# JAVA GAME PROJECT

# TECHNICAL EXPLANATION

**PREPARED BY:**

**ABBAS SHAZALI IBRAHIM**

## INTRODUCTION

The major goal of this project is to realize a simple Java programming Game of 52 cards to:

1. Construct and then display a card deck in the standard console.

2. And the Game is to be played between 4 players, under certain conditions:

   - The Players are to dealt a hand of cards at the beginning.

   - The players are to select card at random each time they are playing a new round.

   - The player with the highest card will be the winner of that round and he will takes all the cards of that particular round.

   - The first player to win all the cards is deemed the winner of the game.

   - The results are displayed at the end showing the scores and the players' names.

3. An XML file is used to store all the match records, including the Player name and his records of winning and losing.

## SPECIFICATIONS:

As the requirement was that:

- The program should manage a data structure of 52 cards, each card having a value and a color.
- Initialize this data structure correctly (13 values and 4 colors), then display the list on the screen, enforcing this order:
  - Club
  - Diamond
  - Heart
  - Spade

- Then Add the Player and the game notions, with method to start a new game, composed of a number of rounds.

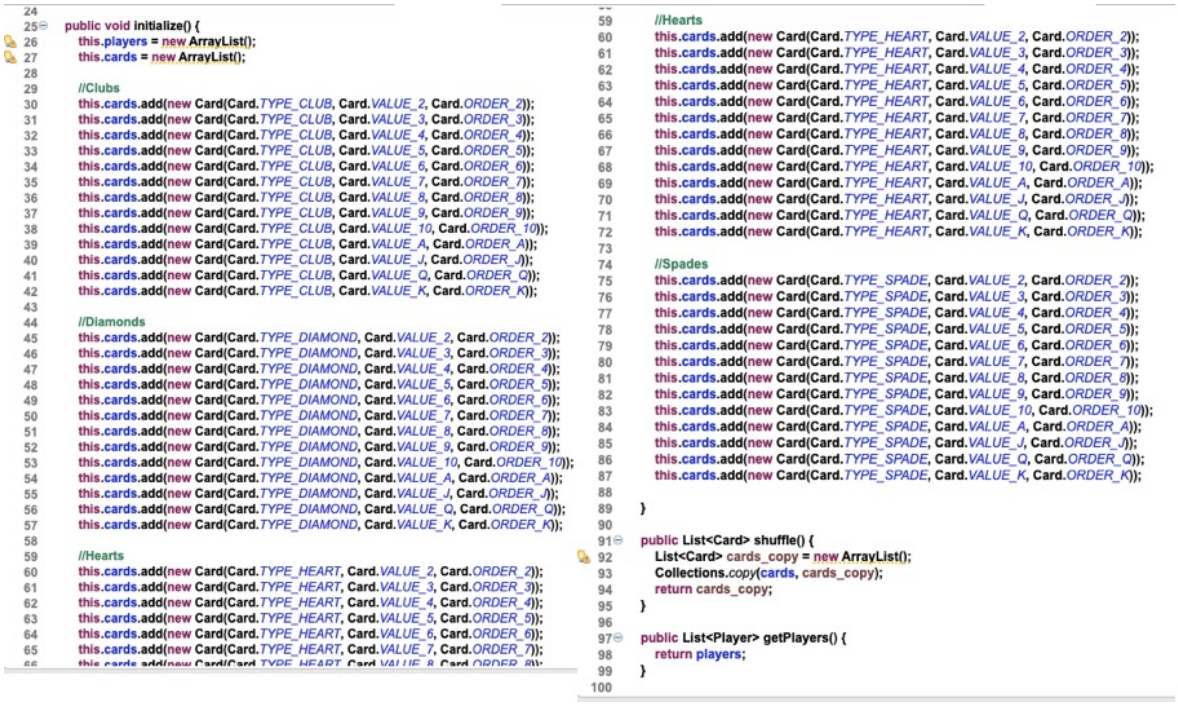
- Display the result of the game on the screen, with how many rounds were played before the winner was found, and who was the winner.

