# ASSIGNMENT 2

COMP-202, Winter 2017, All Sections

Due: Friday, February 10$^{th}$ 2017, (23:59)

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

**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. :)

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

- Follow all directions below
- Make sure that your code compiles
  - Non-compiling code will receive a very low mark
- Write your name and student ID is written as a comment in all .java files you hand in

## 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)
Write a method `swap` which takes as input two int values x and y. Your method should do 3 things:

1. Print the value of x and y
2. Swap the values of the variables x and y, so that whatever was in x is now in y and whatever was in y is now in x
3. Print the value of x and y again.

For example, if your method is called as follows: `swap(3,4)` the effect of calling your method should be the following printing

```
inside swap:  x is:3 y is:4
inside swap:  x is:4 y is:3
```

**Warm-up Question 2** (0 points)
Consider the program you have just written. Create 2 int (integer) variables in the main method. Call them x and y. Assign values to them and call the swap method you wrote in the previous part.

After calling the `swap()` method—inside the main method— print the values of x and y. Are they different than before? Why or why not?

**Warm-up Question 3** (0 points)
Write a method that takes three integers x, y, and z as input. This method returns true if z is equal to 3 or if z is equal to the sum of x and y, and false otherwise.

**Warm-up Question 4** (0 points)
Let's write a method incrementally:

1. Start by writing a method called `getRandomNumber` that takes no inputs, and returns a random `double` between 0 (included) and 10 (excluded).
2. Now, modify it so that it returns a random `int` between 0 and 10.
3. Finally, let the method take two integers `min` and `max` as inputs, and return a random integer greater than or equal to `min` and less then `max`.

**Warm-up Question 5** (0 points)
Create a file called `Counting.java`, and in this file, declare a class called `Counting`. This program takes as input from the user a positive integer and counts up until that number. eg:

```
> run Counting 10
I am counting until 10:  1 2 3 4 5 6 7 8 9 10
```

**Warm-up Question 6** (0 points)
For this question you have to generalize the last question. The user will give you the number they want the computer to count up to and the step size by which it will do so.

```
> run Counting 25 3
I am counting to 25 with a step size of 3:
1 4 7 10 13 16 19 22 25
```

## Part 2

*The question in this part of the assignment will be graded.*

**Question 1: Craps** (100 points)

Craps is a dice game where each player bets on the outcome of the dice rolls. The goal of this question is to write several methods in order to create a program that simulates the outcome of a Pass Line Bet (see below) in a game of Craps. All the code for this question must be placed in a file named Gambling.java.

### The Pass Line Bet

Craps is a game of rounds. The first dice roll of a round is called the Come-out Roll. When a player is making a Pass Line Bet, they will place a bet before the Come-Out Roll. Depending on the result of the roll, the player might win, lose, or go to the "next stage" of the game:

- If a 7 or an 11 is rolled, then the player wins.
- If a 2, 3, or 12 is rolled, then the player loses.
- If any other number is rolled, the player must go to the next stage.

For the next stage, it is important to remember which number was rolled in the Come-Out Roll, this number is called the *point*. In the second stage, the player *will keep rolling the dice until* one of the following happens:

- A 7 is rolled, and the player loses the bet
- The *point* is rolled again, and the player wins the bet

The payout is 1:1: the players win as much as he bets. Thus, if the player bets $5 and wins he will receive an additional $5. If he loses, he will lose the entire bet.

Let's see a couple of examples:

- The result of the Come-Out Roll is a 3. → The player loses!
- The result of the Come-Out Roll is a 5 → The dice are rolled again until either a 7 or a 5 is rolled. Supposed that the results of the rolls are the following: 10, 11, 4, 7. → The player loses!
- The result of the Come-Out Roll is a 9 → The dice are rolled again until either a 7 or a 9 is rolled. Let the results of the rolls be as follow: 3, 5, 9. → The player wins!

Now that we now how the game works, let's see which methods we need to simulate the result of a Pass Line bet in a game of Craps.

### 1a. A method to simulate a dice roll

In a game of Craps, players are betting on the outcome of a roll of two six-sided dice. Write a method called `diceRoll` that simulates the roll of two six-sided dice. Such method will have to return an int between 2 and 12 (included), which is the sum of the result of rolling two six-sided dice. Notice, that to simulate the roll of two six-sided dice, you will have to generate two random numbers between 1 and 6 (both included) and sum the results together. If you have any doubts on how to achieve this, make sure you first understand the warm-up Question 4, where you are asked to build a method called `getRandomNumbers`.

