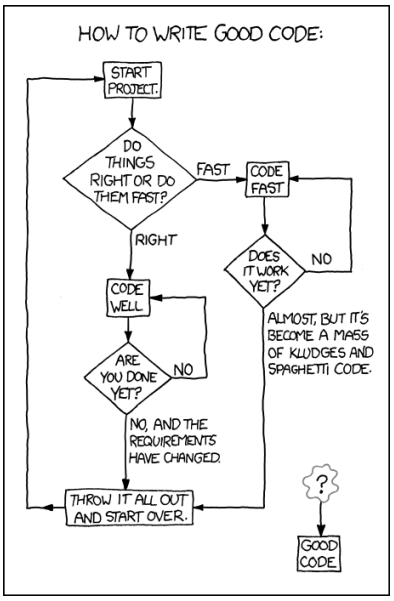
Intro2cs Ex2 – Variables, Operators, Flow Control, API



Objectives

This exercise's purpose is to get you a little more comfortable with some basic programming in Java. The exercise will focus on working with variables, operators, conditions and API.

Tasks

You are required to solve tasks a-c.

The grading of your solution composed from two parts: an automatic testing of the correctness of your code and manual review of the quality of your code. All of the automatic testers supplied to you in ex2testing.jar. See detailed instructions for running the testers.
Note that the automatic testers compare your outputs to the school solution outputs. Therefore, you should verify that your solution behaves EXACTLY like the school solution (pay attention to capital/lowercase letters, redundant spaces, etc.)

In the following description of the tasks, the `Courier New font represents text that should be the output of your program,` and **`the Courier New bold font represents text that entered by the user of the program.`**

a. Target heart rate calculator

A target heart rate is useful in establishing a suitable exercise routine. To determine this rate, subtract a person's age from 220. The upper limit of the target is 85% of this value; the lower limit is 65%. Write a class **Ex2a** that reads a person's age and prints the range of that person's target heart rate.

The output of the program should be as following:

```
This program calculates your target heart rate while exercising.
Enter your age: 24
Your estimated target heart rate zone is 127 – 167 beats per
minute.
```

Note that your implementation should print the EXACT lines above (and only the numbers in bold may vary depending on the users input).

Decimal precision: As shown in the example, the requested output is an integer. However, for maximal precision use doubles arithmetic and round the result to the nearest integer. You may find helpful using the method `Math.round(double num)`.

Assume that the input is always legal (integer, non-negative).

b. Solving a system of quadratic equation

Write a class **Ex2b** that prints all real solutions to the quadratic equation: $ax^2 + bx + c = 0$. The program should receive as input the three coefficients as doubles in the order *a b c* (spaces between each coefficient). Solve the quadratic equation and display the solutions.
The program should output:

1. The system we are going to solve.
2. One of the following:
    - `There are no real solutions.`
    - `There is one real solution: x.`
    - `There are two real solutions: x1, x2.`
      If there are two solutions, the solutions should be printed in ascending order: $x_1<x_2$.
    - `All real numbers are solutions.`

The solution should be rounded to 2 decimal places. As in part A, use doubles arithmetic and round only when printing the results. There are many ways to do that in Java, one is to use `System.out.format` instead of `System.out.println`.

For example:

```
double pi = 3.14159265; // Outside this example, use Math.PI instead.
double e = 2.718281; // Outside this example, use Math.E instead.
System.out.format("PI is %.2f, Euler's number is %.4f%n", pi, e);
```

Will print: `PI is 3.14, Euler's number is 2.7183`
'%.xf' – prints a float number with x decimal places after the point.
'%n' – jumps to the next line.

For more information, see this Java tutorial.

Another problem that should be addressed is comparing the determinant of the quadratic formula to zero (and we have only one solution). The problem is that when comparing floating-point numbers (float, double), we quickly discover that we get round-off errors. This has to do with the limited precision of Java floating point variables. For examples and information on possible solutions, see this blog.

Last problem that should be mentioned: Java can return -0 instead of 0 (for example when b=0 and c=0, a>0). A simple work around is adding 0 to your results.

The output of the program should be as following:

```
This program solves a quadratic equation.
Enter the coefficients a b c: 1 3 2
Equation: 1.00*x^2 + 3.00*x + 2.00 = 0
There are two real solutions: -2.00, -1.00.
```

Note again that your implementation should print the EXACT lines above (and only the numbers in bold may vary depending on the users input).

Another run may be:

```
This program solves a quadratic equation.
Enter the coefficients a b c: 0 -2 1.5
Equation: 0.00*x^2 + -2.00*x + 1.50 = 0
There is one real solution: 0.75.
```

Or:

```
This program solves a quadratic equation.
Enter the coefficients a b c: 1 2 3
Equation: 1.00*x^2 + 2.00*x + 3.00 = 0
There are no real solutions.
```

Or:

```
This program solves a quadratic equation.
Enter the coefficients a b c: 0 0 0
Equation: 0.00*x^2 + 0.00*x + 0.00 = 0
All real numbers are solutions.
```

c. Playing with strings

Write a class **Ex2c** that uses Java's String object to manipulate strings.