- Save the match results in a database or in an XML file.

**Steps to use:**

1. The application ask “How many players are going to play”? The user must give a numeric input for the number of players and press enter.
2. The application will then ask “How Many rounds do you need to play?” The user as well must give another numeric input specifying the number of rounds.
3. The Application will then ask you to input the names of the players one after the other.
4. Once all these are collected, the game Starts.
5. The user plays the game by inputting his cards choices as the cards are serialized.
6. Once all the number of rounds are been played, the application will display the final scores to the user on the console the players name as well as their scores in a tabular form and stored the Match result in an XML file.

**Screenshots of the realization**



```
public void initialize() {
    this.players = new ArrayList();
    this.cards = new ArrayList();

    //Clubs
    this.cards.add(new Card(Card.TYPE_CLUB, Card.VALUE_2, Card.ORDER_2));
    this.cards.add(new Card(Card.TYPE_CLUB, Card.VALUE_3, Card.ORDER_3));
    this.cards.add(new Card(Card.TYPE_CLUB, Card.VALUE_4, Card.ORDER_4));
    this.cards.add(new Card(Card.TYPE_CLUB, Card.VALUE_5, Card.ORDER_5));
    this.cards.add(new Card(Card.TYPE_CLUB, Card.VALUE_6, Card.ORDER_6));
    this.cards.add(new Card(Card.TYPE_CLUB, Card.VALUE_7, Card.ORDER_7));
    this.cards.add(new Card(Card.TYPE_CLUB, Card.VALUE_8, Card.ORDER_8));
    this.cards.add(new Card(Card.TYPE_CLUB, Card.VALUE_9, Card.ORDER_9));
    this.cards.add(new Card(Card.TYPE_CLUB, Card.VALUE_10, Card.ORDER_10));
    this.cards.add(new Card(Card.TYPE_CLUB, Card.VALUE_A, Card.ORDER_A));
    this.cards.add(new Card(Card.TYPE_CLUB, Card.VALUE_J, Card.ORDER_J));
    this.cards.add(new Card(Card.TYPE_CLUB, Card.VALUE_Q, Card.ORDER_Q));
    this.cards.add(new Card(Card.TYPE_CLUB, Card.VALUE_K, Card.ORDER_K));

    //Diamonds
    this.cards.add(new Card(Card.TYPE_DIAMOND, Card.VALUE_2, Card.ORDER_2));
    this.cards.add(new Card(Card.TYPE_DIAMOND, Card.VALUE_3, Card.ORDER_3));
    this.cards.add(new Card(Card.TYPE_DIAMOND, Card.VALUE_4, Card.ORDER_4));
    this.cards.add(new Card(Card.TYPE_DIAMOND, Card.VALUE_5, Card.ORDER_5));
    this.cards.add(new Card(Card.TYPE_DIAMOND, Card.VALUE_6, Card.ORDER_6));
    this.cards.add(new Card(Card.TYPE_DIAMOND, Card.VALUE_7, Card.ORDER_7));
    this.cards.add(new Card(Card.TYPE_DIAMOND, Card.VALUE_8, Card.ORDER_8));
    this.cards.add(new Card(Card.TYPE_DIAMOND, Card.VALUE_9, Card.ORDER_9));
    this.cards.add(new Card(Card.TYPE_DIAMOND, Card.VALUE_10, Card.ORDER_10));
    this.cards.add(new Card(Card.TYPE_DIAMOND, Card.VALUE_A, Card.ORDER_A));
    this.cards.add(new Card(Card.TYPE_DIAMOND, Card.VALUE_J, Card.ORDER_J));
