Project 1: One Armed Bandit

Submission Due: March 19 by 11:59PM

Program Description:

For this project you will be creating a simulation of a slot machine. Slot machines in the form that we typically see in casinos and other gambling establishments, or almost every establishment in certain cities in the U.S., trace their roots back to the late 1890's in San Francisco. The "Liberty Bell", created by Charles Fey, contained three spinning reels decorated with five different symbols: bells, spades, hearts, horseshoes, and diamonds. A player would insert a nickel into the slot, hence the term slot machine, and could win a whopping 50 cents as the biggest payout[i]. The machines have progressed considerably since then and can now be found themed as many of your favorite movies, books, historical time periods, professions and the maximum payouts can be in the tens of millions of dollars.

While your simulation will not be bringing in any taxable revenue to the state, constructing many of the aspects of a slot machine should provide a complex task. Your slot machine will need to be an engaging and interactive experience for the player, just without the lights and sirens. It will need to manage the player's winnings, allow them to add additional funds and of course play the game. It is also recommended that you take some time to design your program before you begin coding to improve its overall flow and organization.

Program Requirements

- The player should start with 100.00 dollars
- The machine should start with a balance of 0.00 dollars
- The default bet should be 1.00 dollar
- The machine should have an array of seven words, of your choice, that could be selected for each of the three reels
- The set of three reels should be stored as a single, 3X3, 2-dimensional array
- Your program should print out a welcome message, explain how to win the game (see below), and inform the user of the current bet amount
- The following are possible winning combinations
    - Three of a kind on any single row, column, or diagonal (Payout is 3X the bet amount)
    - Two of a kind adjacent to each other on any single row, column, or diagonal (Payout is 2X the bet amount)
        - i.e. If the row/column/diagonal had A-A-A, it must be counted as a three of a kind and not two, two of a kinds
    - If multiple instances of three of a kind or two of a kind exist, the payouts should be summed

- Your program will need a menu, implemented as a switch, in which you provide the following options and prompt the player for which action they wish to perform with an appropriate message:
    - Add money to the machine
    - Change their bet amount
    - Play the game
    - Leave the machine and pay out all of their winnings
- Each of the four options must be implemented as its own method, which should be called via the appropriate case in the switch
    - Each method should have an appropriate name, return type, list of parameters, and comment block (multi-line comments or Javadoc comments)
- Your program will need to allow the player to make choices and play the game until they wish to leave
    - The program should inform the player of how much money they have, the machine's balance, the current bet amount, redisplay the main menu and prompt for a choice at the beginning of the game and after every main menu choice
- If the player chooses to add money to the machine:
    - Prompt the player for how much they wish to add
    - Determine if that amount is less than or equal to the amount the player current has
        - If that amount is, then add the amount to the machine and deduct it from the player
        - If it is not, then inform the player they do not have that much money
    - Display how much money the player has, and the current balance of the slot machine, with an appropriate message
- If the player chooses to change their bet amount:
    - Prompt the player for how much they would like to bet for future games (this number must be greater than 0) with an appropriate message
        - If the player's bet amount is greater than 0, inform the player that all future bets will be of this value until they change it again
        - If the player's bet amount if 0 or less, inform them of the mistake and that the bet amount remains unchanged
- If the player chooses to play the game:
    - First, determine if the machine has enough money in its balance to cover the current bet amount
        - If it does, inform the player that the game is about to start
        - If it does not, inform the player they do not have enough money to pay for the current bet amount
    - Deduct the current bet amount from the slot machine's current balance
    - Using a random number generator, choose one of the words from your array of seven to apply to each of the nine positions on the reels. Note that the same word could be chosen multiple times. These words should be displayed to the player in a pleasing grid format (i.e. all columns must be the same width without excessive white space), similar to a real slot machine.
    - If the player has at least one winning combination, as described above
        - Using an appropriate message, inform the player of each way they won, and the total payout

        - Credit the slot machine's current balance the total payout
    - If the player has no winning combination, as described above
        - Using an appropriate message, inform the player that they lost this round but to try again
- If the player chooses to leave the machine and pay out all of their winnings:
    - Using an appropriate message, inform the player:
        - How much they won or lost (i.e. their winnings minus the losses and the money they personally added to the machine)
        - That they will be paid the current balance of the slot machine and display that balance
    - Exit the game
- If the player inputs an option that is not on the menu, then inform them of their mistake with an appropriate message, redisplay the menu options, and ask for a new choice
- No global variables are allowed for this project
- Your code should be well documented in terms of comments. For example, good comments in general consist of a header (with your name, date, and brief description), comments for each variable, and commented blocks of code
- Your program source code should be named **SlotMachine.java**

Project Timing

An important aspect of project development is planning out the work, dividing the work into portions, and then estimating how long each piece will take. This strategy is quite common when using popular development strategies such as SCRUM or Extreme Programing. However, generating adequately sized portions and then estimating how long that portion will take to complete is a skill. Therefore, we will be using the projects in this course to help you develop this skill!

