**Jack-Poké-pot** **Machine**

Understand the Problem:

Jackpot / slot machines are really common in casinos, so the goal of this program is to create a fully functioning jackpot machine using GUI. This program should show your understanding about GUI, inheritance, implementations, event handling, the concept of Object and classes, try/get and exceptions. The program is consist of three class (+1 if main client is included), two of them is a JFrame class. The first frame would ask the user their name and the initial money they had, also the first frame will give a text to inform the player about the instructions. The second frame is the Jackpot machine itself, where the player can set the bet, and “pull the lever” by clicking the roll button, then the three pictures will get randomized, and inform the player whether he wins or not.

Basically it's just a pokemon themed jackpot machine with simple concepts:

* GUI (pokemon pics!)
* Action Listeners (buttons & sliders)
* Media Player (Pokemon song!)
* Inheritance (the jackpot machine and player interface)
* File Reader (to give instructions to the reader)
* Exceptions (if necessary)
* Many others if you prefer to add more functionality to the frame.

Phase 1: The Player Class : you should be able to finish this in 5 minutes!

This class is just an object that encapsulate two informations: the name of the player and the money it had, so the PlayerInterface class could pass the object to the Jackpot class. also it could be used to store the data to a .txt file that could work as a save file (we won’t do it now, but it could be an improvement for our later Jack-Poke-pot program). The same goes to compareTo() method, is not really necessary now, but it would be a great foundation if we prepare it for future upgrade.

**Member data**:

**private** String name;

**private** **int** money;

**Public methods**:

You should have at least this method in your PlayerInterface class constructor

* **Player**(String name, double money) - make use of the mutator method.
* **Player**() - default constructor, chain with the parametirzed constructor
* **Mutators**/**Accessors** - for name and money
* double **compareTo**(Player other) - returns the difference of their money

**Private methods**:

I’m not too strict about private methods, you can use as many private method as you want, as long as it’s existence is meaningful, and use it efficiently.

* **validate**(String name) - name must be between (0-100).
* **validate**(double money) - money must be between ($1 - $1,000,000). no poor player allowed to use this jackpot machine. on the other hand, $1,000,000 is a lot, if somebody got that much money, he should probably go to monte carlo rather than playing this game. (actually the money is 1mil to avoid any integer error, set it so it quite far below the int maximum number.)
* **other private methods** - up to you, as long as it existence is necessary

Phase 2: The PlayerInterface Class :

A frame that lets the player to register their name and money, and and take a look for the instructions. extends JFrame, or you could extend any class that’s a subclass of JFrame. You could add several functionalities such as playing sounds such as extending the JFrame and add a looping music background code in setVisible() method, or you could add background image to the components by extending the corresponding component and override the paintComponent() method.

**Member data**:

Player currentPlayer;

**Objects in the Frame**:

The minimum objects required in this frame:

* Two JTextFields: one for input the name and input the money
* Two JButtons: one for register, and one for load
* JLabels: as much as you want, as long as it's existence is necessary

**Public methods**:

Notice that, since it’s a frame, the most important public method is only the constructor to create the Frame, all the commands are controlled by an event handler class. You’ll create a private handler classes that control all the interactors on the frame.

* **PlayerInterface**(Player currentPlayer) - instantiate the JFrame with the member objects and a new empty Player member data. Modify the frame to look as beautiful as possible, think of yourself as an artist here, since there’ll no computation or algorithm at this part. But don’t forget to add action listeners to the interactors.

**Private methods**:

There are no requires private methods, all of these are the recommended private methods

* **readText()** - read the .txt file located in “instructions/instructions.txt” and return a huge line of string that consist everything inside that txt file

**Private Classes**:

* **A class that implements ActionListener Class:** this will control every action in your frame. In this class, override actionPerformed() method, and listen to any button that are clicked.
  + If the player clicked the instruction button e.g. (e.getSource() == buttonInstructions), then read the instructions.txt file, and show it to the player.
  + If the player clicked the play button e.g. (e.getSource() == buttonPlay), then create a new Player object that takes the two textfields as the input String and integer. If the Player object is successfully created assign the new Plyer object to the Player member data, and pass the object to a new Jackpot Frame, otherwise, show an error message to the player.

**Extra Challenge (Bonus Credit)**:

* Make the frame plays a pokemon audio, located in “audio/Pokemon\_Opening.wav” when the first time the frame was set to visible.
* Make the labels have a background image, located in “images/background.jpg”, you can do this by extending JLabel objects and override the paint component method of that class. Let’s call it ImageLabel, use that ImageLabel object as the labels instead of JLabels, voila you got background imaged labels.
* Tips: make use of Panels in the frames and try bunchload of layout managers

Phase 3: The Jackpot Class

**Member data**:

Player currentPlayer;

**private** **int** bet;

**Public methods**:

To some extent this frame is kind of similar to the PlayerInterface class. Extend JFrame, or any class that extends JFrame. This frame is the jackpot machine frame.

* **Jackpot**(Player currentPlayer) - instantiate the JFrame with the Player member data. Set the Player member data as the passed Player parameter since the PlayerInterface frame will pass a player object as a parameter. Add the JLabels, Buttons, etc. to the frame, make it as neat as possible. Remember to put an interactor that control the amount of bet. You could create a textbox that take integer value to determine the bet. Or you could use a JSlider so that if the player slide, they increase their bet while reduce their current money. Don’t forget to add ActionListeners or ChangeListeners

**Objects in the Frame**:

The minimum objects required in this frame:

* One **JButton**: the “roll” or “pull” button. This button will indicate that the three main labels will be randomized.
* Three-Five Labels: three for the main jackpot slots picture. The rest are for either showing the current money or the bet. You could also add more JLabels, if you want to add more info to the frame, or just to make the frame look more beautiful.

**Private methods**:

You could use a lot of private methods, as long as its existence is meaningful. These are the recommended methods for you to have. Note that, these methods will significantly help you, in the event handling class, since you just need to call these specific methods to get the job done.

* Icon **pull**(): a method that returns an Icon. You could use Math.random() inside this method, and use some sort of algorithm that’ll produce some Icons with specific chance of occurrence.
* int **multiplier**(): return some specific multiplier integer if the current pictures have any combination of pictures. Lets say, if the pictures are 2 charmander and one squirtle, return integer of 10. This integer will be the bonus multiplication that the player will get. return 0 if the three pictures in the Jackpot Machine doesn’t match any combination
* void **message**(): return some kind of output to show the player what happened after the pull, could be easily done with JOptionPane.

**Private Classes**:

Note that this is the requires ActionListener class, you might need other implementation of other listeners such as ChangeListener which is up to you to find out.

* **A class that implement ActionListener interface:** this will be the class that handles your button. Just like the handler in PlayerInterface class, there is only one method that you need to override, the actionPerformed() method. Handle the roll button, if the player clicked the button, randomize the three picture, calculate the current money, and give the player a message. Note that this three step is essential. You could use the help of many other private methods.

**Extra Challenge (Bonus Credit)**:

* Use a JSlider as the bet interactor.
* Use cool pokemon pictures as the three pictures.
* Change the background of the frame.

Phase 4: Main Method

Your main class should be named Foothill, and it should be really brief, give some welcoming message and create a new PlayerInterface object.