    this.cards.add(new Card(Card.TYPE_DIAMOND, Card.VALUE_Q, Card.ORDER_Q));
    this.cards.add(new Card(Card.TYPE_DIAMOND, Card.VALUE_K, Card.ORDER_K));

    //Hearts
    this.cards.add(new Card(Card.TYPE_HEART, Card.VALUE_2, Card.ORDER_2));
    this.cards.add(new Card(Card.TYPE_HEART, Card.VALUE_3, Card.ORDER_3));
    this.cards.add(new Card(Card.TYPE_HEART, Card.VALUE_4, Card.ORDER_4));
    this.cards.add(new Card(Card.TYPE_HEART, Card.VALUE_5, Card.ORDER_5));
    this.cards.add(new Card(Card.TYPE_HEART, Card.VALUE_6, Card.ORDER_6));
    this.cards.add(new Card(Card.TYPE_HEART, Card.VALUE_7, Card.ORDER_7));
```

```
    //Hearts
    this.cards.add(new Card(Card.TYPE_HEART, Card.VALUE_2, Card.ORDER_2));
    this.cards.add(new Card(Card.TYPE_HEART, Card.VALUE_3, Card.ORDER_3));
    this.cards.add(new Card(Card.TYPE_HEART, Card.VALUE_4, Card.ORDER_4));
    this.cards.add(new Card(Card.TYPE_HEART, Card.VALUE_5, Card.ORDER_5));
    this.cards.add(new Card(Card.TYPE_HEART, Card.VALUE_6, Card.ORDER_6));
    this.cards.add(new Card(Card.TYPE_HEART, Card.VALUE_7, Card.ORDER_7));
    this.cards.add(new Card(Card.TYPE_HEART, Card.VALUE_8, Card.ORDER_8));
    this.cards.add(new Card(Card.TYPE_HEART, Card.VALUE_9, Card.ORDER_9));
    this.cards.add(new Card(Card.TYPE_HEART, Card.VALUE_10, Card.ORDER_10));
    this.cards.add(new Card(Card.TYPE_HEART, Card.VALUE_A, Card.ORDER_A));
    this.cards.add(new Card(Card.TYPE_HEART, Card.VALUE_J, Card.ORDER_J));
    this.cards.add(new Card(Card.TYPE_HEART, Card.VALUE_Q, Card.ORDER_Q));
    this.cards.add(new Card(Card.TYPE_HEART, Card.VALUE_K, Card.ORDER_K));

    //Spades
    this.cards.add(new Card(Card.TYPE_SPADE, Card.VALUE_2, Card.ORDER_2));
    this.cards.add(new Card(Card.TYPE_SPADE, Card.VALUE_3, Card.ORDER_3));
    this.cards.add(new Card(Card.TYPE_SPADE, Card.VALUE_4, Card.ORDER_4));
    this.cards.add(new Card(Card.TYPE_SPADE, Card.VALUE_5, Card.ORDER_5));
    this.cards.add(new Card(Card.TYPE_SPADE, Card.VALUE_6, Card.ORDER_6));
    this.cards.add(new Card(Card.TYPE_SPADE, Card.VALUE_7, Card.ORDER_7));
    this.cards.add(new Card(Card.TYPE_SPADE, Card.VALUE_8, Card.ORDER_8));
    this.cards.add(new Card(Card.TYPE_SPADE, Card.VALUE_9, Card.ORDER_9));
    this.cards.add(new Card(Card.TYPE_SPADE, Card.VALUE_10, Card.ORDER_10));
    this.cards.add(new Card(Card.TYPE_SPADE, Card.VALUE_A, Card.ORDER_A));
    this.cards.add(new Card(Card.TYPE_SPADE, Card.VALUE_J, Card.ORDER_J));
    this.cards.add(new Card(Card.TYPE_SPADE, Card.VALUE_Q, Card.ORDER_Q));
    this.cards.add(new Card(Card.TYPE_SPADE, Card.VALUE_K, Card.ORDER_K));

}

public List<Card> shuffle() {
    List<Card> cards_copy = new ArrayList();
    Collections.copy(cards, cards_copy);
    return cards_copy;
}

public List<Player> getPlayers() {
    return players;
}
```

