INF100 H17 Assignment 2

Due on Friday, September 8, 2017

Albin Severinson, Dag Haugland

Abstract

For this second exercise for INF100, we will build an interactive currency conversion program! We will start off with the most basic functionality and will gradually add more features. This means that you will be working on the same program throughout the entire assignment. We will only grade (pass/fail) the program for the final problem of the assignment. If you have questions specific to this assignment you should send them to me at `albin.severinson@uib.no`.

Submission

1. Add your program file to a `.zip` archive named `ex1_firstname_lastname.zip`. You should only add your final program, i.e., the one for problem 5.
2. Log in to `https://mitt.uib.no` and press the box marked with "INF100" to enter the page for this course.
3. Press "Oppgåver" on the left. Next, select "Assignment 2".
4. Press "Lever oppgåve" and upload the `.zip` file you created in step 1.

General Guidelines

These are general guidelines that will (hopefully) help you maintain your sanity through your programming endeavours.

- You should make a copy of all your code after completing each problem so that you can roll back to a previous version when you inadvertently break something. Even the developers at Google sometimes have to roll back.
- Set aside time to clean up your code. The number that gets bandied around on the internet is that the best programmers spend roughly half their time cleaning up and rewriting their code, i.e., you are halfway done when you get it working.
- You will save time by making sure you properly understand your current program before moving on to the next problem.
- You should probably be spending a significant part of your time on stackoverflow when working on these assignments.
- If you want to figure out how a standard class or method works (`Scanner`, say), you can look it up on the Java docs page. Instead of trying to find what you are looking for at the Java docs webpage you should probably just Google it (searching for "java Scanner" on Google gives you its docs page as the first result).

Problem 1

Write a program that converts NOK to USD. The amount of NOK to convert and the NOK-to-USD exchange rate should be defined as two separate variables of type double. As a reminder, here is an empty program for you to start off with.

```
public class Exchange {
    public static void main(String[] args) {

    }
}
```

Problem 2

Update your program (remember to first make a copy!) to get the amount of NOK from the user. Stackoverflow has you covered in case you don't know how to read user input in Java: `https://stackoverflow.com/questions/15914676/how-to-get-input-via-command-line-in-java`.

Example

Your program output should look like this. We have added the > to lines with user input.

```
Hello, how much would you like to convert?
> 100
10.808603647206144
```

Problem 3

Considering the current political climate we should extend our conversion software to other, non-USD, currencies. Extend your program to allow converting NOK to EUR and GBP (and any other currencies you might fancy) in addition to USD. This means that you need to add at least two more exchange rates and that you need to ask the user what currency to convert to. Remember that you should use .equals and not == to compare strings in Java.

Example

Your program output should look like this. Note that we have made the output look a bit nicer (are you going back to clean up and improve your code once you get it working?).

```
What would you like to convert to?
Type: USD / EUR / GBP
> EUR
How much to convert?
> 100
100.0 NOK equals 10.808603647206144 EUR
```

Problem 4

You have probably noticed that humans are very imprecise beings. Some humans would even argue that "eur", "Eur", and "EUR" mean the same thing! Computers obviously do not agree with this. However, because your software has to interact with these humans you do need address this issue. Specifically, improve your program such that case does not matter, i.e., entering "eur", "Eur", and "EUR" have the same result. You can do this with only 1 additional line of code.

Example

Your program output should look like this.

```
What would you like to convert to?
Type: USD / EUR / GBP
> eur
How much to convert?
> 100
100.0 NOK equals 10.808603647206144 EUR
```

Problem 5

I doubt even the high-frequency traders care about printing 14 decimal points! Use printf instead of println to only print a more reasonable 2 decimal points.

Example

Two iterations of your program output should look like this.

```
What would you like to convert to?
Type: USD / EUR / GBP
> USD
How much to convert?
> 10
10.0 NOK equals 1.28 USD
```

Getting Ahead

You may not be competing with Forex any time soon, but you have already built a simple currency conversion program! Congratulations. If you are looking to improve your piece of software (or your programming skills) you should not rest on your laurels quite yet though.

This part is meant as preparation for the coming course material and will not be graded. Then again, you never have to catch up if you are always getting ahead.

Loops

Re-running your program for each conversion is not very efficient. Instead of exiting after printing its output, have it ask the user input for a new currency and amount. This means you will have to learn about `while` loops. thenewboston is pretty good at explaining these things: `https://www.youtube.com/watch?v=8ZuWD2CBjgs`. Furthermore, note that Scanner.nextInt() does not read the newline character. This is something you will have to deal with. Two iterations of your program output should look like this.

```
What would you like to convert to?
Type: USD / EUR / GBP
> USD
How much to convert?
> 10
10.0 NOK equals 1.28 USD
What would you like to convert to?
Type: USD / EUR / GBP
> EUR
How much to convert?
> 200
200.0 NOK equals 21.62 EUR
```

Input Validation

As you have probably noticed, humans are not very good at following instructions. In fact, you suspect that some of your users may misspell the name of the currency that would like to convert to! Even worse, maybe someone will enter a non-double value when asked for the amount of currency with the explicit intention of crashing your software. Furthermore, even without any ill intention we sometimes manage to destroy stuff (such as this $500 million rocket) due to computers receiving a value they do not expect. Specifically, please fix the below bugs.

Bugs:

- Entering an invalid currency name should result in an error message and the user being asked again.
- Entering something other than a number when asked for the amount of currency should result in an error message and the user being asked again. There is an easy way and a hard way to do this. The hard way is to use exceptions. If you like doing things the hard way thenewboston has an explanation at `https://www.youtube.com/watch?v=K_-3OLkXkzY`. The easy way uses the `Scanner` somehow.