# Index

## Java

- Java an object-oriented, cross platform, multi-purpose programming language which was designed by Sun Microsystems in the early 1990s to solve the problem of connecting many household machines together.

- Java can be used to create complete applications that may run on a single computer or be distributed among servers and clients in a network.

- It can also be used to build a small application module or applet for use as part of a webpage.

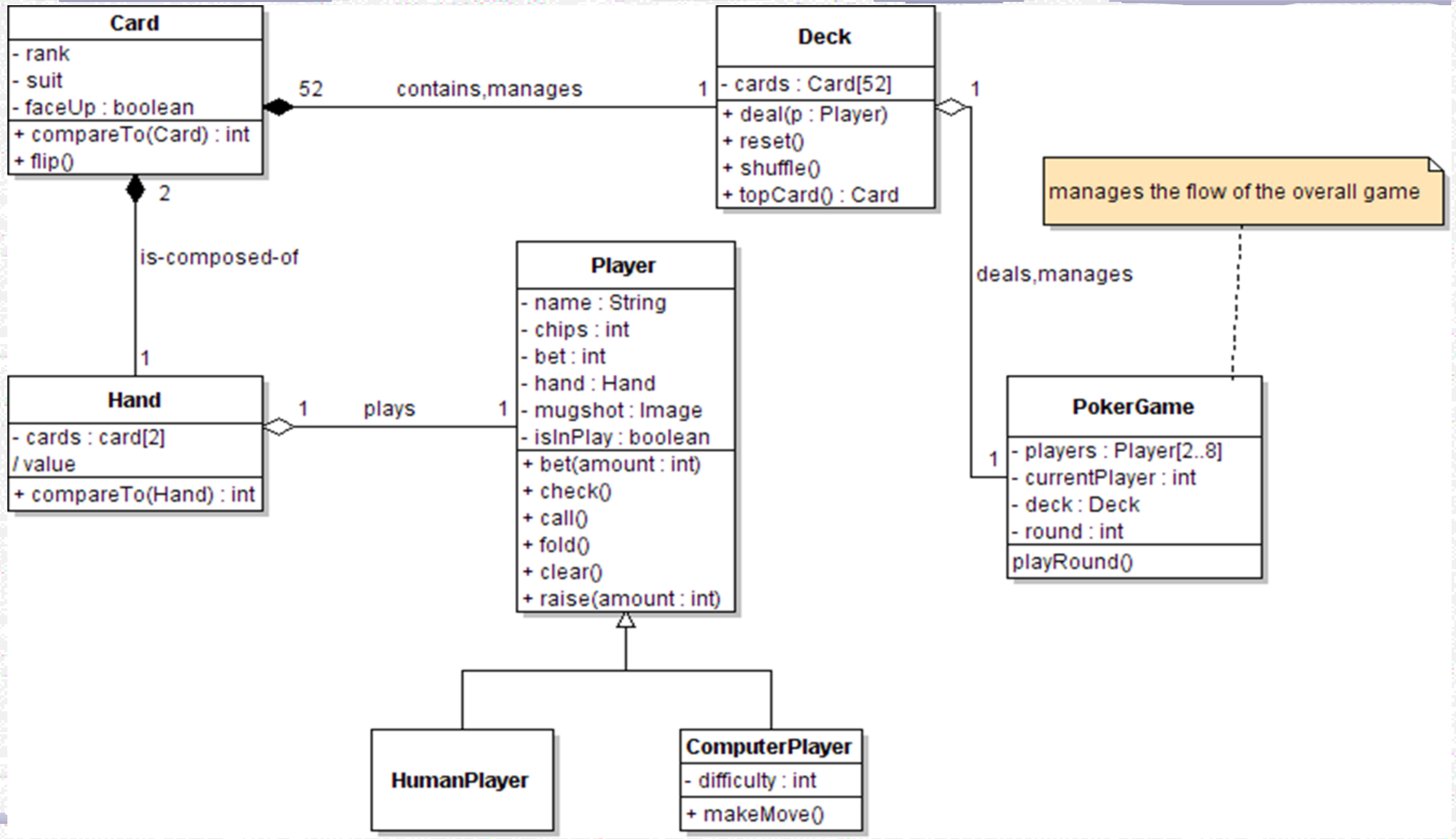
- **Advantages**:
  **1.Simple**: Java was designed to be easy to use, write, compile, debug, and learn than other programming languages.
  **2.Object-Oriented**: Allows you to create modular programs and reusable code
  **3.Platform-Independent**: Ability to move easily from one computer system to another.

