CS 342 **Project#2** **Spring 2019**

Pitch JavaFX GUI

Due Dates:

- **Part 1:** Sunday February 24th 2019, @ 11:59pm
- **Part 2:**Tuesday March 5th 2019, @11:59pm

Description:

In this project, you will implement the card game Pitch in Java and create a GUI utilizing JavaFX. We will follow the rules of the game found here: Pitch
This project will be completed in two parts. Part 1 will be a UML like schematic detailing your classes and interfaces as well as the data members and methods in each.
Part 2 will be the implementation of your project along with JUnit 5 test code and a short description of your AI algorithms.

Implementation Details:

Your application will start with a welcome screen allowing the user to chose between 2,3 and 4 players. There will be a button to start the game and a button to exit. Exit will close the application and start will start the game and load a new screen. Note that the user will always play against computer opponents.
The game play screen will show the users six cards after they are dealt and a blank field where the cards played will be displayed for every turn. The game play screen must also show the players bids for the round once they are made as well as current points. It must include a mechanism for the user to bid. It must have buttons to play the next hand, exit the game and start a new game. Exit will bring the user back to the welcome screen and start a new game will reset the current game to the beginning with a newly shuffled deck.

- The user will always get to bid first by convention.
- All games will be played to 7
- In case of a tie score at 7, game still ends with multiple “winners”
- At the end of the game, user can press exit of start new game buttons.

You are responsible for the AI algorithm for bidding and playing of the computer opponents. For computer opponents, the AI algorithm must evaluate it’s hand and produce a bid based on that hand and bids already made. It can not just do the same thing every time like “pass”. For game play, it must deploy a reasonable strategy based on what it has in it’s hand and what is played.

At the end of each hand, your program should display who won High, Low, Jack and Game. If one of the players has won, a message should display and the user can chose to exit or start a new game. If no winner, the user will press next hand button to continue play which will clear the bids from the last hand, clear the cards played for that hand and update the scores.

Your implementation must include, at least, these 6 classes:

- **Card**: a class to represent a single playing card
- **Deck**: a class that represents a single deck of cards
- **Player:** a class that represents a player in this game
- **AIPlayer**: a class that represents a computer opponent and extends Player class
- **Pitch**: a class that represents a game of pitch should implement Interface DealerType.
- **Pitch Dealer**: This class will implement the Dealer interface and represent how to deal cards for the game pitch

Your implementation must also include these two interfaces:

```
interface DealerType{
        public Dealer createDealer()
}

interface Dealer{
        public ArrayList<Card>  dealHand()
}
```

Note: The purpose of the two interfaces is that different card games are dealt differently at the beginning (BlackJack each player starts with two cards and in Pitch it's six). A deck of cards is independent of how it will be dealt. However, every card game needs a deck of cards and might need to deal them differently. These interfaces work in the same way as the Generic List Iterator code we did in class and follow the same design pattern. You do not have to use ArrayList to return a hand of cards but you will most likely need some java collection to do so.

UML Diagrams: Part 1

You are required to include a "UML like" schematic of your classes and interfaces. This schematic must include the data members and methods in your classes as well as the access modifiers of each. Submit as a pdf to the link on Blackboard.

CS 342 Project#2 Spring 2019

Game Play: Part 2

The program starts with the user choosing how many players the game will have. Once the user presses the start button, the game starts and the user gets dealt a hand which is displayed. The opponents are dealt hands as well but are not displayed. The user gets to bid first. Whomever wins the bid will play a card first. The GUI will allow the user to see when it's his/her turn. Each card played per hand will be displayed. At the end of each hand, there will be a display as to who won High, Low, Jack and Game. If there is a winner, the user can chose to exit or start a new game. Otherwise, the user can press next hand which will clear the bids from the last hand, the cards played will be cleared and scores will be updated and new hands will be dealt. If there are not enough cards left in the deck to play another hand, the deck will be reshuffled.

Testing Code: Part 2

You are required to include a folder containing the JUnit 5 test suite you created to test the classes and methods in your program. A minimum of five tests per class although some of your classes should contain a lot more.

AI Algorithms: Part 2

You are required to include a brief description (one page or less) of how your AI algorithms decide on a bid and play a hand. Format this as a PDF and include in the .zip file with your source code.

How to Start:

1. To start this project, I suggest you play the game and learn the ins and outs. It's a simple game but there is some strategy involved in both bidding and playing. You will need to know some of those strategies to program the AI algorithms.
2. On paper, draw out the classes and interfaces you are going to use, including data members and methods contained in each **before** you start coding. To program this game, there are a lot of moving parts to keep track of. Think of your UML diagram as your game plan.
3. You have a lot of discretion as to the look and feel of your game. Research and test JavaFX components that you think will create the effect your shooting for. Writing simple examples will help you see how you can incorporate them into your project.
4. Write test cases and test your classes and methods as you go. Implementing things one at a time.
5. Start now! This project will take time to implement and most of you will have many questions along the way.

CS 342 Project#2 Spring 2019

Electronic Submission:

Part 1: submit a pdf to the link on the Blackboard course website.

Part 2: You will zip all source code files and image files needed to run your application, the test suite folder and AI description pdf file and submit to the link on the Blackboard course website.

Assignment Details:

Late work **is accepted**. You may submit your code up to 24 hours late for a 10% penalty and 24 to 48 hours late for a 20% penalty. Anything later than 48 hours will not be graded and result in a zero.
Unless stated otherwise, all work submitted for grading *must* be done individually. While we encourage you to talk to your peers and learn from them, this interaction must be superficial with regards to all work submitted for grading. This means you *cannot* work in teams, you cannot work side-by-side, you cannot submit someone else's work (partial or complete) as your own. The University's policy is available here:

https://dos.uic.edu/conductforstudents.shtml.

In particular, note that you are guilty of academic dishonesty if you extend or receive any kind of unauthorized assistance. Absolutely no transfer of program code between students is permitted (paper or electronic), and you may not solicit code from family, friends, or online forums. Other examples of academic dishonesty include emailing

your program to another student, copying-pasting code from the internet, working in a group on a homework assignment, and allowing a tutor, TA, or another individual to write an answer for you. It is also considered academic dishonesty if you click someone else's iClicker with the intent of answering for that student, whether for a quiz, exam, or class participation. Academic dishonesty is unacceptable, and penalties range from a letter grade drop to expulsion from the university; cases are handled via the official student conduct process described at https://dos.uic.edu/conductforstudents.shtml.