Java lab Assignment 4

Om Varshney. AI ML B2. 21070126117.

Card.java

```
package Assn_4;

public class Card {
    private int type, value;
    private String[] cardType = {"Clubs", "Spades",
"Diamonds", "Hearts"};
    private String[] cardValue = {"Ace", "King", "Queen",
"Jack", "10","9", "8", "7", "6", "5", "4", "3", "2"};

    public Card(int type, int value) {
        this.type = type;
        this.value = value;
    }

    public String toString() {
        String finalCard = cardValue[value] + " of " +
cardType[type];
        return finalCard;
    }

    public static boolean equalTo(Card firstCard, Card
secondCard) {
        return firstCard.type == secondCard.type &&
firstCard.value == secondCard.value;
    }

    public static boolean greaterThan(Card firstCard, Card
secondCard) {
        return firstCard.type > secondCard.type ||
firstCard.value > secondCard.value;
    }
}
```

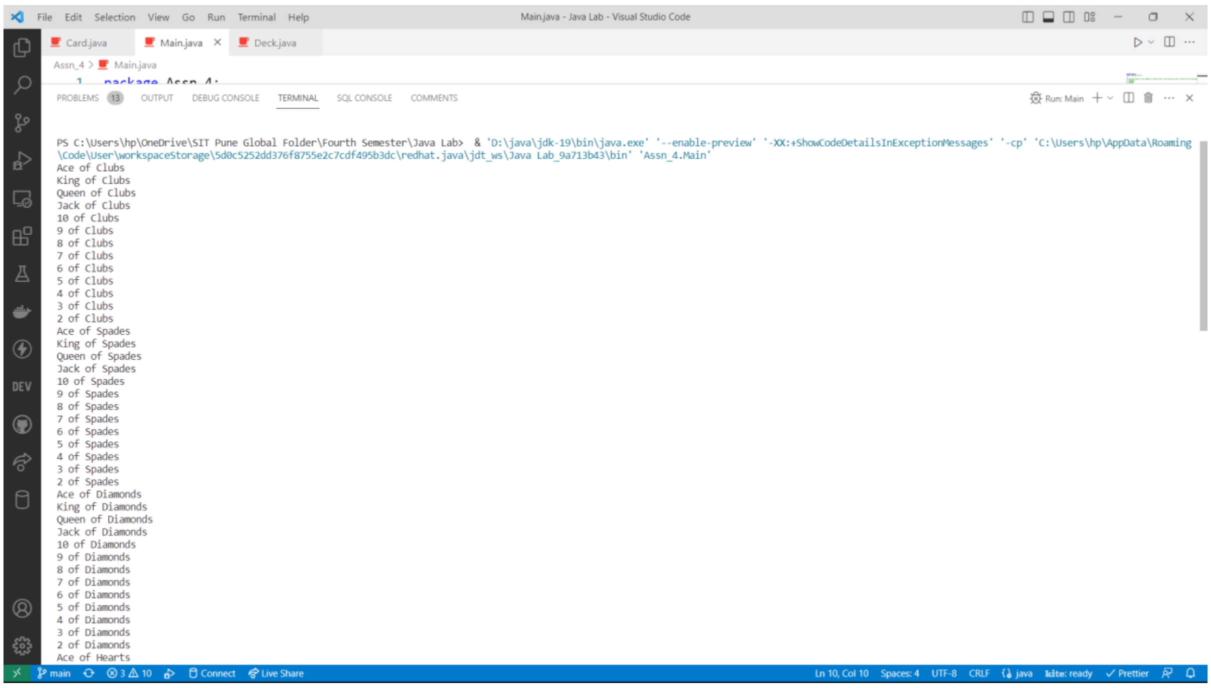
Deck.java

```
package Assn_4;
import java.util.Collections;
import java.util.Random;
import java.util.Vector;

public class Deck {
    private Vector<Card> cards;

    public Deck() {
        cards = new Vector<Card>();
        for(int a=0; a<=3; a++) {
            for(int b=0; b<=12; b++) {
                cards.add(new Card(a,b));
            }
        }
    }

    public Card drawRandomCard() {
        Random generator = new Random();
        int index = generator.nextInt(cards.size());
        return cards.remove(index);
    }

    public boolean sameCard(Card firstCard, Card secondCard)
{
        // Returns true if cards are equal. false otherwise.
        return Card.equalTo(firstCard, secondCard);
    }

    public static boolean compareCard(Card firstCard, Card
secondCard) {
        // Returns true if first card is greater than second
card. false otherwise.
        return Card.greaterThan(firstCard, secondCard);
    }

    public void sortDeck() {
        // A simple code to use bubble sort to swap the
cards.
        int maxLimit = cards.size();
        for (int i = 0; i < maxLimit; i++) {
            for (int j = 1; j < maxLimit - i - 1; j++) {
                if (compareCard(cards.get(j), cards.get(j +
1))) {
                    Collections.swap(cards, j, j + 1);
                }
            }
        }
    }

    public boolean findCard(Card card) {
        return cards.contains(card);
    }

    public void dealCards() {
        // Deal 5 random cards from the deck
        for (int i = 0; i < 5; i++) {
            int maxLimit = cards.size();
            int randIdx = (int) (Math.random() * (maxLimit +
1));

            System.out.println(cards.remove(randIdx));
        }
    }

    public void printDeck() {
        for (Card card : cards) {
            System.out.println(card);
        }
    }

    public String toString() {
        String result = "Cards remaining in deck: " + cards;
        return result;
    }
}
```

