ASSIGNMENT FOR CORE-JAVA AND HTML/ CSS.

Authors [kedar_tokekar@persistent.com and mukta_peshave@persistent.com]
Date : 2023.04.07.

ASSIGNMENT – 1
Part-1 : CORE JAVA:
An application needs to be created for writing a poker Card game. Following are the requirements for the same.

Implementation guidelines:

1. Implementation needs to be Object oriented. To make it object oriented, you can use these guidelines.
2. Do not read any data from command line unless specifically required (In this assignment its not required). Project will be run from hard coded data in main(…) method. Sample for main(…) is provided at the end.
3. Identify nouns and verbs from statement.
    a. Nouns are potential candidates for becoming a class, attribute, or object name. Use these nouns for naming classes, attributes, or object ref names.
    b. Nouns representing single primitive value are potential candidates for attributes.
    c. Verbs are potential candidates for becoming methods of class.
4. Identify signature (parameters and return type) required for each method.
5. Identify relationship between different classes.
6. If it is A KIND OF relationship
7. If its HAS-A or PART-OF, use containment. Ignore differences between Composition and Aggregation at this stage.
8. Don’t keep any classes in default package. Keep related classes together in appropriately designed packages.
9. Use base package name as com.psl.gems and extend it with readable application package names
10. Write main(…) method in com.psl.gems.client.TestClient class. This main(…) will be starting point for application. Use provided template for main (…).

Application description:

- Playing cards come as Full Pack of 52 cards. (4 Suits X 13 Ranks). Ref the picture. If you are not aware about playing cards best source is Wikipedia.

- 4 Suits are CLUB, DIAMOND, HEART, SPADE (from smallest to highest in value)
- Ranks are called as TWO, THREE, …, TEN, JACK, QUEEN, KING, ACE (from smallest to highest in value)
- Any group of cards (containing 1..52 cards) is called Pack of Cards.
- Full pack is a Pack of Cards initialized with all 52 cards, but in the game as one or more cads are drawn from Full Pack, the count may go down till zero.
- Pack of Cards will maintain the sequence unless changed by any act like shuffle, reset etc.
- A Player can do following actions with Pack of Cards
    - can shuffle
    - can get top card
    - can add a card (duplicates are not allowed)
    - can get random card
    - can query size of pack.
    - Can reset sequence in ascending/ descending order.
- Game begins by initializing the Full pack. [main(…) will be a starting point]
- Player (with name) registers with the game. [In short game will keep collection of Players registered]
- Once two players register for the Game (as a poker game), game will automatically start. [hint: will call its play() method]
- When game is played (in a game.play()), three cards will be given to each Player as one card at a time in alternate manner. Plyers add this each card in their Cards in Hand.
- Once three cards are distributed, Each Player has Cards in Hand (as a pack of 3 cards). Based on following rule, game will declare winner player. (This being simple luck game, player does not take any action)
    - Rule for winner.
        - Each card has a value computed using following algorithm.
            - Each suit has suit-weight associated as {SPADE(3), HEART(2), DIAMOND(1), CLUB(0)}
            - Each rank has rank-weight associated as {TWO(0), THREE(1),.., JACK(9), QUEEN(10), KING(11), ACE(12)}
            - Value of the card is computed as Rank-weight * 4 + Suit-weight. (e.g. CLUB-JACK will have value 9 * 4 + 0 = 36, SPADE-SIX will have weight 4*4 + 3 = 19)

        - Highest card for player is card with highest value in "Cards in Hand" for player.
        - Each player's highest card is Card with highest value.
        - Game will compare highest cards for each player, and player with higher "Highest Card" wins the game.
- Example
    - [Player 1 (
        - SPADE-THREE (1 * 4 + 3 = 7),
        - DIAMOND-FIVE (3 * 4 + 1 = 13),
        - CLUB-QUEEN(10*4+0 = 40)

      ]
    - Highest Card is CLUB-QUEEN
    - [Player 2 (
        - SPADE-JACK (9 * 4 + 3 = 39),
        - HEART-QUEEN (10 * 4 + 2 = 42),
        - DIAMOND-KING (11*4+1 = 45)

      ]
    - Highest Card is DIAMOND-KING
- Winner is player 2 (as KING is higher than QUEEN)
- Also programmatically write following output in to *cardgame.html* in following format by replacing actual data for name and cards in hand,
- This *cardgame.html* will be used for next part of assignment.

```
<!DOCTYPE html>
<html lang="en">
<head>
   <title>Document</title>
</head>
<body>
    <section>
        <div id="Suresh">
            <div data-card="SPADE-THREE" />
            <div data-card="DIAMOND-FIVE" />
            <div data-card="CLUB-QUEEN" />
        </div>
        <div id="Ramesh">
            <div data-card="SPADE-JACK" />
            <div data-card="HEART-QUEEN" />
            <div data-card="DIAMOND-KING" />
        </div>
   </section>
</body>
```

