Product source code

Game.java

```
package in.techdive.game;
public interface Game
{
	void playGame(int numberOfPlayers);
	void displayWinners();
}
```

CARD.java

```
package in.techdive.game;
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;
public class CARD implements Comparable<CARD>
{
	private CARD()
	{
	}
	public enum CARDNUMBER
	{
		TWO(2), THREE(3), FOUR(4), FIVE(5), SIX(6), SEVEN(7), EIGHT(8),
NINE(9), TEN(10), JACK(11), QUEEN(12), KING(13), ACE(
				14);
		private int	ord;
		private CARDNUMBER(int i)
		{
			this.ord = i;
		}
				public int getOrd()
		{
			return ord;
		}
	}
	public enum CARDTYPE
	{
		CLUB, DIAMOND, HEARTS, SPADE;
	}
	private CARDNUMBER	cdNumber;
	private CARDTYPE	cdType;
	public CARDNUMBER getCdNumber()
	{
		return cdNumber;
	}
	public CARDTYPE getCdType()
	{
		return cdType;
	}
	public static List<CARD> getPackOfCards()
	{
		List<CARD> crdLst = new ArrayList<CARD>();
		for (CARDTYPE types : CARDTYPE.values())
		{
			for (CARDNUMBER cNums : CARDNUMBER.values())
			{
				CARD cd = new CARD();
				cd.cdNumber = cNums;
				cd.cdType = types;
				crdLst.add(cd);
			}
		}
		return crdLst;
	}
	public static void shuffleCards(List<CARD> cards)
	{
		Collections.shuffle(cards);
	}
	@Override
	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;
	}
	@Override
	public String toString()
	{
		return "CARD [cdNumber=" + cdNumber + ", cdType=" + cdType + "]";
	}
}
```

Player.java

```
package in.techdive.game;
public class Player implements Comparable<Player>
{
		public Player(int id)
```

```
{
        this.playerId = id;
}
private int             playerId;
private String  playerName;
private int             points;
private String  result;
public int getPlayerId()
{
        return playerId;
}
public void setPlayerId(int playerId)
{
        this.playerId = playerId;
}
public String getPlayerName()
{
        return playerName;
}
public void setPlayerName(String playerName)
{
        this.playerName = playerName;
}
public int getPoints()
{
        return points;
}
public void setPoints(int points)
{
        this.points = points;
}
public String getResult()
{
        return result;
}
public void setResult(String result)
{
        this.result = result;
}
@Override
public int hashCode()
{
        final int prime = 31;
        int result = 1;
        result = prime * result + playerId;
        return result;
}
@Override
public boolean equals(Object obj)
{
        if (this == obj)
                return true;
```

```
		if (obj == null)
			return false;
		if (getClass() != obj.getClass())
			return false;
		Player other = (Player) obj;
		if (playerId != other.playerId)
			return false;
		return true;
	}
	@Override
	public int compareTo(Player o)
	{
		if (this.getPoints() == o.getPoints())
		{
			return 0;
		}
		else if (this.getPoints() > o.getPoints())
		{
			return 1;
		}
		else
			return -1;
	}
}
```

