**Project Report: CS106**

This project required all the skills that I have learnt throughout the semester, therefore I found it quite interesting. Before the code, I started to create the design in paper by drawing it , so I could have a better idea of how the final outcome was going to be. So, in the left side of the frame I decided to put two JButton objects which I named Submit and NextFlag. Than in the middle, I decided to put a JLabel object which would contain the Icon inside it and the TitledBorder which would create a border around the icon and would have written on it “Flags”. On the right side, I decided to put another JLabel that would display “Select Country” words and below that the JCombobox which would list all the names of the countries vertically and they would be called items. Below JCombobox would be the JTextField object that would display Correct or Incorrect, according to the player’s selection and below that a table that would display the score. After this I created this other design in paper and I started to code.

|  |  |  |
| --- | --- | --- |
| Instance Variables | Constructor | Methods |
| TitledBorder Flags | QuizzGame() | public void actionPerformed() |
| TitledBorder ScoreTableTitle |  | pre-defined methods of Java |
| JButton Submit |  |  |
| JButton NextFlag |  |  |
| JCombobox Combobox |  |  |
| String[] CountryNames |  |  |
| boolean[] CountryDisplay |  |  |
| JTextField Text |  |  |
| JLabel Result |  |  |
| JLabel SelectCountry |  |  |
| JLabel FlagsLabel |  |  |
| JPanel left |  |  |
| JPanel middle |  |  |
| JPanel right |  |  |
| JPanel left\_right\_adjust |  |  |
| JPanel Correct\_Incorrect\_Table\_Header |  |  |
| JPanel Score |  |  |
| JLabel Correct |  |  |
| JLabel Incorrect |  |  |
| JTextField CorrectPoints |  |  |
| JTextField IncorrectPoints |  |  |
| JPanel Table |  |  |
| int Randnum |  |  |
| int i=0 |  |  |
| int CorrectP=0 |  |  |
| int IncorrectP=0 |  |  |

In the coding part, I first imported all the packages I needed for the GUI game. These packages were java.awt.\* that contained all the classes for creating user interfaces; javax.swing.\* that contained classes similar to java.awt.\* package and was needed to create most of the objects that I needed that started with the J prefix; java.awt.event.ActionListener and java.awt.event.ActionEvent that were needed for the interaction between the player and the program. The last package that I imported was javax.swing.border.TitledBorder that was needed for the creation of the border around the flag and the table that I was going to create.

After that, I created a public class called Quizz Game that would contain inside the instance variables necessary to be declared for creating the objects, a constructor that would create the objects and a method called public void actionPerformed (ActionEvent e) that would make the objects useful when the user interacted with the program. I extended the class that I created by typing public class QuizzGame extends JFrame so it would inherit all the elements of the JFrame class and added implements Action Listener which is an interface that would be notified every time the player would click a button or a menu item inside the program, so the objects would interact with each other and the user inside the actionPerformed method. Then, I created another class that would “Run” my program by inheriting all the elements of the QuizzGame class. So, I called it App\_Run that would extend the QuizzGame and then inside it I typed: “public static void main (String[] args)” and inside this main method I created the frame object since App\_Run class had inherited QuizzGame class which had inherited JFrame class. After the creation of the JFrame I used pre-defined methods to set a title to JFrame using the frame object name that I created that I called frame and by adding the method .setTitle(“String”). In the same way I added the size and I made it visible and not resizable through the methods: .setSize(700,300); .setResizable(“false”); .setVisible(true).

**Instance Variables**

After creating the frame, I went back to the Quizz Game class and started to create the instance variables that were needed to be declared to create all the objects later in the constructor part. I declared two TitleBorder objects, one named Flags that would surround the flages images and the other one named ScoreTableTitle that would surround a JPanel that I would use as a “table” where I could put the score. Then I declared two JButtons, one named Submit and the other one named NextFlag; a JCombobox named Combobox, a StringArray named CountryNames that would be put inside the JCombobox so all the Country Names would be displayed inside it in the order of the String array, a JTextField called Text that would display the message “Correct” or “Incorrect”; 5 JLabels, the first one named Result, the second one named Select Country, the third one named FlagsLabel where the Icons of the flags were going to be set; the fourth one named Correct and the fifth one named Incorrect that would be displayed above the scores that I would put inside the “table”; Two JTextFields named CorrectPoints and IncorrectPoints that were going to be put below the JLabels named Correct and Incorrect and would display the scores. Later I declared the JPanels, actually six of them named left, right, left\_right\_adjust, CorrectIncorrectTableHeader, Score and Table that I would use to arrange the objects to my desired design and I will explain their usage below, an integer that I named RandNum that would be used to display different flags every time I would press the NextFlag button. In the end I declared two other integers named CorrectP and IncorrectP that would be used as a text inside JTextFields named CorrectPoints and IncorrectPoints, so points could show up. Later I created another integer named i that would serve as a counter to close the program and a BooleanArray named CountryDisplay that would remove the flags one by one by checking which flags has been shown before.

