Project 1: Switch

DUE: Sunday, Feb 18 at 11:59pm

Basic Procedures

You must:

- Fill out a readme.txt file with your information (goes in your user folder, an example readme.txt file is provided)
- Have a style (indentation, good variable names, etc.)
- Comment your code well in JavaDoc style (no need to overdo it, just do it well)
- Have code that compiles with the command: `javac *.java` in your user directory
- Have code that runs with the command: `java PlaySwitch`

You may:

- Add additional methods and variables, however these methods **must be private**

You may NOT:

- Delete/change any code in your starter package such as changing class names, etc.
- Make your program part of a package
- Add additional public methods or variables
- Use any built-in Java Collections Framework classes anywhere in your program (e.g. **no ArrayList**, **LinkedList**, HashSet, etc.)
- Use any arrays anywhere in your program (except in the Hand class data field)
- Alter any method signatures defined in this document of the template code
- Add any additional libraries/packages which require downloading from the internet

Setup

- Download the project1.zip and unzip it. This will create a folder `section-yourGMUUserName-p1`
- Replace "`section`" with `z001, z002, k003`, and `z004` for the 4 sections respectively.
- Rename the folder replacing `yourGMUUserName` with the first part of your GMU email address/netID. Example: `k003-jkrishn2-p1`
- Complete the `readme.txt` file (an example file is included: `exampleReadmeFile.txt`)

Submission Instructions

- Make a backup copy of your user folder!
- Remove all test files, jar files, class files, etc.
- You should just submit your java files and your readme.txt
- Zip your user folder (not just the files) and name the zip "`k003-yourGMUUserName-p1.zip`" (no other type of archive) where "`yourGMUUserName`" is your GMU email address / netID.
- Submit to blackboard.

Grading Rubric

Due to the complexity of this assignment, an accompanying grading rubric pdf has been included with this assignment. Please refer to this document for a complete explanation of the grading.

Topics Covered

Generics, Array Lists, and Linked Lists

Step 0: Overview

You are going to build a small card game called **`SWITCH`** using expendable array and linked list. You will be given two abstract classes – **`Card`** and **`Board`**, from which you will implement the game specific classes – **`CardSwitch`** and **`BoardSwitch.`** You will also implement an array list-like data structure that keeps track of the cards in a player's hand. This is the **`Hand`** class, used to represent a **`Player`** and a **`Deck`**. The **`Board`** class maintains a linked list-like data structure that keeps track of all players in the game and decides the winner of the game.

Step 1: JavaDocs

You are required to complete the JavaDoc comments for **ALL** classes including the ones you did not implement (except for **`PlaySwitch`**). As part of understanding this project, start with writing JavaDoc comments for the classes that are implemented for you: **`Card,`** **`Deck`**, and **`Board`**. Yes, you will be graded on this, and yes, you need to add comments to the methods and instance variables.

Step 2: Implementing and Understanding the Code

There is a total of **`7`** classes in the project. However, you are only required to implement some of them. See the project starter package for all the files.

The `abstract Card` Class (see Card.java)

[code provided: should not be modified except for adding JavaDoc comments]

A card consists of rank and suit. A regular deck of 52 cards contains 13 different ranks and 4 different suits. Ranks consist of `ACE, TWO, THREE, FOUR, FIVE, SIX, SEVEN, EIGHT, NINE, TEN, JACK, QUEEN,` and `KING`. Suits consist of `HEARTS, CLUBS, DIAMONDS,` and `SPADES`. In our design, every card has a point and a priority which should be implemented in its inherited classes.

The `CardSwitch` Class (see CardSwitch.java)

Write a class specific to the game SWITCH by extending the abstract class Card. Required methods:

`public CardSwitch(Rank rank, Suit suit)`

- Create a card with a given rank and suit

`public int getPoints()`

- Returns an integer according to the point of this card. **`ACE`** = 1 point, **`2`** = 2 points, **`3`** = 3 points, . . . , **`9`** = 9 points, **`(10, JACK, QUEEN, KING)`** = 10 points each.

`public boolean equals(Card anotherCard)`

- Returns true if the card is of the same rank and suit of another Card

`public String toString()`

- Return a string representation of the card in the form of (Rank,Suit) with all capital letters and no space. For example: `(KING,SPADES)` or `(EIGHT,HEARTS)`

The **`Hand<T extends Card>`** Class (see Hand.java)

Hand holds an array of cards that can be expanded. You need to implement the required methods and make sure their big-O value is as required. Required methods:

`public Hand()`

- Create an empty hand

`public int numCards()`

- Returns the number of cards in hand

`public T getCard(int index)`

- Returns the card at the given index. Should throw a **`RuntimeException`** for invalid index

`public void setCard(int index, T c)`

- Place the given card at the given index: this will replace a card that is already at that index. Should throw a **`RuntimeException`** for invalid index

`public void addCard(T c)`

- Append a card to the end of the hand. Remember to expand the array if it gets full

`public int indexOf(T c)`

- Return the index of the given card in the hand. If the card is not present, return -1

