# ASSIGNMENT 1

COMP-202, Winter 2019

Due: Friday, Feb. 1st, 11:59pm

**Please read the entire PDF before starting. You must do this assignment individually.**

| | |
|---|---|
| Question 1: | 35 points |
| Question 2: | 35 points |
| Question 3: | 30 points |
| | 100 points total |

**It is very important that you follow the directions as closely as possible.** The directions, while perhaps tedious, are designed to make it as easy as possible for the TAs to mark the assignments by letting them run your assignment, in some cases through automated tests. While these tests will never be used to determine your entire grade, they speed up the process significantly, which allows the TAs to provide better feedback and not waste time on administrative details. Plus, if the TA is in a good mood while he or she is grading, then that increases the chance of them giving out partial marks. :)

Up to 30% can be removed for bad indentation of your code as well as omitting comments, or poor coding structure.

**To get full marks, you must:**

- Follow all directions below
  - In particular, make sure that all classes and method names are **spelled and capitalized exactly** as described in this document. Otherwise, a 50% penalty will be applied.
- Make sure that your code compiles.
  - Non-compiling code will receive a very low mark.
- Write your name and student ID as a comment in all .java files you hand in
- Indent your code properly
- Name your variables appropriately
  - The purpose of each variable should be obvious from the name
- Comment your work
  - A comment every line is not needed, but there should be enough comments to fully understand your program

## Part 1 (0 points): Warm-up

*Do* **NOT** *submit this part, as it will not be graded. However, doing these exercises might help you to do the second part of the assignment, which will be graded. If you have difficulties with the questions of Part 1, then we suggest that you consult the TAs during their office hours; they can help you and work with you through the warm-up questions. You are responsible for knowing all of the material in these questions.*

**Warm-up Question 1** (0 points)
Create a file called `HelloWorld.java`, and in this file, declare a class called `HelloWorld`. This class should define only one method called `main()`. In the body of this method, use `System.out.println()` to display "`Hello world!`". You can find such a class in the lecture slides; make sure you can compile and run it properly.

**Warm-up Question 2** (0 points)
Create a file called `Diagram.java`, and in this file, declare a class called `Diagram`. This class should define only one method called `main()`. In the body of this method, use five statements of `System.out.println()` to display the following pattern:

```
  22
2   2
   2
 2
22222
```

Use *Strings* composed out of the space character and the character '2'. For a bit of fun, try to draw the pattern '202'.

**Warm-up Question 3** (0 points)
**Practice with Number Bases**:
We usually use base 10 in our daily lives, because we have ten fingers. When operating in base 10, numbers have a `one`s column, a `ten`s column, a `hundred`s column, etc. These are all the powers of 10.

There is nothing special about 10 though. This can in fact be done with any number. In base 2, we have each column representing (from right to left) 1,2,4,8,16, etc. In base 3, it would be 1,3,9,27, etc.

Answer the following short questions about number representation and counting.

1. In base 10, what is the largest digit that you can put in each column? What about base 2? Base 3? Base n?
2. Represent the number thirteen in base 5.
3. Represent the number thirteen in base 2.
4. What is the number 11010010 in base 10?

**Warm-up Question 4** (0 points)
**Logic**:

1. What does the following logical expression evaluate to?

   `(false or false) and (true and (not false))`

2. Let $a$ and $b$ be boolean variables. Is it possible to set values for $a$ and $b$ to have the following expression evaluate as *false*?

   `b or (((not a) or (not a)) or (a or (not b)))`

**Warm-up Question 5** (0 points)

**Expressions**: Write a program `EvenAndPositive` that takes an integer as input and displays on your screen whether it is true or false that such integer is even, positive, or both.

An example of what you could see on the Interactions pane is:

```
> run EvenAndPositive -2
-2 is an even number: true
-2 is a positive number: false
-2 is a positive even number: false

> run EvenAndPositive 7
7 is an even number: false
7 is a positive number: true
7 is a positive even number: false
```

**Warm-up Question 6** (0 points)

**Conditional statements**: Write a program `HelloBye` that takes an integer as input. If the integer is equal to 1, then the program displays `Hello everyone!`, otherwise it displays `Bye bye!`.