CardGame.java

```
package in.techdive.game;
import java.util.ArrayList;
import java.util.Collections;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Scanner;
import java.util.TreeMap;
public class CardGame implements Game
{
	private List<CARD>				cards;
	private List<Player>			players				= new ArrayList<Play
er>();
	private Map<Player,
List<CARD>> cardsPlayerMap			= new HashMap<Player, List<CARD>>();
	private int					currentPlayerIdx		= 0;
	private static final int		numberOfCardsPerPlayer	= 4;
	private int					numberOfPlayers			= 2;
	public int getNumberOfPlayers()
	{
		return numberOfPlayers;
	}
```

```
public List<Player> getPlayers()
{
        return players;
}
public CardGame()
{
        cards = CARD.getPackOfCards();
}
public void distributeCardsForPlayers(List<Player> plys)
{
        this.players = plys;
        CARD.shuffleCards(cards);
        if (cardsPlayerMap.size() == 0)
                cardsPlayerMap.clear();
        int m = 0;
        for (Player pl : players)
        {
                pl.setPoints(0);
                List<CARD> cds = new ArrayList<CARD>();
                int cardLimit = m + numberOfCardsPerPlayer;
                for (int i = m; i < cardLimit; i++)
                {
                        cds.add(cards.get(i));
                }
                m = cardLimit;
                cardsPlayerMap.put(pl, cds);
        }
}
public void playGame(int numberOfPlayers)
{
        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");
                        System.out.println("Chance for Player..." + player.getPlayerId());
                        System.out.print("Please provide your option : ");
                        Scanner in = new Scanner(System.in);
                        i = in.nextInt();
                        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;
						}
						else
						{
							if (maxCard.compareTo(c) < 0)
							{
								maxCard = c;
								maxPlayer = player;
							}
						}
						selCards.add(c);
						break;
					case 2:
						return;
				}
				System.out.println();
				s++;
			} while (s < players.size());
			if (maxPlayer.getPlayerId() > 0)
				maxPlayer.setPoints((maxPlayer.getPoints()) + 1);
			maxCard = null;
			maxPlayer = null;
			displayScores();
		}
	}
	private void displayScores()
	{
		for (Player pl : players)
		{
			System.out.println("Player " + pl.getPlayerId() + " Score ->
" + pl.getPoints());
		}
	}
	private void displayCardsForPlayer(Player pl)
	{
		int cards = cardsPlayerMap.get(pl).size();
		for (int i = 0; i < cards;)
		{
			System.out.print((++i) + " ");
		}
	}
	public void displayWinners()
	{
		Collections.sort(players);
		int maxPoints = 0;
		Map<String, List<Player>> playerPointsMap = new TreeMap<String, 
List<Player>>();
		for (Player p : players)
		{
			maxPoints = p.getPoints();
			if (playerPointsMap.get(maxPoints + "") != null)
			{
				List<Player> lst = playerPointsMap.get(maxPoints + "");
				lst.add(p);
				playerPointsMap.put(maxPoints + "", lst);
			}
			else
			{
				List<Player> lst = new ArrayList<Player>();
				lst.add(p);
				playerPointsMap.put(maxPoints + "", lst);
			}
		}

		String pts = new Integer(players.get(players.size() - 1).getPoints()).toString();
		if (playerPointsMap.get(pts) != null && playerPointsMap.get(pts).size() > 1)
		{
			System.out.println("Its a draw among the following players ");
			for (Player p : players)
			{
				System.out.println("Player -> " + p.getPlayerId());
			}
		}
		else if (playerPointsMap.get(pts) != null)
		{
			System.out.println("And the winner is :");
			System.out.println("Player -> 
" + playerPointsMap.get(pts).get(0).getPlayerId());
		}
	}
	private void createMultipleUser(int j)
	{
		if (players.size() != 0)
		{
			players.clear();
		}
		for (int i = 0; i < j; i++)
		{
			int id = i + 1;
			Player usr = new Player(id);
			players.add(usr);
		}
		distributeCardsForPlayers(players);
	}
	private Player getNextPlayer()
	{
		Player p = null;
		if (currentPlayerIdx == players.size())
		{
			currentPlayerIdx = 1;
			p = players.get(0);
		}
		else
		{
			p = players.get(currentPlayerIdx);
			currentPlayerIdx++;
		}
		return p;
	}
}
```

CardGameDemo.java

```
package in.techdive.game;
import java.util.Scanner;
public class CardGameDemo
{
	public CardGameDemo()
	{
	}
	/**
	 * @param args
	 */
	public static void main(String[] args)
	{
		CardGame sl = new CardGame();
		System.out.println("Card Game \n Player Options");
		System.out.println("1. Start Game \n  \n2. Exit Game");
		System.out.print("Please provide your option : ");
		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( 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);
                }
                System.out.println();
                System.out.println("Card Game \n Select User Options");
                System.out.println("1. Start Game \n2. Exit Game");
                System.out.print("Please provide your option : ");
            }
        }
}
```