`public T removeCard(int index)`

- Remove and return the card at the given index in the hand. Should throw a **`RuntimeException`** for invalid index

`public boolean removeCard(T card)`

- Remove the first occurrence of the given card from the hand and return true if removal was successful

The **`Player<T extends Card>`** Class (see Player.java)

Player is a linked list node class consisting of name, points, hand, and next player. Required methods:

`public Player(String name)`

- Create a player setting the player's name and initializing points, hand, and the next player

`public void setNext(Player<T> p)`

- Set the next player. i.e., connect the linked list node to its next

`public Player<T> getNext()`

- Get the next player. i.e., get the linked list node connected to the current

`public boolean hasNext()`

- Return true if the player has a next player. i.e., if the current linked list node is connected `public int getPoints()`

- Return the player's points. Total points accumulated by a player in `SWITCH` game is the sum of points of all cards the player currently has in the hand.

`public String getName()`

- Return name of the player

`public boolean receiveCard(T c)`

- Add a card to the hand and update points accordingly. Return false if the hand already contains the card, otherwise return true

`public boolean hasCard(T c)`

- Return true if the given card is present in the hand

`public boolean playCard(T card)`

- Remove the given card from the hand and update points accordingly. Return false if the hand does not have the card, otherwise return true

`public T playCard(int index)`

- Remove and return the card at the given index in the hand. Should throw a **`RuntimeException`** for invalid index

The `abstract Board<T extends Card>` Class

[code provided: should not be modified except to add JavaDoc comments]
The board keeps track of the deck, current player, and the number of players. A board should be able to add players and maintains a **circularly linked list** of players. i.e. `Player1` → `Player2` → `Player3` → `Player1` for a 3-player board.

The `BoardSwitch<T extends Card>` Class (see BoardSwitch.java)

Write a class specific to the game SWITCH by extending the abstract class Board. Required methods:

`public BoardSwitch(Deck<T> deck)`

- Construct a board with the given deck. Make necessary initializations for other fields of Board

`public Player<T> getCurrentPlayer()`

- Return current player

`public int getNumPlayers()`

- Return the number of players on the board

`public Deck<T> getDeck()`

- Return board's deck

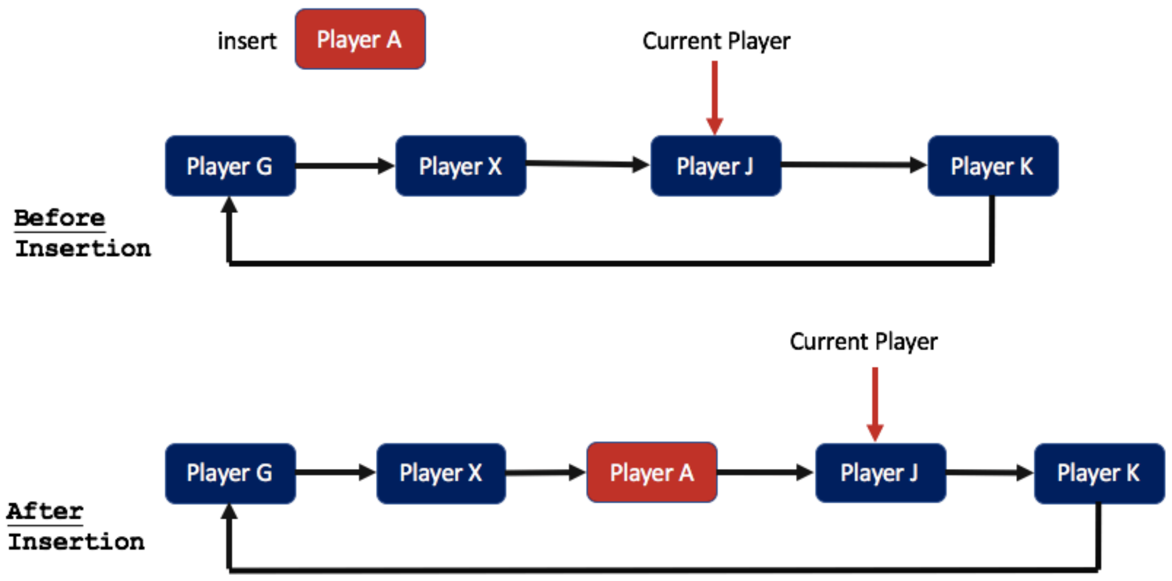
`public boolean changeTurn()`

- Change current player to the next player in the circular linked list. Return true if successful `public Player<T> findWinner()`

- Go through the players on the board and return the player with maximum points

`public boolean addPlayer(Player<T> p)`

- Add the given player to the left of the current player. Don't forget to keep the linked list circular. See example below:



The `Deck<T extends Card>` Class (see Deck.java)

[code provided: should not be modified except for adding JavaDoc comments]

A deck consists of a Hand of cards. The Deck class should be able to add cards, shuffle them, deal the next card, and check the number of cards. You will need to perform big-O analysis to all methods listed here except for the constructor.