- Main(…) method skeleton

```
public static void main(String[] args) {
```

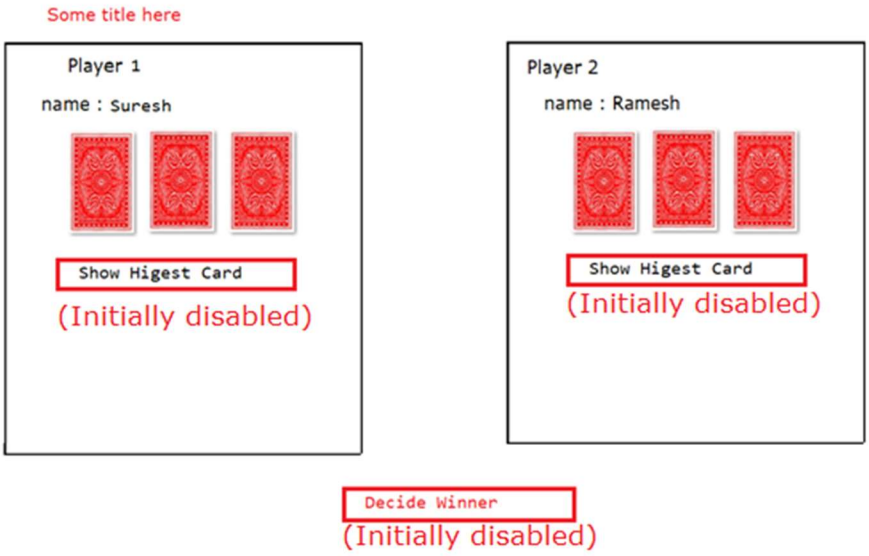
```
        //CREATE PLAYER 1 AND PLAYER 2 WITH SOME NAMES (say Suresh and Ramesh)
        //CREATE AND INITIALIZE CARD-GAME (shuffle the Full Card Pack)
        //LET BOTH THE PLAYERS REGISTER TO THE GAME
        //        (something like game.register(player1))
        // IF GAME HAS 2 PLAYERS, PLAY THE GAME.
        // INSIDE game.play() ALLOT 3 CARDS TO EACH PLAYER
        // as per rules decide highest card for each player
        // as per rule decide winner and print it.
        // WRITE THE PLAYER DATA AND PLAYERS' CARDS IN PROVIDED HTML TEMPLATE IN
                TO cardgame.html file.
}
```

Part-2 : JAVASCRIPT + HTML & CSS:

As mentioned in above java part of problem statement, it will generate raw html *cardgame.html*
By using this basic html implement further UI features

Functionality to be implemented at client side:

1. **Display cards, display highest valued card and decide winner**
   - First using HTML5 and CSS3, design html page as displayed below in image.
   - You should add more styling parameters (e.g., Background-color, borders, margin, padding images etc.) to make it more attractive
   - Web page should be responsive. It should render correctly on all devices.
   - On initial page load, all cards should be displayed facing backside (you can use one common image as backside of each card) for both players as displayed in below image. (Note: single page shows data for both players side-by-side)



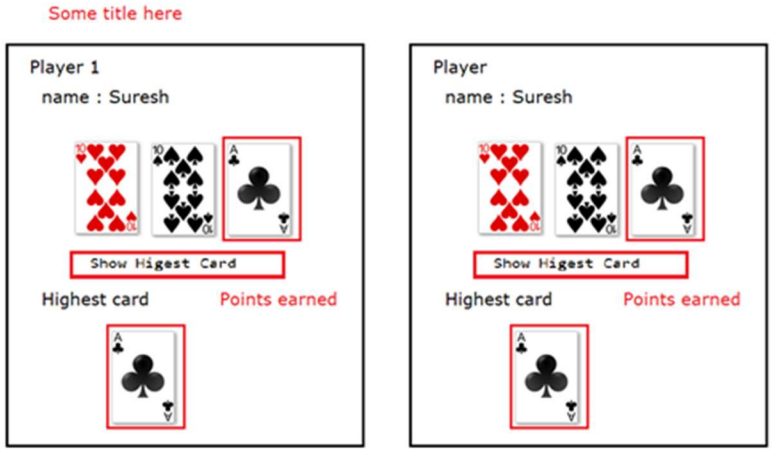
   - On clicking any card, it should flip (i.e., other side of a card should be displayed, use should not be able to click and flip it back)
   - Keep card images folder locally. (U can decide from where to get Card images)
     - Hint: U can organize all 52 images by following below convention:
       - \images\spade\ace.jpg

        - \images\club\ten.jpg
        - \images\diamond\jack.jpg
- When all cards are open for any player, "Show Highest Card" button should be enabled.
- On click of button "Show Highest Card", show highest card and its value for that player. [Note: calculate value of each card for that player and highest card will be card with highest Value].
    - Compute the value of any card as follows:
        - Suit-weights: CLUB(0), DIAMOND(1), HEART(2), SPADE(3)
        - Rank-weights: TWO(0), THREE(1)...QUEEN(10), KING(11),ACE(12)
        - Value of card = rank-weight * 4 + suit-weight

    e.g.: if we are getting a card in HTML as below:

    value= 9 * 4 + 3 = 39

Below image is for player 1, similarly there will image for other player side-by-side.



- **Declare winner**
    - After highest card for each players is displayed, "decide winner" button should be enabled.
    - On click this button display name of winner along with its score as value of g=highest card on same page(You can decide where to display this data)

***