

CS 111, Programming Fundamentals II


Homework 1: Slot Machine Simulation



Computer Science

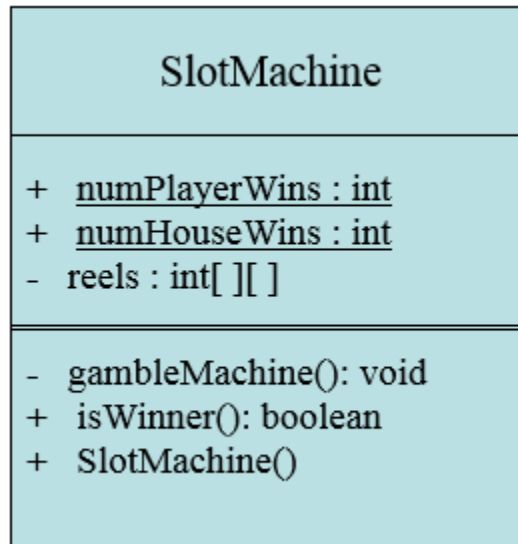
For this programming assignment you will create a simple gambling simulation. I'm sure you have heard that the odds of gambling are always in the favor of the casino. You will write a little program that will simulate the slot machines, and calculate the chance of winning on those machines. If you are not familiar with slots, they are machines with three or more reels which spin when a button is pushed. The reels have numbers or pictures of fruits written/drawn on them, and the goal is to have the reels spin, and stop, such that at least one of the 3 visible rows of numbers among the three reels are the same.

You will simulate the slot machines in a casino by writing two classes, *SlotMachine.java* and *Casino.java*. The *SlotMachine* class will have a 3x3 multidimensional array. This array will represent three reels with three spots per reel, have a look below at **Figure 1b**. A winner will be determined if any of the rows have the same values, for example 1, 1, 1 or 4, 4, 4. If a diagonal has the same values, it will count as a win as well. Have a look below at **Figure 1c-1f**, all of those are winning slots.

	<table border="1"> <tr><td>1</td><td>5</td><td>7</td></tr> <tr><td>4</td><td>3</td><td>7</td></tr> <tr><td>8</td><td>2</td><td>8</td></tr> </table>	1	5	7	4	3	7	8	2	8	<table border="1"> <tr><td>7</td><td>4</td><td>3</td></tr> <tr><td>5</td><td>5</td><td>5</td></tr> <tr><td>8</td><td>8</td><td>1</td></tr> </table>	7	4	3	5	5	5	8	8	1									
1	5	7																											
4	3	7																											
8	2	8																											
7	4	3																											
5	5	5																											
8	8	1																											
Figure 1a. A slot machine	Figure 1b. Multidimensional array representing three reels.	Figure 1c. A winning slot, the second row has all 5's																											
<table border="1"> <tr><td>7</td><td>7</td><td>7</td></tr> <tr><td>6</td><td>1</td><td>7</td></tr> <tr><td>3</td><td>7</td><td>9</td></tr> </table>	7	7	7	6	1	7	3	7	9	<table border="1"> <tr><td>1</td><td>9</td><td>3</td></tr> <tr><td>4</td><td>1</td><td>7</td></tr> <tr><td>8</td><td>5</td><td>1</td></tr> </table>	1	9	3	4	1	7	8	5	1	<table border="1"> <tr><td>2</td><td>5</td><td>3</td></tr> <tr><td>7</td><td>3</td><td>5</td></tr> <tr><td>3</td><td>8</td><td>8</td></tr> </table>	2	5	3	7	3	5	3	8	8
7	7	7																											
6	1	7																											
3	7	9																											
1	9	3																											
4	1	7																											
8	5	1																											
2	5	3																											
7	3	5																											
3	8	8																											
Figure 1d. A winning slot, the first row has all 7's	Figure 1e. A winning slot, the main diagonal has all 1's	Figure 1f. A winning slot, the counter diagonal has all 3's																											

I. *SlotMachine* Class

Implement the *SlotMachine* class according to the UML diagram below. Have a look at the slides or the book if you don't remember how to read the UML. Underlined fields or methods in a UML, specify that they are **static**. The two fields *numPlayerWins* and *numHouseWins* are both static and act as counters to keep track of wins.



The method *gambleMachine* will populate *reels*, which is a 3x3 array. Populate the array using the random number generator. Use numbers in the range between 1 and 9, including 9. The method *isWinner* will check if any of the three rows have the same values, it will also check if the one of the two diagonals has the same values. If any of the rows or the diagonals have the same numbers, it will return *true*, otherwise it will return *false*. The non-default constructor just calls the method *gambleMachine*.

II. *Casino* Class

The *Casino* class will just have the *main* method. In the *main* method create an array of type *SlotMachine* of size 1,000,000. Using a for-loop create objects of type *SlotMachine* and populate the array. Inside of the same for-loop check if the slot is a winner, if it is then increment the field *numPlayerWins*. If the slot is not a winner then increment the field *numHouseWins*. As you create objects of type *SlotMachine*, the invoked constructor will call the method *gambleMachine* automatically, all you have to do is call the method *isWinner* on each object.

After the for loop print out the number of times the casino has won and the number of times that the player has won. Also calculate and print out the chance of the player winning overall (numbersWon/totalNumbersPlayed).

Keep in mind, static fields can be called directly on the class name (*SlotMachine*). All of the instances of *SlotMachine* will share the same values for the fields *numPlayerWins* and *numHouseWins*. This is because they are static, and they belong to the class and not any specific instance of that class.

Sample output can be seen below in **Figure 2**.

```

----jGRASP exec: java Casino
The Casino won 939605 of times.
The Player won 60395 of times.
The chance of you winning is : 6.04 %.
----jGRASP: operation complete.

----jGRASP exec: java Casino
The Casino won 940065 of times.
The Player won 59935 of times.
The chance of you winning is : 5.99 %.
----jGRASP: operation complete.

----jGRASP exec: java Casino
The Casino won 939337 of times.
The Player won 60663 of times.
The chance of you winning is : 6.07 %.
----jGRASP: operation complete.

```

Figure 2: Sample Outputs

III. Upload your work to Canvas

Make sure that you upload the following files to the Homework 1 assignment in your Canvas account:

SlotMachine.java
Casino.java
Screenshot image containing an output of the program

Using the **Snipping Tool** take a screenshot of the output and save it as an image.

There will be additional files (the .class files, that you've generated when compiling your code) in your homework 1 folder, but don't upload them to Canvas. Those files are not graded; they are just the byte code files that are used by the Java Virtual Machine.

V. Rubric

File / Lab	Points
Class <i>SlotMachine</i> is implemented correctly	30
Class <i>Casino</i> is implemented correctly	30
Code is formatted and commented	7
Screenshot of the output	3
Total	70