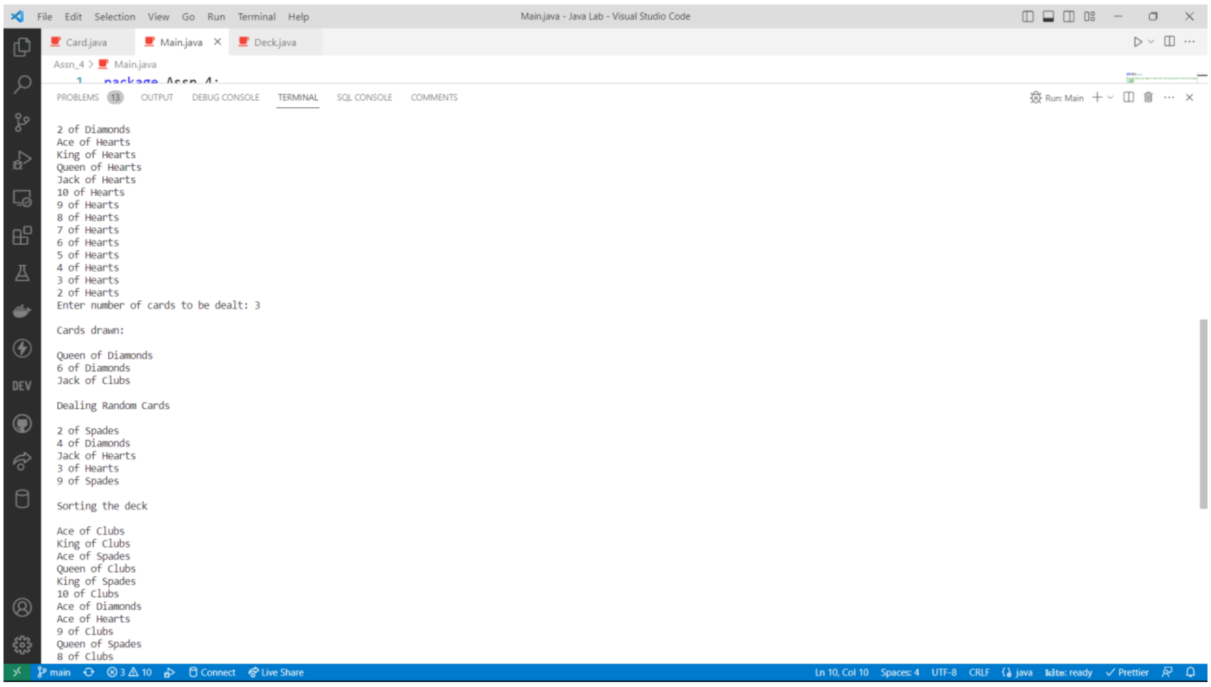
Main.java

```
package Assn_4;
import java.util.Scanner;

/*
In this Java Program we were supposed to simulate a deck of
cards using java vector. We have built the following
functions-
1. printDeck
2. printCard
3. sortDeck
4. dealCards
5. drawRandomCard
and others which complete the implementation of deck of
cards in java.
*/

public class Main {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        Card C;
        Deck deck = new Deck();
        deck.printDeck();
        System.out.print("Enter number of cards to be dealt:
");
        int numberCards = scan.nextInt();
        System.out.println("\nCards drawn: \n");
        for (int i = 0; i < numberCards; i++) {
            C = deck.drawRandomCard();
            System.out.println(C);
        }
        scan.close();
        System.out.println("\nDealing Random Cards\n");
        deck.dealCards();
        System.out.println("\nSorting the deck\n");
        deck.sortDeck();
        deck.printDeck();
    }
}
```