## Program Struture

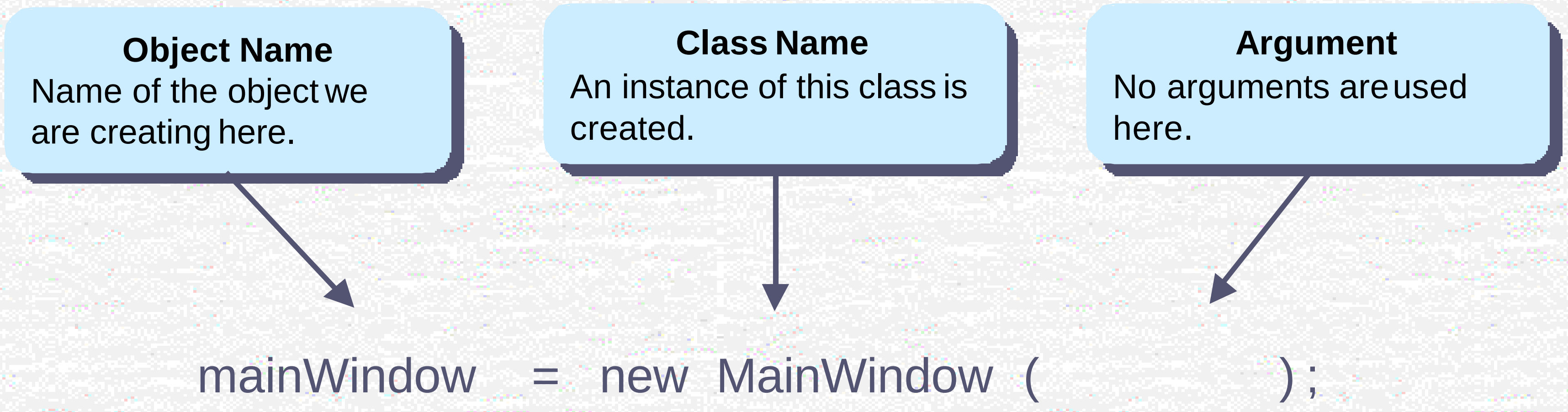
A Java program is composed of:

- Comments,
- Package Statement,
- Main Method Class,
- Import statements, and
- Class declarations.

# Object Diagram & Class Diagram



## Object Creation



**Object Name**
Name of the object we are creating here.

**Class Name**
An instance of this class is created.

**Argument**
No arguments are used here.

```
mainWindow   =  new  MainWindow  (               ) ;
```

# Three types of comments

```
/*
     This is a comment with
     three lines of
     text.
*/
```

**Multiline Comment**

```
// This is a comment
// This is another comment
// This is a third comment
```

**Single line Comments**

```
/**
 * This class provides basic clock functions. In addition
 * to reading the current time and today’s date, you can
 * use this class for stopwatch functions.
 */
```

**javadoc Comments**

## Built-In Types Of Variables

| Type | Description |
|---|---|
| byte | 8 bit signed integer |
| short | 16 but signed integer |
| int | 32 bit signed integer |
| long | 64 bit signed integer |
| float | 32 bit signed real number |
| double | 64 bit signed real number |
| char | 16 bit Unicode character (ASCII and beyond) |
| boolean | 1 bit true or false value |
| String | A sequence of characters between double quotes ("") |

## Keywords in Java

| | | | | | | |
|---|---|---|---|---|---|---|
| abstract | boolean | break | byte | case | catch | char |
| class | const | continue | default | do | double | else |
| extends | final | finally | float | for | goto | if |
| implements | import | instanceof | int | interface | long | native |
| new | package | private | protected | public | return | short |
| static | super | switch | synchronized | this | throw | throws |
| transient | try | void | volatile | while | | |

## Project - Introduction

- This project is proposed to students in order to evaluate their skills for the fundamental period of Java Programming and UML.