**The Constructor**

In the constructor that I created named QuizzGame, firstly I added inside it the RandNum variable which had a piece of code behind it that would give an integer from 0 to 24 randomly and that I would use as an index to display all the flags. Then I proceeded into creating all the objects necessary using all the variables that I declared above. After creating the objects I made a little bit of research on how to put inside a JLabel, an image and the code that I used for that was: FlagsLabel.setIcon(new ImageIcon(getClass().getResource(“The relative path where the image is located/Name of the image” +Randnum+ “.gif”))). Since I named the flags: flag0,flag1….flag24 each time the Randnum changed Java would check for the resource that would change according to the randnum and would return the url for getClass(). After it found the URL, a new object called ImageIcon would be created using this URL and then this object using the method .setIcon() would be set inside the JLabel called FlagsLabel in my case. About the JTextFields CorrectPoints and IncorrectPoints that I created, I used the methods .setEditable(false), so the player could not change the actual score that would be displayed. As I mentioned above I also created a lot of JPanels and below I am explaining how I used them:

I created the first three ones that would put the objects in the left, in the middle and in the right named left,middle,right . All of them used the BoxLayout Manager and the code for that is: JPanel’s Name.setLayout(new BoxLayout( JPanel’s Name(target), BoxLayout. Y\_AXIS). The reason why I put the direction in Y\_AXIS is because I wanted to arrange the objects inside each panel one after another vertically. Then I proceeded into adding in the left panel the two JButtons Submit and NextFlag using the JPanel’s Name.add(object’s name) method. I knew that the order of the buttons would be Submit and NextFlag under it but the problem was that they were next to each other and I did not want that in my design. I wanted all my objects separated so I made a little bit of research and I found that between every object inside JPanel I could add inside them an invisible component from pre-defined class named Box that could affect my layout by creating some space, so I typed between these two objects added inside left JPanel: left.add(Box.createRigidArea(new Dimension(0,5)). This piece of code shows that an invisible box shape component will be created from the Class Box with (x,y) dimensions so my invisible “box” area would create a vertical space between my two objects since x dimension is 0. Since I wanted these two buttons a little bit more down from the top I added the same code above the left.add(Submit). Then I set the TitleBorder for the FlagsLabel using the method JLabel’s name.setBorder(TitleBorder object that I created named flags). After that, I put the FlagsLabel inside the JPanel named middle. Using the same idea as I used in the left JPanel, I created the right one where I put the objects named SelectCountry, Combobox, Text and Table. The way how I created the Table was by creating two JPanels first, the first one named CorrectIncorrectTableHeader where I put JLabels named Correct and Incorrect and in this JPanel I created again a Rigid Area between the JLabel objects but this time the area would be created horizontally so I wrote: CorrectIncorrectTableHeader.add(Box.createRigidArea(new Dimension (10,0)) between the two added objects of this JPanel. Than I created another panel called Score where I added the TextFields named CorrectPoints and IncorrectPoints. I did the same thing with them for creating a space, just like in the CorrectIncorrectTableHeader panel. Proceeding, I created another JPanel named Table and after that I set a BoxLayout Manager that would arrange the objects vertically, in this case the CorrectIncorrectTableHeader and Score so the objects would be arranged as I wanted. For this I wrote the code: **Table.setLayout(new BoxLayout(Table,BoxLayout.*Y\_AXIS*));**

Than in this big panel that contained the two previous ones that I added, I would set a Border with the object ScoreTableTitle that would display a border around the big panel that would had written in it “Score Table”. After the creation of this Panel that supposedly was going to be a “score table” I added it inside the right JPanel. In the end, to put the left,middle,right panel in the left, middle ,right order I created another JPanel named left\_right\_adjust that also would arrange the panels in the X-Axis so I added the left panel, than the middle one and in the end the right one. Between all of them I created a space using the Box.createRigidArea method(new Dimension()) so between the panels would have spaces.

Later I added the FlowLayout Manager in the ContentPane() using the code: getContentPane().setLayout(new FlowLayout());so the objects would fit equally inside the frame than I added in the ContentPane the left\_right\_adjust JPanel that had all the panels and objects that we created inside it, so everything would display nicely.

