swedish bills and coins that should be returned rounded off to nearest krona (kr). Example:

```
Price: 372.38
Payment: 1000
Change: 628
1000 kr bills: 0
500 kr bills: 1
200 kr bills: 0
100 kr bills: 1
50 kr bills: 0
20 kr bills: 1
10 kr coins: 0
5 kr coins: 1
2 kr coins: 1
1 kr coins: 1
```

Using library classes

These tasks use the above but also libraries that have been introduced in the lectures, for example `Math` and `Random`.

1.8 Area

Write a program `Area.java` which reads a radius (`R`, a float) and computes the area `A` of a circle with radius `R`. An example of an execution:

```
Provide radius: 2.5
Corresponding area is 19.6
```



1.9 Random numbers

Write a program `RandomSum.java` generating and printing the sum of five random numbers in the interval `[1,100]`. An example of an execution:

```
Five random numbers: 6 57 5 95 44
There sum is 207
```

1.10 Distance

Write a program `Distance.java` which reads two coordinates in the form (x, y) and then computes the distance between the points, using the formula

$$distance = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

The answer should be presented with three decimal digits. Below is an example run of the application:

```
Point x1: 9
Point y1: 7
Point x2: 3
Point y2: 2
The distance is: 7,810
```

1.11 Wind Chill – VG-task

Exercise 1.11 is marked as VG task → only mandatory for those of you that aspire for a higher mark (A or B).

Write a program `WindChill.java` that asks the user for a temperature in °C and the wind speed (measured in m/s) and then computes the so-called wind chill temperature T_{wc} using the Wind Chill Index formula (see <https://planetcalc.com/2087/>):

$$T_{wc} = 13.12 + 0.6215T_a - 11.37V^{0.16} + 0.3965T_aV^{0.16}$$



where T is the temperature (Celsius) and V is the wind speed (kilometers per hour). Notice you must convert the wind speed from meter per second to kilometers per hour before you can apply the formula. And we expect you to present the result with one decimal (correctly rounded off). For example:

Temperature (C): -7.8

Wind speed (m/s): 8.4

Wind Chill Temperature (C): -16.7

Selection

The primary focus of the following tasks is *selection* but you will need to work with what has been done previously in order to solve them.

1.12 Doctor Who TV-series

One of the longest running TV-series is the British Doctor Who series. It began in 1963 but had a pause from 1989 till 2005 when it was resumed. It has now been confirmed that the series will continue at least till 2020. Write a program called `DoctorWho.java` where you ask the user for a year and return what season of the series it was. The years from the start until 1989 is usually called the original series, while the later series are called the modern series. The program must of course look for the pause and the time before it started and after 2020. No iteration is required, the examples below are from repeated runs of the program:

```
What year? 1975
That was the original series.
```

```
(program re-run)
```

```
What year? 1995
That year it was a pause.
```



(program re-run)

What year? 2019

That is the modern doctor.

(program re-run)

What year? 2025

No news on a doctor yet, but one can hope!

1.13 Hit a nine

In this task you are to create a program called `Nine.java` which will simulate a simple game using selection (no iteration, only one play-through per execution). The game is one using two normal six sided dice. You play against the computer and the objective is to get as close to nine as possible, but not more.

The game works as follows. Both you and the computer have two possible dice to roll. First you roll a die and then decide if you would like to roll another. When you are done, that is if you decide to roll another or to skip the second roll, it is time for the computer to do the same. The computer rolls the first die and if it is four or lower, it will roll another, otherwise it will skip the second roll.

When both players have done their rolls, the program should calculate who, if anyone, won. As stated, the one closest to nine wins, unless it is ten or more. If either player gets ten or more, the player is declared "fat" and automatically loses. As with the previous task, only one run-through is needed per execution.

Playing a game

=====

Ready to play? (Y/N) Y

You rolled 3

Would you like to roll again? (Y/N) Y

You rolled 6 and in total you have 9



The computer rolled 3
The computer rolls again and gets 4 in total 7
You won!

1.14 Numbers in order

Write a program called `Numbers.java` that accepts three number and prints them in order from lowest to highest. The program must work with both positive and negative values, but only integers. This is an example run:

```
First number: 64
Second number: 32
Third number: 128
Correct order: 32 64 128
```

1.15 Programming Languages – VG-task

Exercise 1.15 is marked as VG task → only mandatory for those of you that aspire for a higher mark (A or B).

Let the user enter three programming languages in a program called `Programming.java`. When the three languages have been entered, the program will print them out in the correct alphabetic order. See below for an example run:

```
What is the first programming language? Dart
And the second? Java
And the third? Ada
Alphabetic order is Ada, Dart and Java
```