## 1b. A method to simulate the second stage of the bet

Write a method `secondStage` that simulates the second stage of the Pass Line Bet. This method takes as input one integer value that corresponds to the *point*, the number rolled in the Come-Out Roll (either 4, 5, 6, 8, 9, or 10), and returns an int which will either be a 7 or the point itself depending on which one gets rolled first. Let the method print all the dice rolls on the same line until a 7 or the point is rolled. For example, if the method is called with input 5, the following might get printed on your screen:

```
4, 6, 11, 7
```

Make sure to use `diceRoll` to obtain the value of each roll.

## 1c. A method to check if the player has enough money

Write a method `canPlay` which takes two doubles as input and returns a boolean value. The first input value corresponds to the money the player has, the second corresponds to how much money the player would like to bet. A player is allowed to play **only if** he has money and he doesn't bet more than he owns. If the player is allowed to play, your method should return true. The method should return false otherwise.

For Example: `canPlay(5.25, 5)` should return `true`, while `canPlay(0.0, 2.0)` should return `false`.

## 1d. A method to simulate the Pass Line Bet

Finally, write a method `passLineBet` that simulates what happens when a Pass Line Bet is placed. This methods takes two doubles as input: the first one corresponds the total amount of money the player has, the second correspond to how much money the players decides to bet. The method will return a value of type double which corresponds to the amount of money the player has left after one round of Craps.

The method should first check if the player is allowed to play using `canPlay`. If the player is not allowed to play, your method should print out a statement informing the player he has insufficient funds to place the bet and simply return the amount of money the player has at the moment. If the player is allowed to play, then you can go ahead and simulate 1 round of Craps.

Your method should print the result of the Come-Out Roll (the first roll in a round of Craps) as well as what will happen next. For example, here is the player in the first stage of the game, the Come-Out Roll. Recall that the player wins with a 7 or 11, loses with a 2, 3, or 12, and moves to the second stage with any other number. Here are some possible outcomes:

```
A 7 has been rolled.  You win!
A 12 has been rolled.  You lose!
A 5 has been rolled.  Roll again!
```

If necessary, the second stage needs to be simulated (using the `secondStage` method) in order to determine whether the player wins or loses. If at the end of the second stage a 7 was rolled your method should print a statement informing the player he lost, if instead the point was rolled your method should print a statement informing the player he won. Use `canPlay`, `secondStage`, and `diceRoll` in order to implement the simulation of a Pass Line Bet. Remember to update the amount of money the player has left depending on whether he won or lost the bet. This number should be returned as a value of type double.

### 1e. Setting up the main method (Optional - No extra marks)

Set up the main method so that the program receives two inputs of type double via command line arguments. The first input corresponds to the money the player has, the second to the money they would like to bet. From the main method, call `passLineBet` with the appropriate inputs in order to place the bet. Make sure to output a statement informing the player with how much money he has left after his bet. These are a couple of examples of how your program should run:

```
> run Gambling 25.5 7.0
A 5 has been rolled.  Roll again!
4, 3, 12, 5
You win!
You now have:  $39.5
```

```
> run Gambling 5.5 7.0
Insufficient funds.  You cannot play.
You now have:  $5.5
```

```
> run Gambling 10 7.0
A 12 has been rolled.  You lose!
You now have:  $3.0
```

## What To Submit

You have to submit one zip file that contains your files to myCourses - Assignment 2. If you do not know how to zip files, please enquire that information from any search engine or friends. Google might be your best friend with this, and for a lot of different little problems as well.

These files should all be inside your zip. **Do not submit any other files, especially .class files**.

`Gambling.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. On the other hand, it also may lead the TA to notice something that otherwise they would not.

## Marking Scheme

Up to 30% can be removed for bad indentation of your code as well as omitting comments, coding structure, or missing files. Marks will be removed as well if the class and methods names are not respected.

**Question 1**

| | | |
|---|---|---|
| Question 1a: | 20 | points |
| Question 1b: | 35 | points |
| Question 1c: | 10 | points |
| Question 1d: | 35 | points |
| | **100** | **points** |