**Euchre Project Part 5**

**Due, Sunday, July 22, 2018 for maximum 100**

**Monday, July 23, 2018 for maximum 90**

**Tuesday, July 24, 2018 for maximum 80**

**Wednesday, July 25, 2018 for maximum 70**

**Assignment Scope**

For this assignment students will continue developing the front end, or User Interface, portion of the Euchre game. Based on my professional experience working in the industry I have ***always*** had to develop a UI for every application, therefore I translate that experience to students so they can have the same opportunity and be prepared professionally.

Typically, there is a one-to-one correlation of back end functionality to front end UI component. Depending upon the design of the application it doesn’t always correlate perfectly, however with Euchre, it works well.

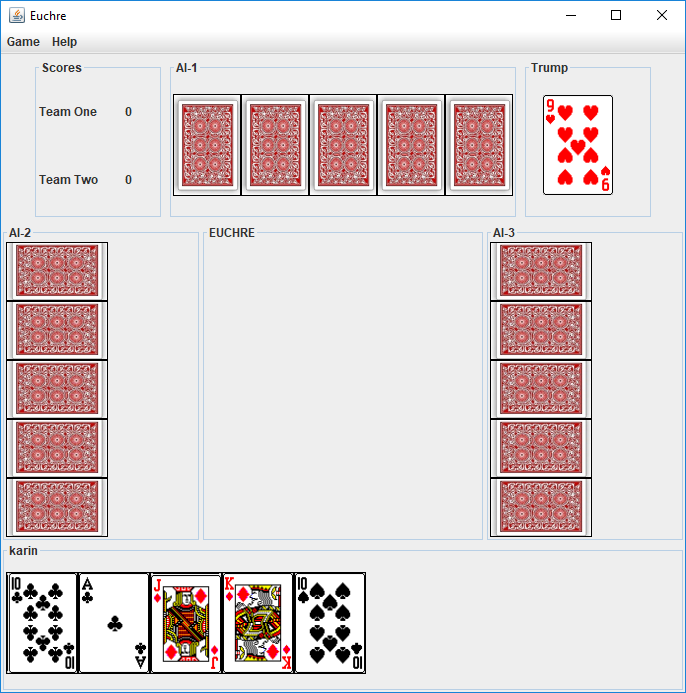
|  |  |
| --- | --- |
| **Back-end functionality** | **Front-end UI component** |
| AiPlayer.java | AiPlayerUi.java |
| Card.java | CardUi.java |
| Game.java | GameUi.java |
| HumanPlayer.java | HumanPlayerUi.java |

The goal is to develop the front-end components of the game Euchre by creating or updating classes:

1. AiPlayerUi.java
2. GameUi.java
3. HumanPlayerUi.java
4. CardUi.java

The UI will be developed in multiple assignments, it is ***not*** expected that for Part 5 the fully functioning UI is complete.

The image that follows is a what my version of the UI will looks like. It does ***not*** have to be an exact match. The rubric will provide guidance and recommendations on how to accomplish this, however feel free to be creative in developing the look and feel of the UI.



To accomplish this:

1. Reference the tasks below for the specifics of the source code requirements.
2. Compress a project and submit to Webcourses
3. Decompress compressed project and verify it is a Netbeans project

**References**

1. Netbeans.docx
2. Setting up a project in Netbeans.docx
3. Netbeans right click menu help.docx

**Deliverables**

To complete this assignment, you must submit your **compressed Netbeans project** to Webcourses.

Please keep in mind that the tasks are guidance to accomplish the goals of the assignment. At times students will be required to do additional research (e.g. Google That S\*\*T (GTS)!) to find implementation options. In the industry, software engineers are **expected** to be very self-sufficient to find the best solution for the task at hand.

***I have provided multiple code examples on Webcourses that shows how to implement numerous of the tasks below, please reference those code examples prior to asking me for help!***

