ASSIGNMENT 2

COMP-202, Winter 2018, All Sections

Due: Thursday, February 15^{th} 2018, (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. :)

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 method names are not **spelled and capitalized exactly** as described in this document.

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 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)
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 two 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 using `x` and `y` as input parameters.

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 (still excluded).
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 than `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 (using `args`) 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: One-armed Bandits (100 points)

A slot machine, also known as a one-armed bandit, is a gambling machine with three reels which spin when a lever is pulled (or, nowadays, when a button is pushed). A player loses or wins a pre-determined amount of money based on the pattern displayed by the machine when the reels are done spinning.

The goal of this question is to write several methods to simulate a very simplified scenario, where a player plays the slot machine until they either lose all their money or they win enough money to reach their daily goal. For example, a player might start with $ 10 in their pocket, invest $ 2.50 every round in the slot machine, and stop playing when they either have $ 20 in their pocket or less than $ 2.5.

Our Slot Machine

Our slot machine will operate in *rounds*, as described below.

Before starting to play, the player decides on how much money will be bet on each round of gambling. Let this amount be called the `roundBet`. Note that the `roundBet` is the same for each round.

They also set a goal they want to reach when playing the slot machine. The player keeps playing until they reach their goal for the day, or their money is not enough to bet on the next round.

In each round, the amount that the player is betting on that round (the `roundBet`) is subtracted from their money. Then, the slot machine will show some of the following symbols: *Cherries*, *Oranges*, *Plums*, *Bells*, *Melons*, and *Bars*.

- If the machine displays three *Bells* in a round, then the machine rewards the player with 5 times their `roundBet`.
  - For instance, if a player has $ 3 as their `roundBet`, and the machine displays three *Bells*, then the player receives $ 15.
- If the machine displays three of the same symbol (not including the *Bells*), then the amount given to the player is 3 times the `roundBet`.
- If two out of the three symbols are the same, then the amount given to the player is 2 times the `roundBet`.
- Otherwise, the machine awards no money.

Imagine there's a person with $ 10 who decides to play the slot machine. This player decides on $ 3.50 as the `roundBet` and on $ 20 as their goal. This is how playing the slot machine might look like for this player:

Example 1:

- Round 1:
  - Symbols displayed: *Bells*, *Oranges*, *Melons*
  - Money left: $ 6.50
- Round 2:
  - Symbols displayed: *Oranges*, *Bells*, *Bars*
  - Money left: $ 3.00

The game ends since the player is left with not enough money to play, since they have less than the `roundBet` of $ 3.50. The player is left with $ 3.00.

Example 2:

- Round 1:
  - Symbols displayed: *Cherries*, *Melons*, *Cherries*
    - Two symbols are the same
  - Money left: $ 13.50
- Round 2:
  - Symbols displayed: *Bars*, *Bells*, *Plums*
  - Money left: $ 10.00
- Round 3:
  - Symbols displayed: *Bells*, *Bells*, *Bells*
    - All three symbols are *Bells*
  - Money left: $ 24.00

The game ends since the player reached their goal ($ 20). The player walks away from the machine with $ 24.00.

In a class called `SlotMachine` write the following methods:

1a. A method to simulate a dice roll

To simulate the slot machine, we want each symbol to have the same probability to be displayed.

Let's start by writing a method called `diceRoll` that simulates the roll of one six-sided die. This method takes no parameters, and it must return an `integer` from 1 to 6 (included). Your method must use `Math.random()` do this.

If you have any doubts on how to write the method, make sure you first understand the warm-up Question 4, where you are asked to build a method called `getRandomNumbers`.

1b. A method to get the appropriate symbol

Write a method `getSymbol` that takes an `integer` as input and returns a `String` indicating the symbol associated to it. For this assignment, the symbols displayed by the slot machine are associated to integers as follows:

1. *Cherries*
2. *Oranges*
3. *Plums*
4. *Bells*
5. *Melons*
6. *Bars*

For instance, `getSymbol(2)` returns the `String` “Oranges”, while `getSymbol(6)` returns the `String` “Bars”.

If an `integer` is entered that is not one of the above options, the `String` “ERROR” must be returned.

1c. A method that returns the multiplier associated to a combination

Write a method `getMultiplier` that takes three `Strings` as parameters representing the symbols displayed by the slot machine. This method returns an `integer` representing the multiplier for the combination of the three input Strings.

