Assignment 4

Name: TANVI JATKAR

Division: AIML-A2

PRN: 21070126043

Write a menu-driven Java Program for the following:
There are 52 cards in a deck, each of which belongs to one of four suits and one of 13 ranks.
Represent a deck of cards as an array of Objects (*you may use the Vector class)

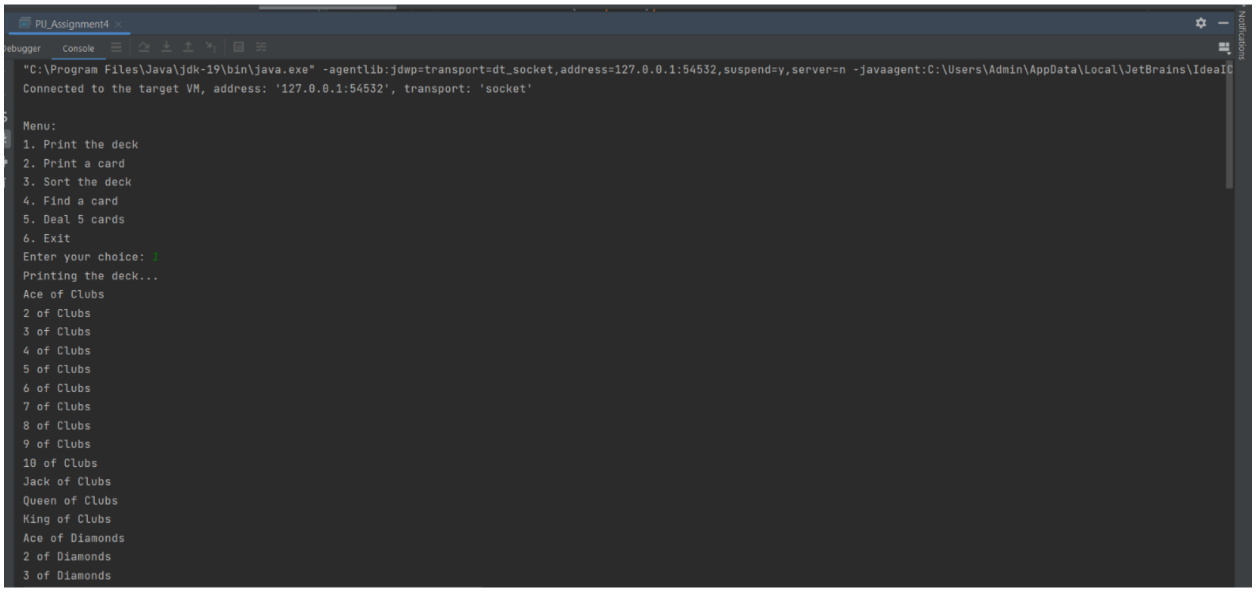
1. Use integers to encode the ranks and suits.
2. Have suitable default & parameterized constructors.
3. all data members to have private access.
4. The class 'Card' to have the following methods:

createDeck(), printCard(), printDeck (),sameCard(),compareCard(), sortCard(), findCard() which searches through an array or vector of Cards to see whether it contains a certain card, dealCards() function: to print 5 random cards from the existing deck.