- Here is a simple game played with a pack of cards, usually by 2 or more players. Initially 'n' number of cards are distributed to all players. In each round, a player is given a chance to select a single card from his own set of cards. The player with maximum card number wins the round and gets a point. At last, player with most number of points wins the game.

- This goal is achieved through the realization of a console application (+ GUI as possible bonus), which aims at managing card game preparation and execution.

## Constructor and instance methods in class Deck:

```
public Deck()
      // Constructor.  Create an unshuffled deck of cards.

public void shuffle()
     // Put all the used cards back into the deck,
     // and shuffle it into a random order.

public int cardsLeft()
     // As cards are dealt from the deck, the number of
     // cards left decreases.  This function returns the
     // number of cards that are still left in the deck.

public Card dealCard()
     // Deals one card from the deck and returns it.
     // Throws an exception if no more cards are left.
```

## Constructor and instance methods in class Hand:

```
public Hand() {
        // Create a Hand object that is initially empty.

   public void clear() {
        // Discard all cards from the hand, making the hand empty.

   public void addCard(Card c) {
        // Add the card c to the hand.  c should be non-null.
        // If c is null, a NullPointerException is thrown.

   public void removeCard(Card c) {
        // If the specified card is in the hand, it is removed.

   public void removeCard(int position) {
        // Remove the card in the specified position from the
        // hand.  Cards are numbered counting from zero.  If
        // the specified position does not exist, then an
        // exception is thrown.
```

## Project Code

```
package in.madeena.game;

public inteface Game
{
	void playGame(int numberOfPlayers);

	void displayWinners();
}
package in.madeena.game;

import java.util.ArrayList;
import java.util.Collections;
import java.util.List;
```

```
public enum CARDTYPE
        {
                CLUB, DIAMOND, HEARTS, SPADE;
        }

        private CARDNUMBER      cdNumber;
        private CARDTYPE        cdType;

        public CARDNUMBER getCdNumber()
        {
                return cdNumber;
        }

        public CARDTYPE getCdType()
        {
                return cdType;
        }
```

```
public int compareTo(CARD o)
        {
                if (this.getCdNumber() == o.getCdNumber())
                {
                        return 0;
                }
                else if (this.getCdNumber().getOrd() > o.getCdNumber().getOrd())
                {
                        return 1;
                }
                else
                        return -1;}
        public String toString()
        {
                return "CARD [cdNumber=" + cdNumber + ", cdType=" + cdType + "]";
```

```
this.numberOfPlayers = numberOfPlayers;
createMultipleUser(numberOfPlayers);
int i = 0;
System.out.println("Game Started.....  ");
List<CARD> selCards = new ArrayList<CARD>();
CARD maxCard = null;
Player maxPlayer = new Player(0);
distributeCardsForPlayers(players);
for (int j = 0; j < numberOfCardsPerPlayer; j++)
{
        int s = 0;
        do
        {
                Player player = getNextPlayer();
                System.out.println("1. display Cards available  \n2. Stop Game");
```