*Fig 1: Cards Initialization in the program*

PlayGame [Java Application] /Library/J

| Order | Type | Value |
|---|---|---|
| 2 | club | 2 |
| 3 | club | 3 |
| 4 | club | 4 |
| 5 | club | 5 |
| 6 | club | 6 |
| 7 | club | 7 |
| 8 | club | 8 |
| 9 | club | 9 |
| 10 | club | 10 |
| 15 | club | A |
| 11 | club | J |
| 13 | club | Q |
| 14 | club | K |
| 2 | diamond | 2 |
| 3 | diamond | 3 |
| 4 | diamond | 4 |
| 5 | diamond | 5 |
| 6 | diamond | 6 |
| 7 | diamond | 7 |
| 8 | diamond | 8 |
| 9 | diamond | 9 |

PlayGame [Java Application] /Library/J

| | | |
|---|---|---|
| 8 | diamond | 8 |
| 9 | diamond | 9 |
| 10 | diamond | 10 |
| 15 | diamond | A |
| 11 | diamond | J |
| 13 | diamond | Q |
| 14 | diamond | K |
| 2 | heart | 2 |
| 3 | heart | 3 |
| 4 | heart | 4 |
| 5 | heart | 5 |
| 6 | heart | 6 |
| 7 | heart | 7 |
| 8 | heart | 8 |
| 9 | heart | 9 |
| 10 | heart | 10 |
| 15 | heart | A |
| 11 | heart | J |
| 13 | heart | Q |
| 14 | heart | K |
| 2 | spade | 2 |
| 3 | spade | 3 |

PlayGame [Java Application] /Library/Java/JavaVirt

| | | |
|---|---|---|
| 8 | heart | 8 |
| 9 | heart | 9 |
| 10 | heart | 10 |
| 15 | heart | A |
| 11 | heart | J |
| 13 | heart | Q |
| 14 | heart | K |
| 2 | spade | 2 |
| 3 | spade | 3 |
| 4 | spade | 4 |
| 5 | spade | 5 |
| 6 | spade | 6 |
| 7 | spade | 7 |
| 8 | spade | 8 |
| 9 | spade | 9 |
| 10 | spade | 10 |
| 15 | spade | A |
| 11 | spade | J |
| 13 | spade | Q |
| 14 | spade | K |

How many players are going to play?

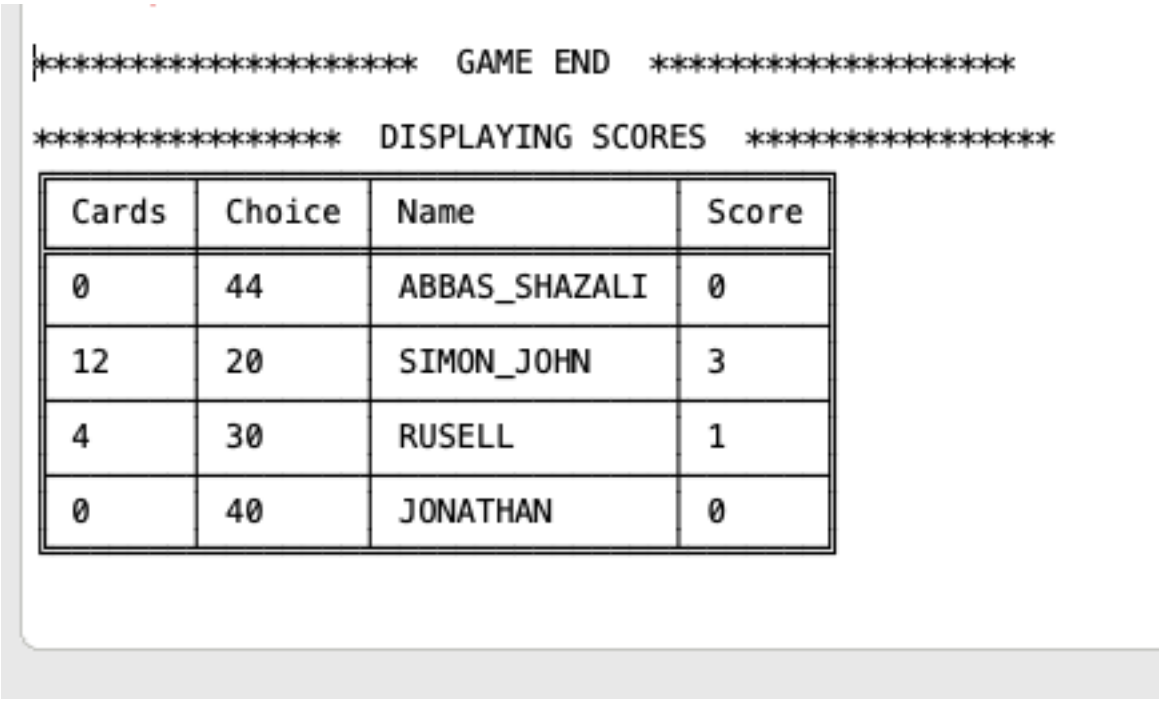
*Fig 2: Cards Display at starting of the game*

```
How many players are going to play?
4
How many rounds?
4
Please enter the player N°1
ABBAS_SHAZALI
Please enter the player N°2
SIMON_JOHN
Please enter the player N°3
RUSELL
Please enter the player N°4
JONATHAN


********** ROUND 0 START **********

-ABBAS_SHAZALI:
32
-SIMON_JOHN: card heart-8


23
card diamond-J
-RUSELL:

15
card diamond-4
-JONATHAN:
30
card heart-6
********** ROUND 1 START **********

-ABBAS_SHAZALI:
```

