Joanna Klukowska
joannakl@cs.nyu.edu

Assignment 1: Getting Started With Java

Due date: Feb. 3, 11:55PM EST.

You may discuss any of the assignments with your classmates and tutors (or anyone else) but **all work for all assignments must be entirely your own**. Any sharing or copying of assignments will be considered cheating.

You should not use any features of Java that have not been covered in class. If you have doubt if you are allowed to use certain structures, just ask your instructor.

The purpose of this assignment is to make sure you have a working Java Development Kit (JDK) and the Eclipse Integrated Development Environment (IDE) on your own computer.

Notice that you are not required to use the Eclipse IDE for this course. You can complete all the programming assignment using a text editor and command line tools or another IDE. I, personally, find the debugging tools to be much more friendly in the IDE, such as Eclipse, then the command line interface to `jdb`. The programs in this assignment can easily be completed using only command line interface and a basic text editor. If you setup Eclipse right away, you will get a chance to get used to it, before the real need for IDE occurs in future assignments.

You can always use the computers in the public computer labs in NYU to complete the homework assignments. See http://www.nyu.edu/its/labs/ for locations, hours and software availability of different labs. In such a case, you do not need to (and in fact you cannot) install the software on lab computers.

Install JDK and Eclipse

You can download Java JDK (Java Development Kit) from http://www.oracle.com/technetwork/java/javase/downloads/index.html. You should download version 8 (or 7). Make sure to pick a version appropriate for your operating system.

You can download Eclipse from https://eclipse.org/. Make sure to pick a version appropriate for your operating system.

Follow instructions in section 1.12 of the textbook to download, install and start using Eclipse.

You may also watch a video tutorial *Up and Running With Eclipse* by Charles Kelly on lynda.com at
http://www.lynda.com/Eclipse-tutorials/Up-Running-Eclipse/111243-2.html?org=nyu.edu.
This tutorial covers languages other than Java, but you may just skip these sections.

Exercises

Problem 1 (30 points): Find the number of years

Write a program that prompts the user to enter the number of minutes and displays the number of years and days for the minutes. For simplicity assume that a year is 365 days.

Sample run when user enters 1000000000 minutes:

```
Enter the number of minutes: 1000000000
1000000000 minutes is approximately 1902 years and 214 days.
```

Sample run when user enters 5555555 minutes:

```
Enter the number of minutes: 5555555
5555555 minutes is approximately 10 years and 208 days.
```

Sample run when user enters 3000 minutes:

```
Enter the number of minutes: 3000
3000 minutes is approximately 0 years and 2 days.
```

You may assume that the user will always enter a positive number.

Call your file: `Minutes.java`.

Hint: You should read about the mathematical operators for integer division and modulus.

Joanna Klukowska
joannakl@cs.nyu.edu

Problem 2 (35 points): Metro card cost

A base fare for a subway or a local bus ride in NYC is currently $2.75. If you purchase a metro card and put at least $5.50 on it, you will receive a bonus of 11% that is credited to your metro card immediately. Write a program that prompts the user for the amount of money that they wish to add to a previously empty metro card and computes how many base fares they will be able to charge on that card (do not use the fractional parts).

Here are some sample runs of the program:

```
Enter the amount of money you wish to put on your card: 5.00
You will have 1 fare(s).
```

```
Enter the amount of money you wish to put on your card: 5.50
You will have 2 fare(s).
```

```
Enter the amount of money you wish to put on your card: 55.00
You will have 22 fare(s).
```

```
Enter the amount of money you wish to put on your card: 550.00
You will have 222 fare(s).
```

You may assume that the user will always enter a positive number.

Call your file: `MetroCard.java`.

Hint: Based on the amount entered, your program needs to decide if the user will receive the 11% bonus.

Problem 3 (35 points): Highest SAT score

Write a program that prompts the user first to enter the number of students and then to enter the name and Math SAT score for each student (the name should be a single string with no spaces, the score should be an integer number from 200 to 800). Your program can assume that both the name and the score are entered correctly.

The program should then display the name and score for the student with the highest score.

Here is a sample run of the program:

```
Enter the number of students: 4
Student 1 name: Bob
Student 1 bonus: 650
Student 2 name: Joan
Student 2 bonus: 575
Student 3 name: Anna
Student 3 bonus: 775
Student 4 name: Jacob
Student 4 bonus: 725

Anna received the highest score of 775.
```

Call your file: `Scores.java`.

Hint: Make sure that your program correctly identifies the largest score even if it is the first or the last one entered.

Note: You are not allowed to use arrays, ArrayLists, etc. (i.e., you should not be storing the information about all the students).

Grading

Does the program compile? If not, you will loose all the points for that problem.

Is the program properly documented? (worth approximately 20% of each problem)

Proper documentation at this point in the course includes:

- preamble with the name of the author, date of creation and brief description of the program (the description should specify what the program does, not that it is a solution to problem 1 of homework 1);

Joanna Klukowska
joannakl@cs.nyu.edu

- appropriately chosen variable names, i.e., descriptive names (a good name for the variable that stores the bonus amount in the last problem is `bonus`, not `x`);
- comments inside the code describing steps needed to be taken to accomplish the goal of the program;
- appropriate formatting, indentation and use of white space to make the code readable.

Remember that the code is read by humans and it should be easy to read for people who were not involved in its development.

Is the program well developed? (worth approximately 40% of each problem) Make sure you create variables of appropriate types, use control statements (conditionals and loops) that are appropriate for the task, accomplish your task in a well designed and simple way (not a convoluted algorithm that happens to produce the correct output for some unknown reason). You should also design a friendly and informative user interface.

Is the program correct? (worth approximately 40% of each problem) Make sure that your program produces valid results that follow the specification of the problem every time it is run. At this point you can assume a "well behaved user" who enters the type of data that you request. If the program is not completely correct, you get credit proportional to how well it is developed and how close you got it to the completely correct code.

What and how to submit?

You should submit three source code files combined into a single **zip** file to NYU Classes. Do not submit all the files that Eclipse creates, just the source code files that have `.java` extensions. Name your classes as specified in the problems.

Questions

Post any questions you have regarding this assignment to Piazza under the "homeworks" topic.