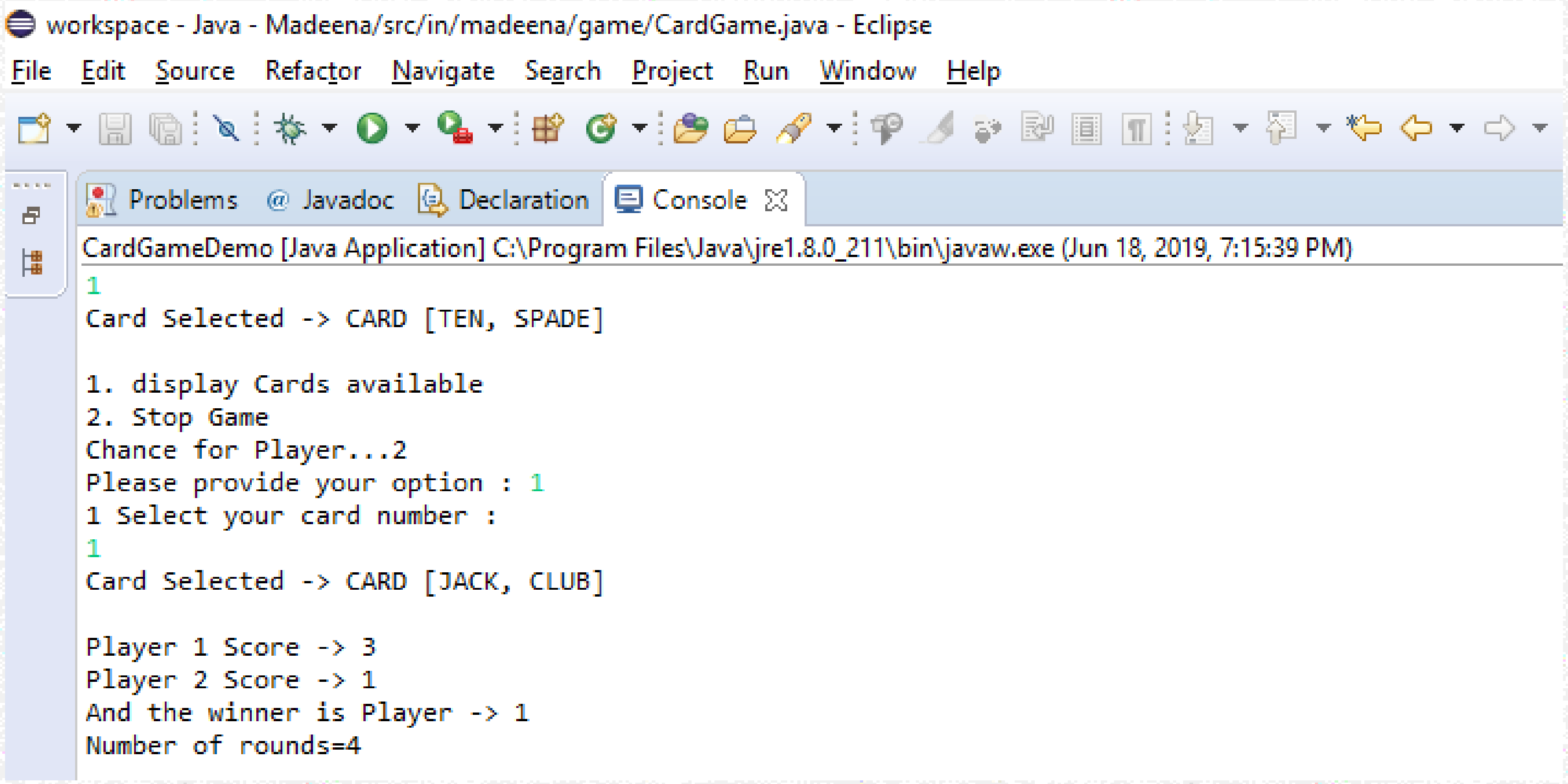
```
switch (i)
 {
        case 1:
                displayCardsForPlayer(player);
                System.out.println("Select your card number :");

                in = new Scanner(System.in);
                int m = in.nextInt();
                CARD c = cardsPlayerMap.get(player).get(m - 1);
                System.out.println("Card Selected -> " + c.toString());
                cardsPlayerMap.get(player).remove(m - 1);
                if (maxCard == null)
                {
                        maxCard = c;
                        maxPlayer = player;
```

```
int i = 1;
   while (i != 0)
                { Scanner in = new Scanner(System.in);
                        i = in.nextInt();

                        switch (i)
                        {case 1:
                                        System.out.println("Provide the Number of Players you want to paly with (
should be greater than 1 and less than 4) : ");
                                        in = new Scanner(System.in);
                                        i = in.nextInt();
                                        sl.playGame(i);
                                        sl.displayWinners();
                                        break;
                        case 2:
                                        System.exit(0);
```

## Project - Outcome


workspace - Java - Madeena/src/in/madeena/game/CardGame.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Problems
Javadoc
Declaration
Console

```
CardGameDemo [Java Application] C:\Program Files\Java\jre1.8.0_211\bin\javaw.exe (Jun 18, 2019, 7:15:39 PM)
1
Card Selected -> CARD [TEN, SPADE]

1. display Cards available
2. Stop Game
Chance for Player...2
Please provide your option : 1
1 Select your card number :
1
Card Selected -> CARD [JACK, CLUB]

Player 1 Score -> 3
Player 2 Score -> 1
And the winner is Player -> 1
Number of rounds=4
```

The End