Console  Git Staging

```
<terminated> PlayGame [Java Application] /Library/Java/JavaVirtualMachines/jdk
********** ROUND 1 START **********

-ABBAS_SHAZALI:
50
card spade-Q
-SIMON_JOHN:
25
card diamond-K
-RUSELL:
21
card diamond-10
-JONATHAN:
37
card heart-Q********** ROUND 2 START **********

-ABBAS_SHAZALI:

40
card spade-3
-SIMON_JOHN:
31
card heart-7
-RUSELL:
9
card club-A-JONATHAN:

18
card diamond-7
********** ROUND 3 START **********

-ABBAS_SHAZALI:
44
card spade-7
-SIMON_JOHN:
20
card diamond-9
-RUSELL:
30
card heart-6
-JONATHAN:
40
card spade-3

**********  GAME END  **********
```

*Fig 3: Declaration of Players and Number of rounds to play*



********** GAME END **********

********** DISPLAYING SCORES **********

| Cards | Choice | Name | Score |
|---|---|---|---|
| 0 | 44 | ABBAS_SHAZALI | 0 |
| 12 | 20 | SIMON_JOHN | 3 |
| 4 | 30 | RUSELL | 1 |
| 0 | 40 | JONATHAN | 0 |

*Fig 4: Displaying the scores for each Player at the end of the game*

```
<game rounds="4" id="6">
  <player>
    <name>ABBAS_SHAZALI</name>
    <score>0</score>
  </player>
  <player>
    <name>SIMON_JOHN</name>
    <score>3</score>
  </player>
  <player>
    <name>RUSELL</name>
    <score>1</score>
  </player>
  <player>
    <name>JONATHAN</name>
    <score>0</score>
  </player>
 </game>
</root>
```

Design Source

*Fig 5: Storing final Scores and the Players name in an XML file.*

After completing the Game, i.e., after playing the final round, the players can see their final score on the console. Meanwhile the scores are saved together with the players name in an XML file.

**UML CLASS DIAGRAM**

In the analysis phase, this diagram represents the entities (information) handled by the users, in the design phase, it represents the object structure of an object oriented development. The UML diagram is very important because it clearly describe the structure of a particular system by modelling its classes, attributes, operations, and the relationships between its objects.

This Game Class Diagram consist of 4 different classes with each having it's attributes.