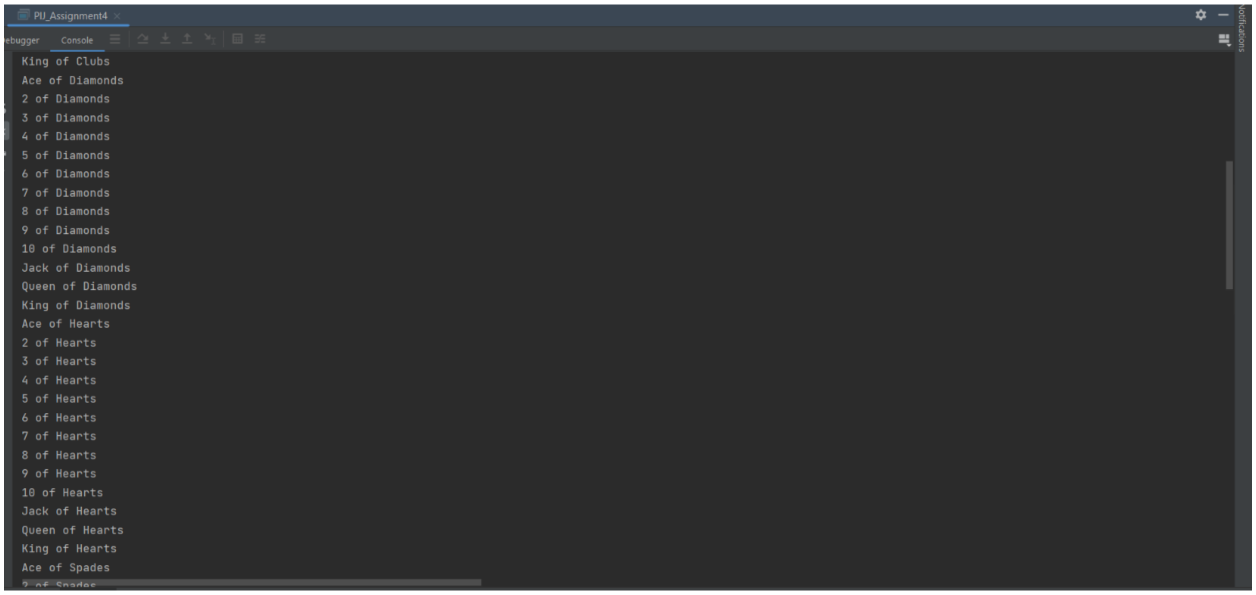
CODE:

```java
import java.util.Random;
import java.util.Vector;
import java.util.Scanner;

class Card {
    private int rank;
    private int suit;

    public Card() {
        this.rank = 0;
        this.suit = 0;
    }

    public Card(int rank, int suit) {
        this.rank = rank;
        this.suit = suit;
    }

    public int getRank() {
        return this.rank;
    }

    public int getSuit() {
        return this.suit;
    }

    public void printCard() {
        String[] ranks = {"Ace", "2", "3", "4", "5", "6", "7", "8", "9",
"10", "Jack", "Queen", "King"};
        String[] suits = {"Clubs", "Diamonds", "Hearts", "Spades"};
        System.out.println(ranks[this.rank] + " of " + suits[this.suit]);
    }

    public static Vector<Card> createDeck() {
        Vector<Card> deck = new Vector<Card>();
        for (int suit = 0; suit < 4; suit++) {
            for (int rank = 0; rank < 13; rank++) {
                deck.add(new Card(rank, suit));
            }
        }
        return deck;
    }

    public static boolean sameCard(Card card1, Card card2) {
        return (card1.rank == card2.rank && card1.suit == card2.suit);
    }

    public static int compareCard(Card card1, Card card2) {
        if (card1.rank > card2.rank) {
            return 1;
        } else if (card1.rank < card2.rank) {
            return -1;
        } else {
            if (card1.suit > card2.suit) {
                return 1;
            } else if (card1.suit < card2.suit) {
                return -1;
            } else {
                return 0;
            }
        }
    }

    public static void sortCard(Vector<Card> deck) {
        deck.sort((card1, card2) -> compareCard(card1, card2));
    }

    public static int findCard(Vector<Card> deck, Card card) {
        for (int i = 0; i < deck.size(); i++) {
            if (sameCard(deck.get(i), card)) {
                return i;
            }
        }
        return -1;
    }

    public static void printDeck(Vector<Card> deck) {
        for (Card card : deck) {
            card.printCard();
        }
    }

    public static void dealCards(Vector<Card> deck) {
        Random rand = new Random();
        System.out.println("Here are your 5 cards:");
        for (int i = 0; i < 5; i++) {
            int index = rand.nextInt(deck.size());
            deck.get(index).printCard();
            deck.remove(index);
        }
    }
}

public class PIJ_Assignment4
{
    public static void main(String[] args)
    {
        Scanner scanner = new Scanner(System.in);
        Vector<Card> deck = Card.createDeck();
        while (true) {
            System.out.println("\nMenu:");
            System.out.println("1. Print the deck");
            System.out.println("2. Print a card");
            System.out.println("3. Sort the deck");
            System.out.println("4. Find a card");
            System.out.println("5. Deal 5 cards");
            System.out.println("6. Exit");
            System.out.print("Enter your choice: ");
            int choice = scanner.nextInt();
            switch (choice) {
                case 1:
                    System.out.println("Printing the deck...");
                    Card.printDeck(deck);
                    break;

                case 2:
                    System.out.print("Enter the rank of the card (0-12): ");
                    int rank = scanner.nextInt();
                    System.out.print("Enter the suit of the card (0-3): ");
                    int suit = scanner.nextInt();
                    Card card = new Card(rank, suit);
                    System.out.print("The card is: ");
                    card.printCard();
                    break;

                case 3:
                    System.out.println("Sorting the deck...");
                    Card.sortCard(deck);
                    break;

                case 4:
                    System.out.print("Enter the rank of the card to find (0-
12): ");
                    rank = scanner.nextInt();
                    System.out.print("Enter the suit of the card to find (0-
3): ");
                    suit = scanner.nextInt();
                    card = new Card(rank, suit);
                    int index = Card.findCard(deck, card);
                    if (index == -1) {
                        System.out.println("Card not found.");
                    } else {
                        System.out.println("Card found at index " + index +
".");
                    }
                    break;

                case 5:
                    Card.dealCards(deck);
                    break;

                case 6:
                    System.out.println("Exiting...");
                    System.exit(0);
                    break;

                default:
                    System.out.println("Invalid choice. Please enter a number
between 1 and 6.");
            }
        }
    }
}
```