An example of what you could see on the Interactions pane is:

```
> run HelloBye 0
Bye bye!

> run HelloBye 1
Hello everyone!

> run HelloBye 329
Bye bye!
```

## Part 2

*The questions in this part of the assignment will be graded.*
The main learning objectives for this assignment are:

- Correctly declare and use variables.
- Learn how to build expressions containing different type of operators.
- Get familiar with String concatenation.
- Correctly use `println` (or `print`) to display information.
- Correctly use and manipulate inputs received by the program through command line arguments.
- Correctly use simple conditional statements.

All of the programs you have to write are run by providing *input arguments* through `args`. If you are using Eclipse, please read this for help with input arguments: `http://www.cs.colostate.edu/helpdocs/eclipseCommLineArgs.html`.

**Question 1: ISBN** (35 points)

The reference book suggested for Comp202 is 'How to Think Like a Computer Scientist: Java Version, 6th edition, 1491929561'. The last 10-digit number in the book (i.e., 1491929561) represent the ISBN of the book. ISBN is the acronym for **I**nternational **S**tandard **B**ook **N**umber and it is used to identify a specific book. One characteristic of this number is that $10d_{10} + 9d_9 + 8d_8 + \cdots + 3d_3 + 2d_2 + d_1$ must be a multiple of 11 (please note that $d_i$ denotes the $i$-th digit from the right). For example, for the ISBN of our reference book we can check the condition as follows:

$$10 \times 1 + 9 \times 4 + 8 \times 9 + 7 \times 1 + 6 \times 9 + 5 \times 2 + 4 \times 9 + 3 \times 5 + 2 \times 6 + 1 = 253,$$

where 253 is multiple of 11. By convention, the digit $d_1$ can be any value from 0 to 10, where the character 'X' is used to denote de digit '10'. The other digits (from $d_{10}$ to $d_2$ ) must all be digits between 0 and 9. For example, given the 9-digit ISBN number 020131452 the value of $d_1$ must be equal to 5 since it is the only value of $d_1$ between 0 and 10 such that $10\times0+9\times2+8\times0+7\times1+6\times3+5\times1+4\times4+3\times5+2\times2+d_1$ is a multiple of 11.

For this assignment, we will work with a shorter version of an ISBN number (a number with only five digits). Particularly, as a Comp202 student, you have been commissioned to write a java program called `ISBN.java` to determine the digit $d_1$ given a 4 digit number. You can assume that the input number will not have any leading 0s. The program should take such number as input through command line arguments (`args`). Note that the program must take as input one number with four digits and **NOT** four numbers with one digit.

Let's look at three examples of what you should see in the Interactions Pane:

```
> run ISBN 2956
The last digit must be 4.
```

NOTES:

- The input represents the ISBN number 2956?, where $d_5 = 2$, $d_4 = 9$, $d_3 = 5$, $d_2 = 6$, and $d_1 =?$ (the value that your program must compute).
- The program displays 4 as the last digit since 4 is the only value for $d_1$ between 0 and 10 such that $5 \times 2 + 4 \times 9 + 3 \times 5 + 2 \times 6 + 4$ (which is 77) is a multiple of 11.

```
> run ISBN 5724
The last digit must be X.
```

NOTES:

- The input represents the ISBN number 5724?, where $d_5 = 5$, $d_4 = 7$, $d_3 = 2$, $d_2 = 4$, and $d_1 =?$ (the value that your program must compute).
- The program displays $X$ as the last digit since 10 is the only value for $d_1$ between 0 and 10 such that $5 \times 5 + 4 \times 7 + 3 \times 2 + 2 \times 4 + 10$ (which is 77) is a multiple of 11.

```
> run ISBN 1521
The last digit must be 0.
```

NOTES:

- The input represents the ISBN number 1521?, where $d_5 = 1$, $d_4 = 5$, $d_3 = 2$, $d_2 = 1$, and $d_1 =?$ (the value that your program must compute).
- The program displays 0 as the last digit since 0 is the only value for $d_1$ between 0 and 10 such that $5 \times 1 + 4 \times 2 + 3 \times 2 + 2 \times 1 + 0$ (which is 33) is a multiple of 11.