**Tasks and Rubric**

|  |  |  |
| --- | --- | --- |
| Activity | | |
| euchre package |  |
| core package |  |
| Game.java | 1. Update class to add the following member variable    1. GameUi ui |
|  | 1. Add method setGameUi to do the following    1. Return type void    2. One parameter of data type GameUi    3. Set member variable ui equal to parameter 2. Add method getTeams to do the following    1. Return type ArrayList<Team>    2. Empty parameter list    3. Returns the member variable of data type ArrayList that contains elements of class Team 3. Add method getTrump to do the following    1. Return type Card    2. Empty parameter list    3. Returns the member variable of data type Card that represents the trump card for the dealt hand |
| Player.java | 1. Update class to add the following member variable    1. JPanel ui 2. Generate a getter/setter for the member variable ui |
| images package | 1. Add package images to the project 2. Add images for the cards to the package |
| userInterface package |  |
| GameUi.java | 1. Update the class to add the following member variables    1. JPanel trumpPanel (if you added bidPanel in the last assignment, please rename it)    2. JLabel teamOneScoreLbl    3. JLabel teamOneScore    4. JLabel teamTwoScoreLbl    5. JLabel teamTwoScore    6. JLabel trumpCard |
|  | 1. Update initComponents to do the following    1. Instantiate the member variable of class ArrayList containing elements of class JLabel that represents the card each player plays on the table JPanel    2. Loop through the four players and do the following       1. Instantiate an instance of class JLabel       2. Based on the position of the player set the size of the JLabel using class Dimension to be horizontal or vertical width and height       3. Instantiate an instance of class CardUi passing the JLabel from step 1 as an argument       4. Set the instance of class JLabel from step 1 equal to method call getLabel() on class CardUi       5. Add the instance of class JLabel from step 1 to the member variable of call ArrayList containing element of class JLabel representing the cards played on the table    3. Add each instance of class JLabel stored in the ArrayList containing elements of class JLabel representing the cards played so that       1. Human player JLabel for card played is in the SOUTH       2. The remaining AI players have a JLabel placed in from of their position at the table |
| AiPlayerUi.java | 1. Update the class to add the following member variables 2. CardUi cardUi; 3. int width; 4. int height; |
|  | 1. Update method initComponents() to do the following    1. Set the member variables width and height based on the orientation of the cards, vertical or horizontal |
|  | 1. Update method displayCards() to do the following    1. In the for loop       1. Instantiate the instance of class CardUi passing as arguments          1. The instance of class Card in the player’s hand at the looping variable’s index          2. The instance of class JLabel created inside the for loop          3. The Player’s position at the table       2. Set local variable of class JLabel representing the card equal to method call getLabel() on class CardUi       3. Comment out the call to method setText() on the instance of class JLabel |
| HumanPlayerUi.java | 1. Update the class to add the following member variables 2. CardUi cardUi; |
|  | 1. Update method displayCards() to do the following    1. In the for loop       1. Instantiate the instance of class CardUi passing as arguments          1. The instance of class Card in the player’s hand at the looping variable’s index          2. The instance of class JButton created inside the for loop       2. Set local variable of class JButton representing the card equal to method call getButton() on class CardUi       3. Comment out the call to method setText() on the instance of class JButton |
| CardUi.java | 1. Add member variables    1. Card card;    2. ImageIcon imageIcon;    3. JButton button;    4. JLabel label;    5. int position; |
|  | 1. Generate getters for member variables    1. Of class JButton    2. Of class JLabel |
|  | 1. Write a custom constructor that do the following    1. Receives two parameters       1. Class Card       2. Class JButton    2. Set member variable of class Card to the passed parameter    3. Set member variable of class JButton to the passed parameter    4. Call method selectFrontImage() |
|  | 1. Write a custom constructor that does the following    1. Receives three parameters       1. Class Card       2. Class JLabel       3. Int    2. Set member variable of class Card to the passed parameter    3. Set member variable of class JLabel to the passed parameter    4. Set member variable of primitive data type int to the passed parameter    5. Based on the position of the player call method selectVerticalBackImage() or selectHorizontalBackImage() |
|  | 1. Write a custom constructor that does the following    1. Received one parameter of class JLabel    2. Set member variable of class JLabel to the passed parameter    3. Calls method selectHorizontalBackImage() |
|  | 1. Write method selectFrontImage() to do the following    1. Instantiate an instance of class String to represent the file name of the image file set equal to explicit text “../images/”    2. Write a switch statement that evaluates the face of the instance of class Card       1. Based on the naming convention of your image files, concatenate to the String instantiated in step a with the appropriate value       2. For example, I named my files as such “AceHearts”, “AceEuchre”, “AceDiamonds”, “AceClubs”       3. Therefore, I concatenate to the String instance “../images/” the explicit text “Ace”    3. Write a switch statement that evaluates the suit of the instance of class Card       1. Based on the naming convention of your image files, concatenate to the String instantiated in step a with the appropriate value    4. Concatenate the explicit file extension to the String instance that represents the filename based on your naming convention, the String instance should look something like “../images/AceClubs.png” when finished    5. Instantiate an instance of class URL set equal to the static method call getClass().getResource() passing the String instance as an argument    6. Using exception handling (i.e. in a try block)       1. Check if the instance of class URL is not null          1. If true, Instantiate the member variable of class ImageIcon passing the instance of class URL as an argument          2. Instantiate the member variable of class JButton passing the instance of class ImageIcon as an argument    7. In the catch block       1. Output an error message that the resource if not found       2. Set the instance of class ImageIcon equal to null |
|  | 1. Write method selectVerticalBackImage() to do the following    1. Instantiate an instance of class String set to explicit text “../images/backVertical.jpg” or whatever you named the back image of the card    2. Instantiate an instance of class URL set equal to the static method call getClass().getResource() passing the String instance as an argument    3. Using exception handling (i.e. in a try block)       1. Check if the instance of class URL is not null          1. If true, Instantiate the member variable of class ImageIcon passing the instance of class URL as an argument          2. Instantiate the member variable of class JLabel passing the instance of class ImageIcon as an argument    4. In the catch block       1. Output an error message that the resource if not found    5. Set the instance of class ImageIcon equal to null |
|  | 1. Write method selectHorizontalBackImage() to do the following    1. Instantiate an instance of class String set to explicit text “../images/backHorizontal.jpg” or whatever you named the back image of the card    2. Instantiate an instance of class URL set equal to the static method call getClass().getResource() passing the String instance as an argument    3. Using exception handling (i.e. in a try block)       1. Check if the instance of class URL is not null          1. If true, Instantiate the member variable of class ImageIcon passing the instance of class URL as an argument          2. Instantiate the member variable of class JLabel passing the instance of class ImageIcon as an argument    4. In the catch block       1. Output an error message that the resource if not found       2. Set the instance of class ImageIcon equal to null |
| Euchre application |  |
|  |  |
| Test Cases | Test Cases pass |
|  | Source compiles with no errors |
|  | Source runs with no errors |
|  | Source includes comments |
| Total |  |