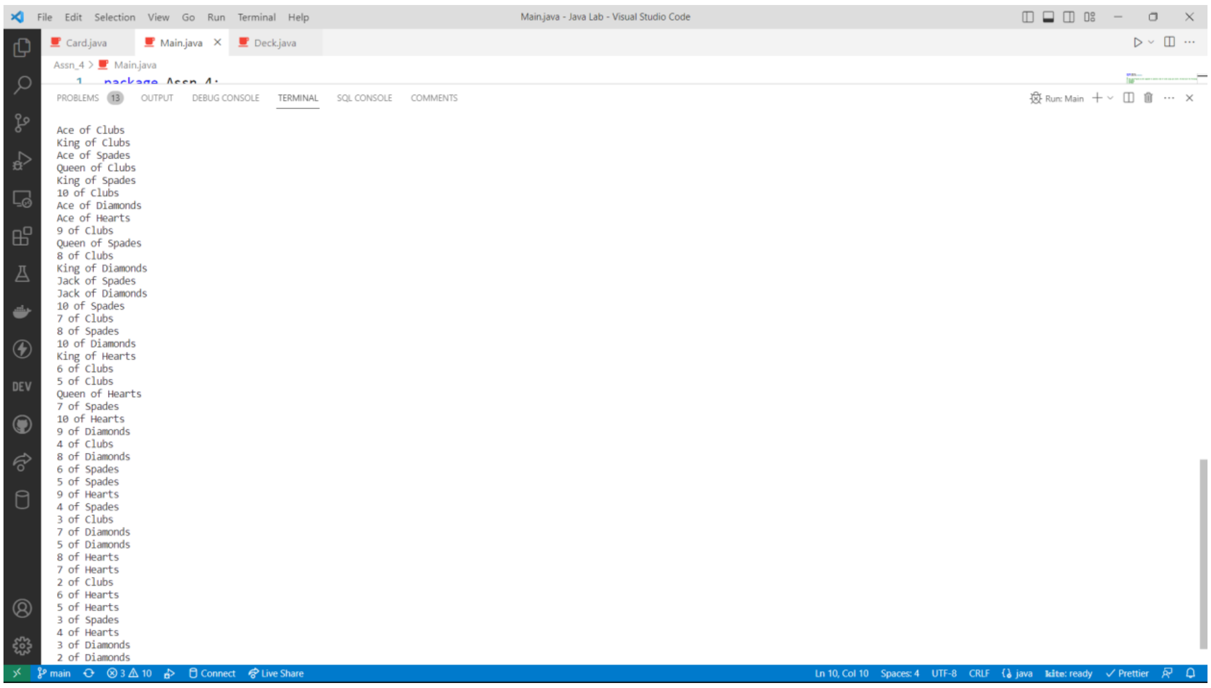
Outputs



```
PS C:\Users\hp\OneDrive\SIT Pune Global Folder\Fourth Semester\Java Lab>  & 'D:\java\jdk-19\bin\java.exe' '--enable-preview' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\hp\AppData\Roaming
\Code\User\workspaceStorage\5d0c5252dd376f8755e2c7cdf495b3dc\redhat.java\jdt_ws\Java Lab_9a713b43\bin' 'Assn_4.Main'
Ace of Clubs
King of Clubs
Queen of Clubs
Jack of Clubs
10 of Clubs
9 of Clubs
8 of Clubs
7 of Clubs
6 of Clubs
5 of Clubs
4 of Clubs
3 of Clubs
2 of Clubs
Ace of Spades
King of Spades
Queen of Spades
Jack of Spades
10 of Spades
9 of Spades
8 of Spades
7 of Spades
6 of Spades
5 of Spades
4 of Spades
3 of Spades
2 of Spades
Ace of Diamonds
King of Diamonds
Queen of Diamonds
Jack of Diamonds
10 of Diamonds
9 of Diamonds
8 of Diamonds
7 of Diamonds
6 of Diamonds
5 of Diamonds
4 of Diamonds
3 of Diamonds
2 of Diamonds
Ace of Hearts
```



```
2 of Diamonds
Ace of Hearts
King of Hearts
Queen of Hearts
Jack of Hearts
10 of Hearts
9 of Hearts
8 of Hearts
7 of Hearts
6 of Hearts
5 of Hearts
4 of Hearts
3 of Hearts
2 of Hearts
Enter number of cards to be dealt: 3

Cards drawn:

Queen of Diamonds
6 of Diamonds
Jack of Clubs

Dealing Random Cards

2 of Spades
4 of Diamonds
Jack of Hearts
3 of Hearts
9 of Spades

Sorting the deck

Ace of Clubs
King of Clubs
Ace of Spades
Queen of Clubs
King of Spades
10 of Clubs
Ace of Diamonds
Ace of Hearts
9 of Clubs
Queen of Spades
8 of Clubs
```



```
Ace of Clubs
King of Clubs
Ace of Spades
Queen of Clubs
King of Spades
10 of Clubs
Ace of Diamonds
Ace of Hearts
9 of Clubs
Queen of Spades
8 of Clubs
King of Diamonds
Jack of Spades
Jack of Diamonds
10 of Spades
7 of Clubs
8 of Spades
10 of Diamonds
King of Hearts
6 of Clubs
5 of Clubs
Queen of Hearts
7 of Spades
10 of Hearts
9 of Diamonds
4 of Clubs
8 of Diamonds
6 of Spades
5 of Spades
9 of Hearts
4 of Spades
3 of Clubs
7 of Diamonds
5 of Diamonds
8 of Hearts
7 of Hearts
2 of Clubs
6 of Hearts
5 of Hearts
3 of Spades
4 of Hearts
3 of Diamonds
2 of Diamonds
```

GitHub Repository Link

https://github.com/om-varshney/Java-Lab/tree/main/Assn_4