In the QuizzGame constructor I also added actionListeners to three objects that were the two JButtons and the JCombobox. The actionListener were needed for these three objects since they would interact with each other and with the player. The code that I provided for this thing was:

Submit.addActionListener(this);

NextFlag.addActionListener(this);

Combobox.addActionListener(this);

In the end I put ActionCommands in the JButtons, so we could use them in the ActionPerformed part when the user clicked on them and I disabled the NextFlag button. The code for this part was:

Submit.setActionCommand("Submit");

NextFlag.setActionCommand("NextFlag");

NextFlag.setEnabled(false);

**public void actionPerfomred(ActionEvent e)**

I used this abstract class in order to make the objects useful and to achieve what the project wanted. The first statement that I wrote was:

**if(e.getActionCommand().equals("Submit") && CountryNames[Randnum].equals(Combobox.getSelectedItem())).**

So if the player would press the submit button and the String Array named CountryNames would have the same “String” chosen in the Random positon as the String item that the user would select from the Combobox it would display the correct message. Since I arranged the Strings to have the same indexes as the flags, the Strings in the StringArray would coincide with the flag images that were needed to be found when the flags were displayed. In our case we knew what the Randnum was since the random number was stored in RandNum. This Randnum would serve as index for the CountryNames array, so if the user selected the right flag name it would coincide with the right flag image index, therefore with the right selected String from the array and the Correct message would be displayed. So, in a way the flags images would help us to find the Randnum in order for the if statement to be completed. Then, I wrote:

**Text.setText("Correct");**

**NextFlag.setEnabled(true);**

**CountryDisplay[Randnum]=true;**

**i++;**

**CorrectP++;**

**CorrectPoints.setText(Integer.*toString*(CorrectP));**

**Submit.setEnabled(false);**

So, when all the conditions of the above statement would be completed I would put inside the JTextField the Correct message, would enable the NextFlag button because I I had it disabled in the constructor part other. The reason why I created the boolean array named CountryDisplay was to check for the correct flags that were shown before and not to show them anymore. By putting every boolean data type to false inside an array, I was able to turn them to true one by one every time this if condition was completed, so the program would check which boolean was false and would turn it to true by not creating the same randnum again and in this way not diplaying the same image. After that, I increased the i counter by one everytime the conditions of this if statement was true so when the user found all 25 flags correctly the program would close, than I increased the integer that I had declared on the top of the program named CorrectP so it would increase the correct points by one in the score table that I had created every time the condition was completed, then I put this integer inside the JTextField CorretPoints using the setText method but I firstly converted it into String in the parantheses part since in the JTextField you are allowed to setText only to String. After that I disabled the Submit button so the user could not have the chance to press submit as many as he wanted and increase correct points in the score table.

Than, I continued the code with:

**else if(e.getActionCommand().equals("Submit") && !CountryNames[Randnum].equals(Combobox.getSelectedItem()))**

**{**

**Text.setText("Incorrect");**

**NextFlag.setEnabled(false);**

**IncorrectP++;**

**IncorrectPoints.setText(Integer.*toString*(IncorrectP));**

**}**

This piece of code does the contrary of the if condition above. If the user presses the submit button and the JCombobox Item that he selects does not match the selected String from the Randnum index of the CountyName, the Incorrect message would be displayed. Between the brackets I put the Text to Incorrect, NextFlag to false so the user could not switch to the next image till he found the correct one, I increased the Integer IncorrectP by one so every time this if statement gets completed the incorrect points will be increased by one in the table. In order for the point to show I put them inside the JTextField named IncorrectPoints using the .setText(Integer.toString()) method.

Than I provided the code when the user pressed NextFlag:

**if (e.getActionCommand().equals("NextFlag")**

**{**

**Combobox.removeItem(Combobox.getSelectedItem());**

**while(CountryDisplay[Randnum]==true)**

**{**

**Randnum = (int)((Math.*random*()\*25));**

**}**

**FlagsLabel.setIcon(new ImageIcon(getClass().getResource("flag"+Randnum+".gif")));**

**NextFlag.setEnabled(false);**

**Submit.setEnabled(true);**

**}**

When the user would press the NextFlag button the selected item from the JCombobox would be removed. Of course the Next Flag button would become visible only after the first if statement was completed so the JCombobox item would be removed only if the first statement was completed and then this statement was completed. Than I needed to display the next Image, but I needed to display it only once so the boolean array named CountryDisplay came in handy. Every time I set the CountryDisplay[Randnum]==true in the first statement, the while loop would be activated in the Next Flag statement and would produce a random number completely different from the previous one because it checked which boolean was true and would not check that anymore but it would produce a Randum that would turn only the false ones to true and would serve as an index for our flags (the loop will continue after all the false Booleans are set to true). Then I wrote the code for the new image that would be displayed that was the same as the one in the constructor part so the image would change every time we pressed NextFlag, disabled the NextFlag for the same reason as mentioned above and enabled the submit button again so the user could have the chance to choose the item from the combobox and if it was correct it would be disabled again and the next flag button would be enabled. This process would continue till all the flags were correctly found.