1. Card Class: this is a class that contains the card how the cards are, with all its card types, value of each card, the order in which they are arranged, as well as the value of each card.
2. Game Class: this is the class that contains the details of the game, like how the game is going to be initialize, and so on
3. Player Class: this class contains the attributes like score, card, choice of the card by the player, the name of the player etc.
4. PlayGame Class: This is the class that contains the attributes of the playing game, which consist of the start of the game, result displaying, round playing, etc…

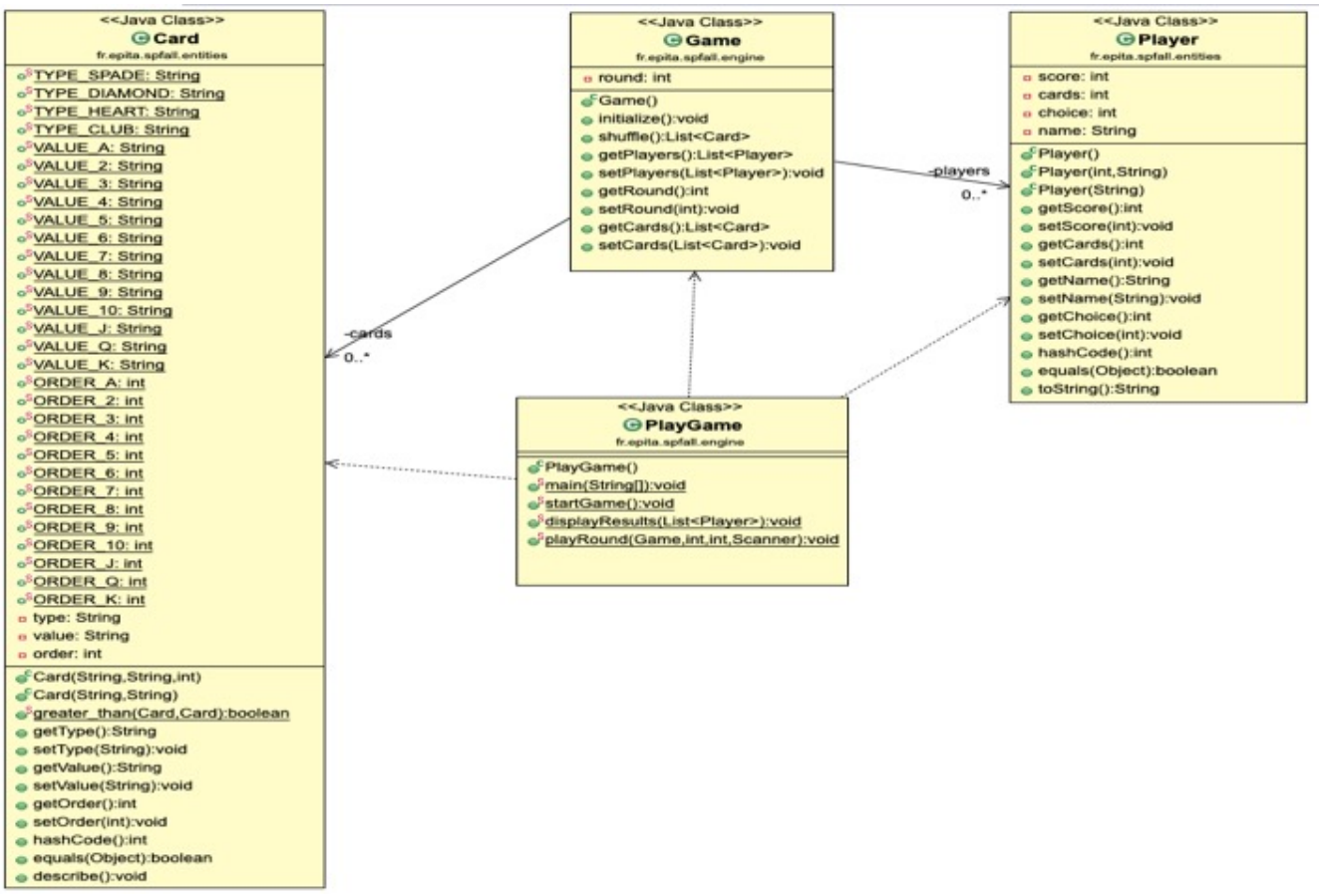



*Fig 6: UML class diagram for the realization.*