The University of Hong Kong

Department of Computer Science

COMP2396 Object-oriented Programming and Java

Assignment 4

Deadline: 11:59pm, 22nd Nov, 2024.

Overview

This assignment tests your understanding of GUI and event-handling, and their implementations in Java. You are going to design and implement a GUI for the Big Two card game you developed in assignment 3. In addition to the classes and interface provided in assignment 3, a CardGameUI interface is provided to model a user interface for a general card game. You may refer to their Javadoc for details of these classes and interfaces. You should **NOT** modify any of these provided classes and interfaces in completing your assignment.

You will be reusing all the classes implemented in assignment 3. You are required to implement a BigTwoGUI class which builds a GUI for the game and handles all user actions. This class implements the CardGameUI interface and has several inner classes responsible for rendering the user interface and handling user actions. To make use of the BigTwoGUI class, you will need to modify your BigTwo class accordingly. You will need to replace the BigTwoUI object with a BigTwoGUI object. You are free to introduce new instance variables and methods to these classes. Besides, you are also free to design and introduce new classes in the inheritance trees as appropriate. With a proper OO design, you do not need to touch the rest of the classes in assignment 3. You are required to write Javadoc for all public classes and their public class members.

Specifications

Graphical user interface

You are free to design a GUI for your Big Two card game. As a minimum requirement, your GUI should

- Have a panel showing the cards of each player as well as the cards played on the table. The cards should be shown in a partially overlapped manner (see Figure 1).
- For each player, the panel should show his/her name and an avatar for him/her.
- For the active player, the panel should show the faces of his/her cards.
- For other players, the panel should show only the backs of their cards.
- For cards played on the table, the panel should show at least (the faces of) the last hand of cards played on the table and the name of the player for this hand.
- Allow the active player to select and deselect his/her cards by mouse clicks on the cards. The selected cards should be drawn in a "raised" position with respect to the rest of the cards (see Figure 1).
- Have a "Play" button for the active player to play the selected cards. Nothing should happen if the "Play" button is clicked with no card selected.
- Have a "Pass" button for the active player to pass his/her turn to the next player.
- Have a text area to show the current game status as well as end of game messages.
- Have a text area showing the chat messages sent by the players.
- Have a text input field for the active player to send out chat messages.

- Have a “Restart” menu item under a “Game” menu for restarting the game.
- Have a “Quit” menu item under a “Game” menu for quitting the game.
- Your application window should support maximizing, minimizing, and resizing.

The BigTwoGUI class

The BigTwoGUI class implements the CardGameUI interface. It is used to build a GUI for the Big Two card game and handle all user actions. Below is a detailed description for the BigTwoGUI class.

Specification of the BigTwoGUI class:

public constructor:

`BigTwoGUI(BigTwo game)` – a constructor for creating a BigTwoGUI. The parameter `game` is a reference to a Big Two card game associated with this GUI.

*private instance variables: **

`BigTwo game` – a Big Two card game associated with this GUI.

`boolean[] selected` – a boolean array indicating which cards are being selected.

`int activePlayer` – an integer specifying the index of the active player.

`JFrame frame` – the main window of the application.

`JPanel bigTwoPanel` – a panel for showing the cards of each player and the cards played on the table.

`JButton playButton` – a “Play” button for the active player to play the selected cards.

`JButton passButton` – a “Pass” button for the active player to pass his/her turn to the next player.

`JTextArea msgArea` – a text area for showing the current game status as well as end of game messages.

`JTextArea chatArea` – a text area for showing chat messages sent by the players.

`JTextField chatInput` – a text field for players to input chat messages.

** These are just suggestions to aid your design. It is perfectly fine if your actual implementation deviates from these suggestions.*

CardGameUI interface methods:

`void setActivePlayer(int activePlayer)` – a method for setting the index of the active player (i.e., the player having control of the GUI).

`void repaint()` – a method for repainting the GUI.

`void printMsg(String msg)` – a method for printing the specified string to the message area of the GUI.

`void clearMsgArea()` – a method for clearing the message area of the GUI.

`void reset()` – a method for resetting the GUI. You should (i) reset the list of selected cards; (ii) clear the message area; and (iii) enable user interactions.

`void enable()` – a method for enabling user interactions with the GUI. You should (i) enable the “Play” button and “Pass” button (i.e., making them clickable); and (ii) enable the BigTwoPanel for selection of cards through mouse clicks.

`void disable()` – a method for disabling user interactions with the GUI. You should (i) disable the "Play" button and "Pass" button (i.e., making them not clickable); and (ii) disable the BigTwoPanel for selection of cards through mouse clicks.

`void promptActivePlayer()` – a method for prompting the active player to select cards and make his/her move. A message should be displayed in the message area showing it is the active player's turn.

*inner classes: **

`class BigTwoPanel` – an inner class that extends the JPanel class and implements the MouseListener interface. Overrides the `paintComponent()` method inherited from the JPanel class to draw the card game table. Implements the `mouseReleased()` method from the MouseListener interface to handle mouse click events.