The method should work as follows:

- If the parameters are three “Bells”, then the method must return 5.
- If the parameters are three `Strings` equal to each other, but they are not “Bells”, then the method must return 3.
- If two out of the three `Strings` are equal, then the method must return 2.
- In all other cases, the method must return 0.

For instance,

- `getMultiplier(‘‘Oranges’’, ‘‘Melons’’, ‘‘Bells’’)` returns 0,
- `getMultiplier(‘‘Bells’’, ‘‘Melons’’, ‘‘Bells’’)` returns 2, and
- `getMultiplier(‘‘Cherries’’, ‘‘Cherries’’, ‘‘Cherries’’)` returns 3.

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

Write a method `canPlay` which takes two `doubles` as parameters and returns a `boolean` value. The first parameter corresponds to money the player has in their pocket. The second parameter corresponds to the amount of money the player decided to insert in the slot machine at every round (the `roundBet`). A player is allowed to play **only if** their money is more than the `roundBet`, and the `roundBet` is greater than zero.

For example: `canPlay(5.00, -5.00)` should return `false`, while `canPlay(10.00, 2.50)` should return `true`.

1e. A method to check if the player reached their goal

Write a method `goalReached` which takes two `doubles` as input and returns a `boolean` value. The first parameter is how much money the player has in their pocket, and the second parameter is the amount of money the player has set as their goal for the day. The method should return whether or not the amount of pocket money is greater than the goal for the day.

For example: `goalReached(450.00, 500.00)` should return `false`, while `goalReached(25.00, 25.00)` should return `true`.

1f. A method to display the symbols

Write a method `displaySymbols` which takes three `String` parameters and does not return a value. These parameters represent the three symbols to be displayed to the user.

This method must print out the three symbols. For example, the simplest version may print out `Cherries Bars Bells`.

However, we encourage you to add in extra print statements to make the result more attractive and for extra practice. For example, in the sample output below, we have used a *for loop* to print out hyphens and added vertical lines between the symbols.

1g. A method to format money

Write a method `formatMoney` which takes one `double` parameter, and returns a `String`. This method will help you properly format the money values you display to the player.

If the `double` parameter is named m, then this method would contain the instruction:
`return String.format("$%.2f", m);`

1h. A method to simulate playing the slot machine

Finally, write a method `playMachine` that simulates playing the slot machine. This method takes three `doubles` as parameters. The first parameter is the amount of money the player walks up to the machine with. The second parameter is the amount of money bet for each round (the `roundBet`). And the third parameter is the amount of money the player has set as their goal for the day. The `playMachine` method will not return any value.

The `playMachine` method must first check if the player has enough money to play using the method `canPlay`. Note that the `canPlay` method accepts two parameters, and return a `boolean` value.

If the player does not have enough money to play, the method must print a statement informing the user that they do not have enough money to play and terminate here the execution of this method. If the player, on the other hand, has enough money to play, then the method checks whether the goal set by the player has already been reached. This must be done using the method `goalReached`.If the goal has indeed been reached, then the method should print out a statement informing the user about it and letting them know that no further gambling is allowed.

The `playMachine` method must only simulate the playing of the slot machine if both these checks pass. That is, to obtain and show the symbols to the player, the player must have enough money to play and the goal set should have not already been reached.

Note that the `playMachine` method will simulate rounds of gambling *until* the player has enough money to play and their goal has not been reached.

If the player is allowed to play the slot machine for this round, then the following steps must occur:

1. A message is printed, telling the player which round number they are playing. The first round is round 1.
2. The `roundBet` is subtracted from the player's current money.
3. Three symbols are generated using the two steps below:
    - First, a random number from 1-6 is generated by calling the `diceRoll` method.
    - Then, the symbol as a `String` is generated by calling the `getSymbol` method and passing the random number.
4. The three symbols are printed using the `displaySymbols` method.
5. The multiplier for the symbol combination is obtained using the `getMultiplier` method.
6. The player's money is increased by `roundBet * multiplier`.

7. The player is informed of their current money.

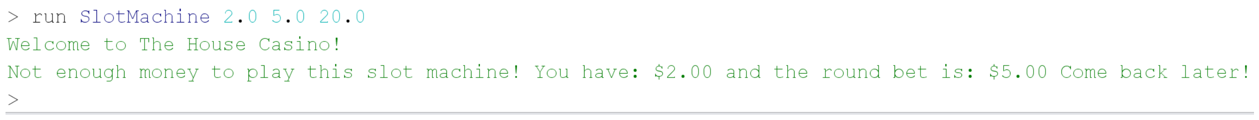
The end of the game is reached when the player runs out of money, or reaches their goal. When this happens, the method should print out whether the player lost or whether they reached their goal. In both cases, a statement informing the player of how much money they have left should be printed.