To make things a bit easier for this project, the work has already been divided into the following portions:

1. Initial program setup (common variables, array of seven words, 3x3 array)
2. Designing and implementing the menu and requisite control structures
3. Designing and implemented the adding money functionality
4. Designing and implemented the changing bet amount functionality
5. Designing and implemented the playing the game functionality
6. Designing and implemented the cashing out functionality

Before you start any coding, for each portion you will need to estimate how long you think implementing that functionality will take, round up to 30 minute increments, and note that down. A clever industry strategy is that once you estimate your time, you should double it. Then while you are implementing a portion, keep track of how long it takes you to successfully develop that functionality, round up to 30 minute increments, and note the amount of time down at the end. Additionally, if your actual time was under or over your estimate, briefly speculate on why that is the case. When you submit your project, you will also need to submit a PDF file containing your timing information and speculations.

NOTE: This is NOT a race! The goal is not to estimate the smallest amount of time possible and try to cram in development as fast as possible. The point is to help you keep track of how long development takes YOU, so you can better plan your schedule during your week. It is desirable to estimate your time well, but you will NOT be graded on how close your actual development time is to your estimated time.

Submission

- Your program will be graded largely upon whether it works correctly
- Your program will also be graded based upon your program style. This means that you should use comments (as directed) and meaningful variable names
- You must submit the **SlotMachine.java** file and your timing information PDF file to Canvas by the due date and time
- Projects will be accepted up to two days late, at a 25% penalty per 24 hour period after the due date
- Projects are meant to be individual coding assignments, so no collaboration is allowed. This includes downloading code off of the internet. Any discovered instances of this will be considered cheating and appropriate actions will the taken according to the course syllabus
- Be sure that you have tested the version of the program you wish to submit to make sure it works correctly. You will not be allowed to resubmit work after the deadline that does not have a third-party timestamp such as Dropbox or an email. Timestamps generated by your operating system will not be acceptable.

Rubric

The entire assignment is worth 100 points and partial credit is possible.

- **Program Executes Successfully**
  - If your program fails to compile 10 points will be deducted from the project, but I will try to fix minor issues (incorrect indentations, stray character, missing import) so that I can execute and test the program. I will not fix major issues that would require functionality to be further implemented, or a reorganization of logic in your code.
- **Setup (10 points)**
  - Program has an array that seven five different words to be used on the reels (5 points)
  - Reels are represented by a 3x3 2-dimensional array (5 points)
- **Menu (10 points)**
  - Menu is correctly implemented as a switch (5 points)
  - Program prints out the menu at appropriate times (5 points)
- **Methods (10 points)**
  - Each of the four options is implemented as its own method with appropriate parameters and return types
- **Player Adds Money (10 points)**
  - Program prompts for and stores the player's added amount (5 points)
  - Program correctly adds an amount the player has and informs the player OR rejects an amount the player does not have and informs the player (5 points)
- **Player Changes Bet (10 points)**
  - Program prompts for and stores the player's bet amount (5 points)
  - Program correctly accepts a positive bet amount and informs the player OR rejects a non-positive bet amount and informs the player (5 points)
- **Player Plays Game (25 points)**
  - A random number generator is used to select nine words to be used for the reels (5 points)
  - Program correctly handles all winning conditions (10 points)
  - Program correctly handles the losing condition (5 points)
- **Player Exits the Game (5 points)**

    - Program correctly informs the player of their winnings/losings, the current balance and informs the player they will be paid the current balance amount (5 points)
- **Program contains sufficient comments (10 points)**
- **Timing document with estimated times, actual times, and speculations (10 points)**

Bonus

For 10 bonus points, rather than having the user be informed of winning or losing a single spin of the wheel via the terminal, your program should utilize Message Dialogs, via the JOptionPane, to really get their attention! The Message Dialog should contain all of the information normally output when a player wins or loses.

---

[i] https://blackmesacasino.com/history-of-slot-machines/