`public Deck()`

- Create a deck initializing the set of cards and the card count

`public boolean addCard(T c)`

- Returns false if the deck already has the card. Otherwise, add the card to the deck and update the card count

`public boolean hasCard(T c)`

- Returns true if the deck has the given card

`public void shuffle()`

- Randomly shuffles the deck

`public T dealNextCard()`

- Returns false if the deck has no cards. Otherwise, return the last card in the deck, remove it from the deck and update the card count `public boolean isEmpty()`

- Returns true if the deck is empty. Otherwise returns false

`public int cardCount()`

- Returns the number of cards in the deck

`public String toString()`

- Returns a string consisting of the cards in the deck

The `PlaySwitch` Class (see PlaySwitch.java)
[code provided: should not be modified. No JavaDoc comments required]
A provided implementation of a simple game with cards. The game follows these steps:

1. Creating a full deck of 52 cards
2. Creates players and adds them to board
3. Deal all cards to users one by one (all players may not receive same number of cards)
4. Switch the first n cards between the players
5. Player with the higher total points of all cards in hand is the winner

Step 3: Big-O

Template given to you in the starter package contains instructions on the REQUIRED Big-O runtime for each method. Your methods should **not** have a higher Big-O. For class **`Deck`**, you are required to present Big-O run time for all methods (as comments within the code) in addition to Javadoc commenting. Again yes, you will be graded on this.

Step 4: Testing

Test cases will not be provided for this project. However, feel free to create test cases by yourselves. In addition, the main methods provided along with the template classes contain useful code to test your code. You can use command like "`java CardSwitch`" to run the testing defined in main(). You could also edit main() to perform additional testing. And yes, a part of your grade will be based on automatic grading using test cases that are not provided to you.

EXAMPLE RUNS

SAMPLE RUN#1

```
Enter the number of players: 3
Enter Name of Player 1:
Daenerys
Enter Name of Player 2:
Jon
Enter Name of Player 3:
Cersei
How many cards should be switched?
5
-----------------------------------
-Starting ROUND 1-
switch from Jon:  card (TEN,CLUBS),  switch to Cersei
switch from Cersei:  card (KING,HEARTS),  switch to Daenerys
switch from Daenerys:  card (JACK,CLUBS),  switch to Jon
-Starting ROUND 2-
switch from Jon:  card (FIVE,CLUBS),  switch to Cersei
switch from Cersei:  card (NINE,SPADES),  switch to Daenerys
switch from Daenerys:  card (SIX,SPADES),  switch to Jon
-Starting ROUND 3-
switch from Jon:  card (JACK,DIAMONDS),  switch to Cersei
switch from Cersei:  card (KING,CLUBS),  switch to Daenerys
switch from Daenerys:  card (KING,DIAMONDS),  switch to Jon
-Starting ROUND 4-
switch from Jon:  card (QUEEN,SPADES),  switch to Cersei
switch from Cersei:  card (SEVEN,DIAMONDS),  switch to Daenerys
switch from Daenerys:  card (JACK,SPADES),  switch to Jon
-Starting ROUND 5-
switch from Jon:  card (ACE,DIAMONDS),  switch to Cersei
switch from Cersei:  card (SIX,HEARTS),  switch to Daenerys
switch from Daenerys:  card (ACE,CLUBS),  switch to Jon
-----------------------------------
Winner is: Daenerys with 127 points
```

SAMPLE RUN#2

```
Enter the number of players: 5
Enter Name of Player 1:
Ironman
Enter Name of Player 2:
Hulk
Enter Name of Player 3:
CaptainAmerica
Enter Name of Player 4:
Hawkeye
Enter Name of Player 5:
BlackWidow
How many cards should be switched?
3
-----------------------------------
-Starting ROUND 1-
switch from CaptainAmerica:  card (TEN,DIAMONDS),  switch to Hawkeye
switch from Hawkeye:  card (NINE,HEARTS),  switch to BlackWidow
switch from BlackWidow:  card (JACK,HEARTS),  switch to Ironman
switch from Ironman:  card (QUEEN,SPADES),  switch to Hulk
switch from Hulk:  card (JACK,CLUBS),  switch to CaptainAmerica
-Starting ROUND 2-
switch from CaptainAmerica:  card (KING,SPADES),  switch to Hawkeye
switch from Hawkeye:  card (SEVEN,HEARTS),  switch to BlackWidow
switch from BlackWidow:  card (KING,DIAMONDS),  switch to Ironman
switch from Ironman:  card (SIX,CLUBS),  switch to Hulk
switch from Hulk:  card (NINE,DIAMONDS),  switch to CaptainAmerica
-Starting ROUND 3-
switch from CaptainAmerica:  card (THREE,CLUBS),  switch to Hawkeye
switch from Hawkeye:  card (FIVE,DIAMONDS),  switch to BlackWidow
switch from BlackWidow:  card (TWO,HEARTS),  switch to Ironman
switch from Ironman:  card (EIGHT,SPADES),  switch to Hulk
switch from Hulk:  card (SEVEN,DIAMONDS),  switch to CaptainAmerica
-----------------------------------
Winner is: Hulk with 77 points
```