Be sure to include the specified text in the output. That is, concatenate a *String* literal with the value of the appropriate variable.

**Question 2: Input Analyzer Program** (35 points)

This question is meant for you to practice creating expressions using the operators we have learned (mathematical, relational, and logical). All the code for this question should go in a file called `InputAnalyzer.java`.

Let $a$, $b$ and $c$ denote the three numbers the program receives as *input arguments*. You program should then display the following information:

1. Whether or not $a$, $b$ and $c$ are all non-negative numbers.
2. Whether or not at least one between $a$, $b$ and $c$ is an even number.
3. Whether or not $a$, $b$ and $c$ were entered in a strictly decreasing order.
4. Whether or not $a$, $b$ and $c$ are all non-negative numbers or entered in a strictly decreasing order.
5. Whether or not $a$, $b$ and $c$ are all non-negative and none of them is even.

Let's look at three examples of what you should see in the Interactions Pane:

Example 1:

```
> run InputAnalyzer 9.0 5.3 0.0
The numbers 9.0, 5.3, and 0.0 are all non-negative: true
At least one between 9.0, 5.3, and 0.0 is even: true
The numbers 9.0, 5.3, and 0.0 are in a strictly decreasing order: true
The numbers 9.0, 5.3, and 0.0 are either all non-negative or in a strictly decreasing order: true
The numbers 9.0, 5.3, and 0.0 are all non-negative numbers and none of them is even: false
```

Example 2:

```
> run InputAnalyzer 31.3 -3.0 -10.5
The numbers 31.3, -3.0, and -10.5 are all non-negative: false
At least one between 31.3, -3.0, and -10.5 is even: false
The numbers 31.3, -3.0, and -10.5 are in a strictly decreasing order: true
The numbers 31.3, -3.0, and -10.5 are either all non-negative or in a strictly decreasing order: true
The numbers 31.3, -3.0, and -10.5 are all non-negative numbers and none of them is even: false
```

Example 3:

```
> run InputAnalyzer 5.0 5.0 3.0
The numbers 5.0, 5.0, and 3.0 are all non-negative: true
At least one between 5.0, 5.0, and 3.0 is even: false
The numbers 5.0, 5.0, and 3.0 are in a strictly decreasing order: false
The numbers 5.0, 5.0, and 3.0 are either all non-negative or in a strictly decreasing order: true
The numbers 5.0, 5.0, and 3.0 are all non-negative numbers and none of them is even: true
```

Be sure to include the specified text on each output line. That is, concatenate a *String* literal with the value of a variable. Note that, if you find yourself writing the same expression more than once, this is a good place to introduce a new variable. It will make your code more readable, easier to write and to maintain.

**Question 3: Pizza Calculator** (30 points)

Last week, Johnny went to a pizza shop and asked for a 12-inch pizza. The shop owner said they didn't have any 12-inch pizzas left and suggested selling Johnny two 6-inch pizzas for the same price. Johnny accepted the offer.

After Johnny got home, his girlfriend told him that he had been ripped off. The area of a circle is

$$Area = \pi r^2$$

Where $r$ is the radius of the circle and $\pi$ is roughly 3.14. So, the 12-inch pizza, with a radius of 6 inches, has an area of

$$3.14 * 6^2 = 113.04$$

whereas a 6-inch pizza, with a radius of 3 inches, has an area of

$$3.14 * 3^2 = 28.26$$

This means that two 6-inch pizzas only have an area of $28.26 * 2 = 56.52$! Thus, two 6-inch pizzas are equivalent to only half of a 12-inch pizza!

Feeling slightly traumatized by this experience, Johnny decides to hire you to write a Java program to prevent being ripped off again by unethical pizza shops.