Note: Throughout the `playMachine` method, you must use the `formatMoney` method to properly format all money amounts presented to the user. For example, everytime the player's cash, goal, or `roundBet` is printed, it must first be passed to the `formatMoney` method, and the returned `String` printed.

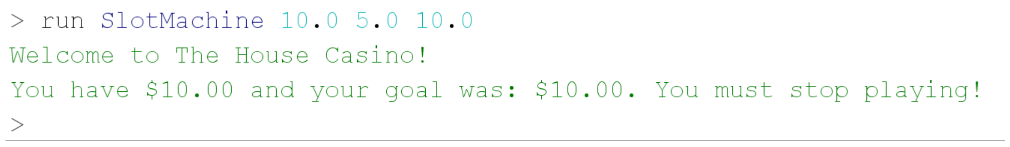
1i. Setting up the main method (Optional - No extra marks)

Set up the main method so that the program receives three `double` parameters via command line arguments. The first parameter is how much money the player has in their pocket, the second parameter is the `roundBet`, and the third parameter is the goal for the day.

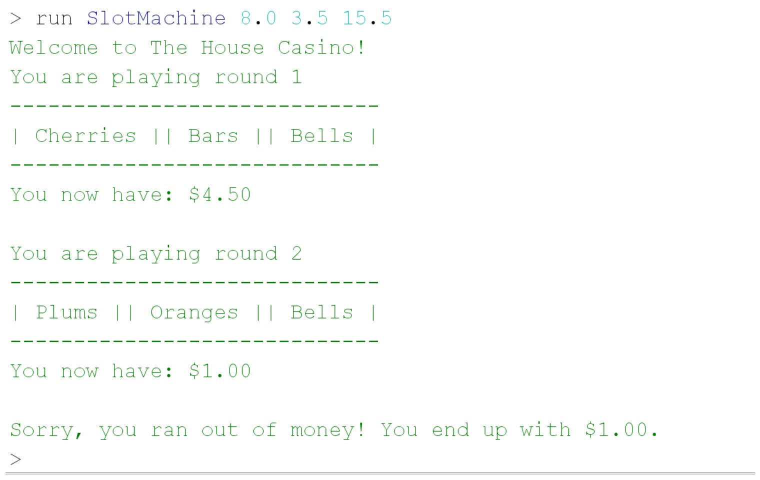
From the main method, call `playMachine` with the appropriate inputs in order to simulate the user playing the slot machine. Below are a couple of examples of how your program should run. Note that due to the use of random numbers, your output might be different.



```
> run SlotMachine 2.0 5.0 20.0
Welcome to The House Casino!
Not enough money to play this slot machine! You have: $2.00 and the round bet is: $5.00 Come back later!
>
```



```
> run SlotMachine 10.0 5.0 10.0
Welcome to The House Casino!
You have $10.00 and your goal was: $10.00. You must stop playing!
>
```



```
> run SlotMachine 8.0 3.5 15.5
Welcome to The House Casino!
You are playing round 1
------------------------------
| Cherries || Bars || Bells |
------------------------------
You now have: $4.50

You are playing round 2
------------------------------
| Plums || Oranges || Bells |
------------------------------
You now have: $1.00

Sorry, you ran out of money! You end up with $1.00.
>
```

```
> run SlotMachine 10.0 5.0 18.3
Welcome to The House Casino!
You are playing round 1
------------------------------
| Melons || Bars || Cherries |
------------------------------
You now have: $5.00

You are playing round 2
---------------------------------
| Cherries || Bells || Cherries |
---------------------------------
You now have: $10.00

You are playing round 3
----------------------------
| Bells || Melons || Bells |
----------------------------
You now have: $15.00

You are playing round 4
---------------------------------
| Oranges || Oranges || Oranges |
---------------------------------
You now have: $25.00

Congratulations, you now have $25.00! Time to celebrate!
>
```

What To Submit

Please put all your files in a folder called Assignment2. Zip the folder (please DO NOT rar it) and submit it in MyCourses. Inside your zipped folder there must be the files listed below. **Do not submit any other files, especially .class files**.

`SlotMachine.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

Question 1

Question 1a:	10	points
Question 1b:	15	points
Question 1c:	20	points
Question 1d:	5	points
Question 1e:	5	points
Question 1f:	5	points
Question 1g:	2	points
Question 1h	38	points
	100	**points**