**Perform the following test cases**

|  |  |  |
| --- | --- | --- |
| Test Cases | | |
|  | **Action** | **Expected outcome** |
| Test Case 1 | **Rregression Testing: Initial JOptionPane displays** | JOptionPane is similar to figure 1 |
| Text Case 2 | **Regression Testing: JOptionPane Prompts User Name** | JOptionPane is similar to figure 2 |
| Test Case 3 | **Euchre Initial UI displays** | Game UI looks similar figure 3 where:   1. The human player’s cards show the appropriate image associated with each card in their hand 2. The AI players’ cards show a back image 3. The table has a place holder for each player when they play a card |
| Test Case 4 | **Project view** | Project view matches figure 4 |

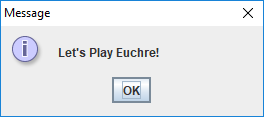


Figure 1 Intro JOptionPane

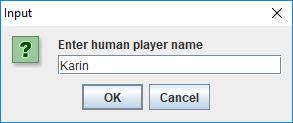


Figure 2 Prompt for User’s Name

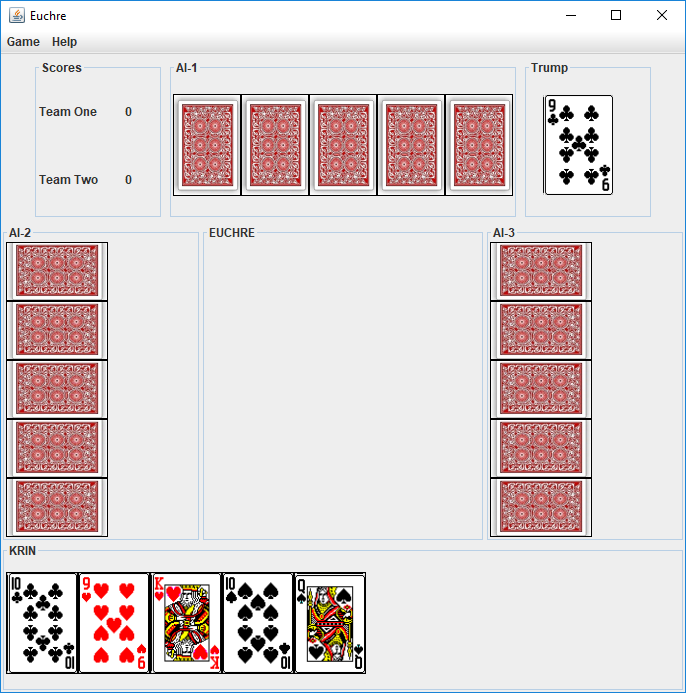


Figure 3 Euchre Initial UI

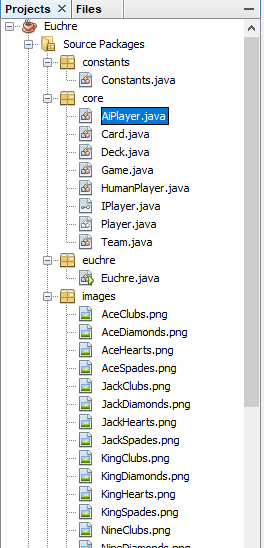


Figure 4 Project View