For this question, create a Java program called `PizzaCalc.java` and write a program that does the following: the first input the program receives is used to determine the mode in which the program runs. This program has two modes: quantity mode and price mode. The integer 1 is used to represent the quantity mode, and 2 is used to represent the price mode. If Johnny (or the user) inputs an integer that is neither 1 nor 2, then the program should output a message saying that the given mode is not supported.

A Quantity Mode (10 points) When Johnny selects the quantity mode, he will also input two additional numbers additional numbers:

   (a) The size (diameter) of the larger pizza (integer)

   (b) The size (diameter) of the smaller pizza (integer)

   The program must first display the information received, and then display the number of smaller pizzas Johnny must order in order for him to get the same amount of pizza as one large pizza. For simplicity, the output number can have decimals.

B Price Mode (20 points) When Johnny selects the price mode, he will also input four additional numbers:

   (a) The size (diameter) of the larger pizza (integer)

   (b) The price of the larger pizza (double)

(c) The size (diameter) of the smaller pizza (integer)

(d) The number of smaller pizzas (integer)

The program must first display the information received, and then display the total price Johnny should be paying to buy the smaller pizzas such that the amount of pizza per dollar is the same as that of the larger pizza. For example, if a 12-inch pizza is 10 dollars, and the shop owner offers to sell you two 6-inch pizzas for the same price, you should be able to say, "a fair price of two 6-inch pizzas should be 5 dollars, not 10 dollars". When using the price mode, the program calculates the fair price of the specified number of smaller pizzas.

Let's look at some examples of what you should see in the Interactions Pane:

Example 1:

```
> run PizzaCalc 1 14 8
You selected "Quantity mode".
The diameter of the large pizza is 14 inches.
The diameter of the small pizza is 8 inches.

You should order 3.0625 small pizzas.
```

Example 2:

```
> run PizzaCalc 2 12 10.0 6 2
You selected "Price mode".
The diameter of the large pizza is 12 inches.
The price of one large pizza is 10.0 dollars.
The diameter of the small pizza is 6 inches.
You want to buy 2 small pizzas.

The fair price to pay for 2 small pizzas is 5.0 dollars.
```

Example 3:

```
> run PizzaCalc 2 14 9.55 8 4
You selected "Price mode".
The diameter of the large pizza is 14 inches.
The price of one large pizza is 9.55 dollars.
The diameter of the small pizza is 8 inches.
You want to buy 4 small pizzas.

The fair price to pay for 4 small pizzas is 12.473469387755102 dollars.
```

Example 4:

```
> run PizzaCalc 214 5
This mode is not supported.
```

NOTES:

- For simplicity, when using quantity mode, the number displayed indicating the number of pizzas to buy can contain decimals. If you want to make you program more realistic then you can round this number up to the next integer. For example, 3.0625 should round up to 4.
- Don't worry about rounding numbers to 2 decimal places. Displaying any number of digits after the decimal point is fine.

- You can make the following assumptions about the inputs:
  - The types are always correct.
  - The number indicating the diameters are positive integers, and the diameter of the large pizza is always greater than the diameter of the small pizza.
  - The price is a positive double.
  - The number of pizzas to buy is a positive integer.
- Note that when in 'Quantity mode' your program should be looking only for 2 additional inputs. When in 'Price mode' your program should be looking for 4 additional inputs. If an incorrect mode is selected, the program should not be looking for any additional inputs.
- Be sure to include the specified text on each output line. That is, concatenate a *String* literal with the appropriate values/variables.

## What To Submit

Please put all your files in a folder called Assignment1. Zip the folder (DO NOT RAR it) and submit it in MyCourses. If you do not know how to zip files, please ask any search engine or friends. Google will be your best friend with this, and a lot of different little problems as well.

Inside your zipped folder, there must be the following files. **Do not submit any other files, especially .class files.** Any deviation from these requirements may lead to lost marks.

`ISBN.java`
`InputAnalyzer.java`
`PizzaCalc.java`
`Confession.txt` (optional) In this file, you can tell the TA about any issues you ran into doing this assignment. If you point out an error that you know occurs in your problem, it may lead the TA to give you more partial credit.