In the end I used the i counter to exit the program.

When the first condition would complete 25 times, it would display in the console part that the game was finished and the mistakes that the user had done during the game. Than it would exit the program. For this thing I provided this code:

**if (i==25)**

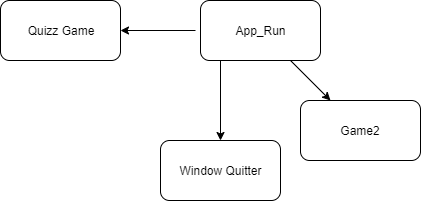
**{**

**System.*out*.print("Game over! You made " +IncorrectP+ " mistakes during the game.");**

**System.*exit*(0);**

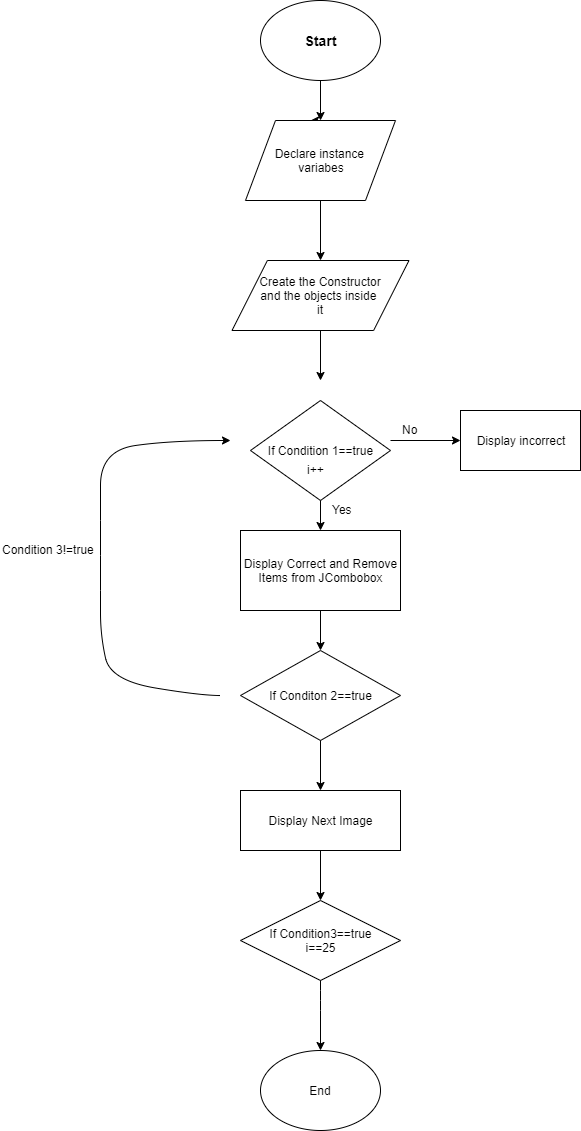
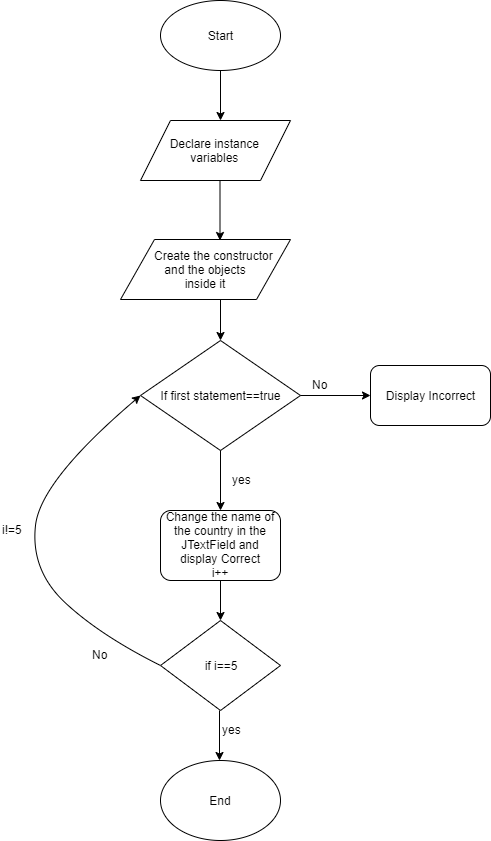
**}**

**Diagram of inter-dependences for classes:**



**Flowchart on the left-First Game**

**Flowchart on the right-Second Game**

­

**Denis Lilaj**