`class PlayButtonListener` – an inner class that implements the ActionListener interface. Implements the `actionPerformed()` method from the ActionListener interface to handle button-click events for the "Play" button. When the "Play" button is clicked, you should call the `makeMove()` method of your BigTwo object to make a move.

`class PassButtonListener` – an inner class that implements the ActionListener interface. Implements the `actionPerformed()` method from the ActionListener interface to handle button-click events for the "Pass" button. When the "Pass" button is clicked, you should call the `makeMove()` method of your BigTwo object to make a move.

`class RestartMenuItemListener` – an inner class that implements the ActionListener interface. Implements the `actionPerformed()` method from the ActionListener interface to handle menu-item-click events for the "Restart" menu item. When the "Restart" menu item is selected, you should (i) create a new BigTwoDeck object and call its `shuffle()` method; and (ii) call the `start()` method of your BigTwo object with the BigTwoDeck object as an argument.

`class QuitMenuItemListener` – an inner class that implements the ActionListener interface. Implements the `actionPerformed()` method from the ActionListener interface to handle menu-item-click events for the "Quit" menu item. When the "Quit" menu item is selected, you should terminate your application. (You may use `System.exit()` to terminate your application.)

* *These are just suggestions to aid your design. It is perfectly fine if your actual implementation deviates from these suggestions.*

Sample output

Figure 1 shows an example of GUI for the Big Two card game that satisfies the minimum requirement[1]. You are not required to reproduce the same GUI exactly.

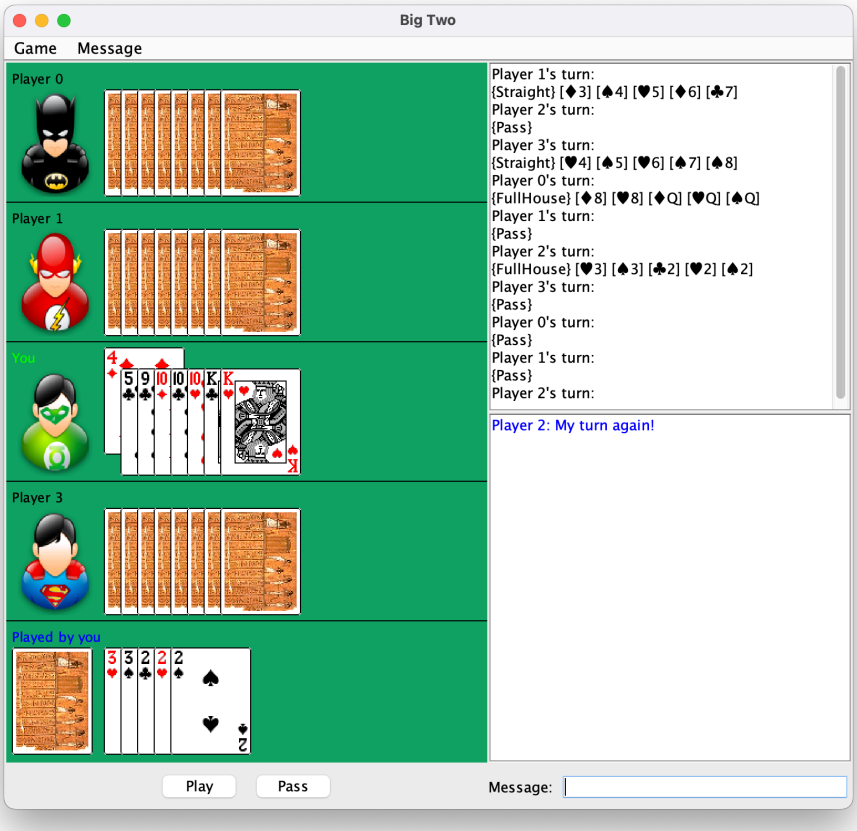

Figure 1. An example of GUI for the Big Two card game.

Marking Scheme

Marks are distributed as follows:

- Implementation of the BigTwoGUI class
  - Rendering of players' cards and cards on the table (20%)
  - Implementation of card selection using mouse clicks (20%)
  - Implementation of the game message area (5%)
  - Implementation of the chat message area (5%)
  - Implementation of the chat input field (5%)
  - Implementation of the "Play" button (5%)
  - Implementation of the "Pass" button (5%)
  - Implementation of the "Restart" menu item (5%)
  - Implementation of the "Quit" menu item (5%)
  - Overall design of the GUI (10%)
- Integration with the BigTwo class (5%)
- Javadoc and comments (10%)

[1] You may download the card images used in this example from https://www.waste.org/~oxymoron/cards/. You are free to use any other card images.

Submission

Please pack the source code (`*.java`) and images of your application into a single zip file, and submit it to the course Moodle page.

A few points to note:

- Always remember to write Javadoc for all public classes and their public class members.
- Always remember to submit the source code files (`*.java`) but **NOT** the bytecode files (`*.class`).
- Always double check after your submission to ensure that you have submitted the most up-to-date source code files.
- Your assignment will not be marked if you have only submitted the bytecode files (`*.class`). You will get zero mark for the assignment.
- Please submit your assignment on time. Late submission will not be accepted.

~ End ~