First, read the API of the class String. Pay special attention to the subsection "Method Summary".

The program should receive a sentence from the user, print a menu with options and respond according to the user's choice.

The menu should include the following options:

1. **Number of characters** – print the number of characters.
2. **Remove vowels** – print the sentence after removing all vowels (aeoiu, lower case only).
3. **Replace a string** – ask the user for two strings – a string to replace and a new string. Print the sentence after replacing all occurrences of the string with the new string.
4. **First and last word** – if the sentence starts and end with the same word, print the sentence after removing this word from both the start and end; else print the sentence without changes.
5. **Remove following characters** – ask the user for a string. Print the sentence after removing 'n' characters that follow this string, while 'n' is the number of characters of the input string (if the string appears more than once, remove only the characters following the first appearance). If there are less than 'n' characters left in the sentence, remove the characters left until the end of the sentence.
6. **Middle characters** – if the number of characters without spaces is even, print the two middle letters; else return the three middle letters. Disregard spaces in this case.
7. **Convert to upper/lower case** – if the sentence contains one or more upper case characters, convert all characters to lower case; else convert all to upper case.

A sample run of the program could be:

```
Please enter a sentence:
I think therefore I
1) Number of characters
2) Remove vowels
3) Replace a string
4) First and last word
5) Remove following characters
6) Middle characters
7) Convert to upper/lower case
Choose option to execute:
1
The result is: 19
```

If the user chooses 2:

```
The result is: I thnk thrfr I
```

If the user chooses 3:

```
String to replace:
I
New string:
you
The result is: you think therefore you
```

If the user chooses 4:

```
The result is: think therefore
```

If the user chooses 5:

```
Enter a string:
think
The result is: I thinkefore I
```

If the user chooses 6:

```
The result is: he
```

If the user chooses 7:

```
The result is: i think therefore i
```

Simplifying assumptions:

1. You can assume that the input sentence has at least one word and at least one character.
2. You can also assume that there is only one space between words, and no spaces at the beginning and end of sentence.
3. You can assume that all the inputs are correct - no need to validate the user's input.

If you are not sure how the program should operate, consult the school solution.

An example of comparing String objects:

```
String str = "ex2";
if (str.equals(input)) {
      System.out.println("Equal!");
} else {
      System.out.println("Not equal!");

}
```

The code above compares two String instances – `str` and `input`. If they are equal it prints "Equal!", else it prints "Not equal!".

Using the school solution

Due to different reasons, using the school solution is possible only inside the CS system: `~introcsp/bin/ex2/ex2a` (`ex2b`/`ex2c` for the other parts).

Testing

For your convenience, we provide you with all of the testers we use to check automatically your code. Your responsibility is to check that you pass them. If you think your output is right and the tester still fails, it is probably because your output is not EXACTLY as described in guidelines of this exercise.

To run the tests on the CS machines, run the following command from the shell in a directory that contains your ex2.jar: `~introcsp/bin/testers/ex2 ex2.jar`

How to run the testers from home:

1. In your working directory, besides your Ex2{a,b,c}.java files and Ex2 Tester files, you should also have two jar files: junit4 package and the intro package.
2. To compile use:
   `javac -cp "*.jar" *.java` (windows)
   `javac -cp \* *.java` (mac/linux).
3. To run the tester use:
   `java -cp "*.jar" Ex2TesterDriver` (windows)
   `java -cp .:\* Ex2TesterDriver` (mac/linux)
4. Note that Ex2TesterDriver runs testers for parts A, B and C. If you want to test only one of them remove the other testers from line 21 in Ex2TesterDriver.java:
   Result res=junit.run(Ex2aTester.class,Ex2bTester.class,Ex2cTester.class);
   and compile again.

Further Guidelines

- Please remember to follow the coding style guidelines (can be found on the course website).
- Document your code – add comments before major code blocks.
- In order to receive input from the user, use the class `Scanner`. Click here to view the `Scanner` class documentation.

- In any case of ambiguity, use the school solution for clarification.

Submission

- You should submit the following files:
  1. `Ex2a.java`
  2. `Ex2b.java`
  3. `Ex2c.java`
  4. README (as explained in the course guidelines on the course website)

  Create a JAR file named ex2.jar containing only these files by invoking the shell command:

  ```
  jar cvf ex2.jar Ex2a.java Ex2b.java Ex2c.java README
  ```

  The JAR file should **not** contain any .class files.

  It is recommended to check your JAR file by copying it to a different directory, unpacking it by executing the command:

  ```
  jar xvf ex2.jar
  ```

  and verifying that all your `.java` files and README were indeed successfully created.
- If you haven't done so already, you must register for the course grading system (only once!).
- You should submit the file ex2.jar via the "Upload File" link on the course home page, under ex2 link.

Good luck!