BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI
Second Semester 2013-14

ONLINE EXAM (OPEN BOOK)
OBJECT ORIENTED PROGRAMMING and
OBJECT ORIENTED PROGRAMMING AND DESIGN

TIME: 90 Min **MAX MARKS:** 60 **DATE:** 13/4/2014

IMPORTANT INSTRUCTIONS

1. Download and extract **Online.zip** file.
2. It consists the following
   A. source code folder in cardGame.zip file
   B. Question paper in SDHC.pdf file.
   C. Sample Input / Output in sample.pdf file.
3. Extract the cardGame.zip file into cardGame folder.
4. Rename the extracted folder as your **student ID number** (for example P2012A7PSXXX).
5. Start eclipse and **create a new project** in the **D drive** of your computers file system. Copy the renamed folder into the **src** folder of the project.
6. You are **not allowed** to do the following things:
   - ✓ Modify the code of any class which is provided to you except those methods for which you are explicitly asked to write the code.
   - ✓ Import any new classes in your solution.
   - ✓ Create new methods.
   - ✓ Declare new instance variables.
   - ✓ Create new classes in your solution.
7. After reading the question paper you have to write the code into only **run ()** function of P**layer** and **GameManager** class.
8. Periodically keep saving your work by pressing **Ctrl+Shift+S** (saves all your modifications). Once you finish writing the code or when the exam time gets over save all your work. Raise hand and call the invigilator.

Consider the SDHC card game. SDHC stands for Spade, Diamond, Heart and Clubs respectively. This game is played between game manager and players. Game manager distributes the card, sets the turn of players, and computes the round winner and game winner. Players throws the card in each round. It has following rules

- There are four players (P1, P2, P3, P4) playing with one deck of cards. (1 deck=52 cards)
- Cards are distributed equally among 4 players and each player gets 13.
- Each Player throws card based on policy **that is described below.**
- Player throws the card in circular fashion. Like first round start by Player P1 then P2, P3 and P4 throws card one after the other. Similarly second round starts with P2 followed by P3, P4 and P1 sequentially. This process is repeated for 13 rounds.
- After each round (when all players throw cards) round winner is computed by game manager based on the rankings of the cards that is mentioned below. The player who throws highest weighted card wins the respective round**.**

- After each round the table is emptied.
- These steps are repeated 13 times till all the cards are exhausted.
- The player who wins the maximum number of rounds will be declared as a game winner.

Ranking of Cards

Weight for number on the card

Number on card	2	3	4	5	6	7	8	9	10	J	Q	K	A
Weight	2	3	4	5	6	7	8	9	10	11	12	13	14

Weight for the category of the card

Category of the card	Spade	Diamond	Heart	Club
Weight	45	30	15	1

Total weight of a card depends on the number and the category. It is computed as

Total weight= weight of the category of the card + weight of the Number on the card.

Rules of the Game

- The player who starts the round, throws the maximum weighted card among available card with him.
- Remaining players check the highest weighted card on the table, compares with his highest weighted card. If his card weight is higher than the highest weighted card on table than throws that card otherwise he throws least weighted card from his cards.
- The player who wins the maximum rounds, is declared as a winner.
- **Assume, there will no tie between players.**

Sample Input / Output is given in sample.pdf.

Cards are shuffled and distributed to the players by the game manager. Currently program is not taking input but it may take.

Four classes are declared. Instance variables and functions of classes are specified below. Each class has **constructor (), accessor () and mutator ()** methods for instance variables. Specific functions of these methods are described below.

1. Card class:

A. Description of instance variables

- String number // It keeps card number like 2, 3……J, Q, K, A.

- **char category** // It keeps card category like s for Spade …

- **int totalWeight**; //It stores the weight of card.

B. Description of functions

+ **void settotalWeight()** /*This method computes and stores the weight of card based on above described policy. */

2. Table class:

Instance of this class is shared among GameManger and Players of SDHC game. **Its methods are not synchronized and modification is not allowed.**

A. Description of instance variables

- **ArrayList<Card> roundCards** /* It is list of cards thrown on table by players. Players add the thrown card into this list. After each round is completed i.e all the four players have thrown card. **GameManager** computes the round winner based on this list after that clears the roundCards .*/

- **int currentTurn** /* This variable is used to communicate between game manger and players. Before each round begins, game manager sets **currentTurn:Table** value to 1. In each round, player checks his **turn:Player** with the **currentTurn:Table**, if it matches, he throws the card and increments the **currentTurn:Table** value so the next player whose turn is equal to new **currentTurn:Table** value, gets the chance.

-**int round** // It keeps the current round value. Its initial value is 1.

-**int maxRounds** // its value is 13.

B. Description of functions

+void setcurrentTurn(int currentTurn) /* This method is used by the game manager and players to set the **currentTurn:Table** . Its usages is described at **currentTurn:Table** instance variable.

3. GameManager Class:

This class extends the **Thread** class. It initializes the players, with name and shared instance of **Table:table**. It also distributes the card to players, starts the players, computes the round winner, winner of game and sets turns of players.

A. Description of instance variables

-**ArrayList<Player> playersList;** // ArrayList of players

-**Table table;** // instance of table

B. Description of functions

GameManager() /*constructor that create an instance of table, list of players, call the initializePlayers(), setPlayersTurn() and starts the itself and players. */

main() // Main function create an instance of **GameManager** class.

run() /* All the activities of game manager during the course of game is performed in this method. Details of each step is mentioned in the code, mark as a comments. */

Player roundWinner() /*This function computes the round winner. It reads the **roundCards:Table** to find the maximum weighted card of round. Then finds the player of that card and returns the round winner player.*/

void setPlayersTurns() /*This function sets the turns of players based on round in circular fashion. */

Player gameWinner() /* This function computes the winner of SDHC game. The player who has maximum number **wonRounds:Player** is declared as a winner. It returns the winner player.

void initializePlayers() /*This function initialized the **playerList:GameManager** with name and shared instable of table. It also initializes the player cards **playerCards:Player**. */

4. Player Class:

This class extends the **Thread** class.

A. Description of instance variables

-**ArrayList<Card> playerCards** //It has 13 cards at start of the SDHC game.

-**Table table** // shareable data structure

-String Name; // name of player

-int turn; // player turn it is decided by game manger. Its value is changed in each round.

-int roundWon; // It keeps the number of rounds won by the player

B. Description of functions

Player(String Name,Table table) // constructor initialized the player instance.

Card selectCardToThrow() /* This function returns the card. In order to find card, it finds the highest weighted card from **roundCards:Table** compare with his **playerCards:Player** highest weighted card. If his card weight is higher than the highest weighted card of **roundCards:Table,** Then throws his highest weighted card otherwise he throws least weighted card from his **playerCards:Player**. */

Run() /* All the activities of players during the course of game are performed in this method. Details of each step is mentioned in the code, mark as a comments. */

You have to write the code for the following:

A. First implement the **run()** method of the **GameManager class** as per the specification and the comments provided in the **GameManager.java** file.
B. Second implement the **run()** method of the **Player class** as per the specification and the comments provided in the **Player.java** file.

_____Best of Luck_____