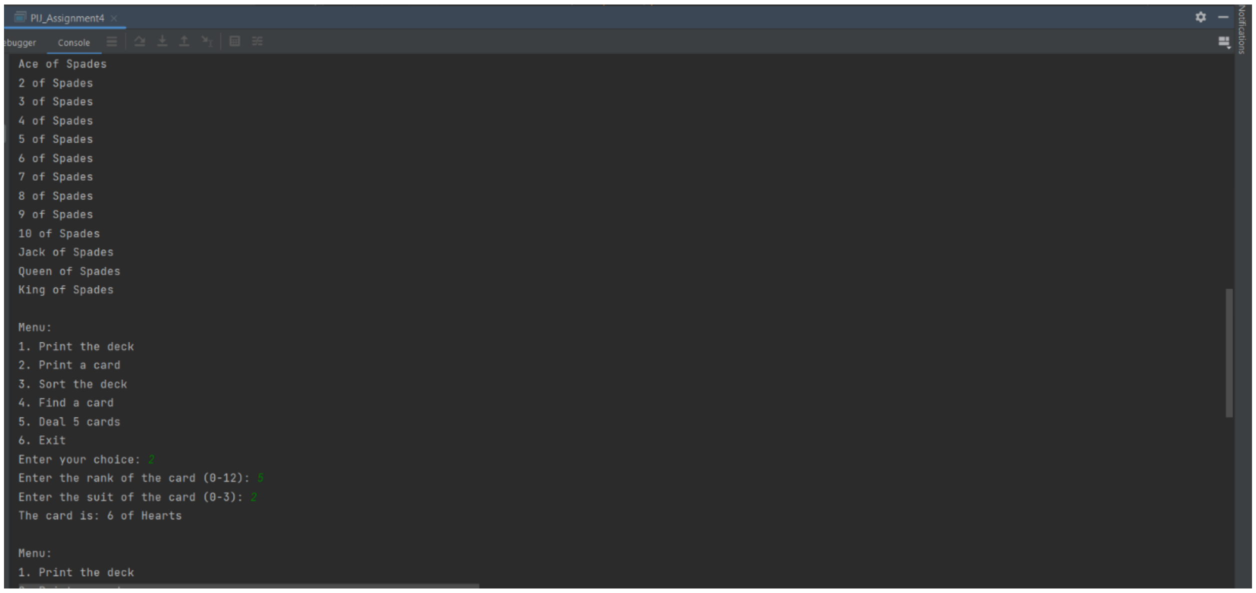
OUTPUT:



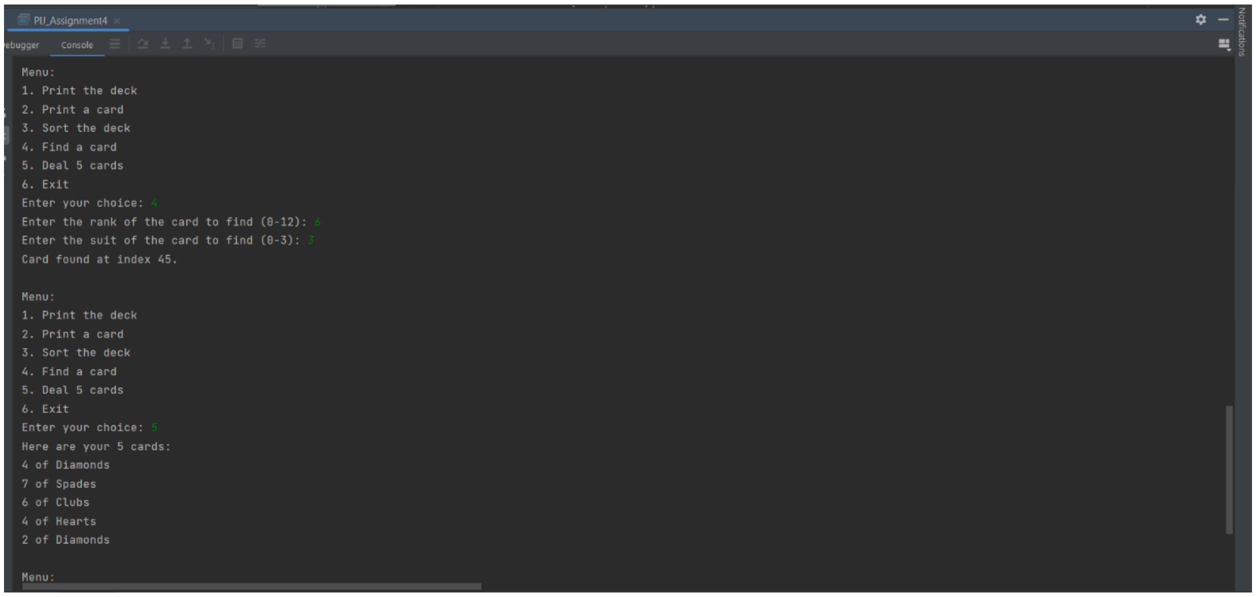
```
"C:\Program Files\Java\jdk-19\bin\java.exe" -agentlib:jdwp=transport=dt_socket,address=127.0.0.1:54532,suspend=y,server=n -javaagent:C:\Users\Admin\AppData\Local\JetBrains\IdeaIC
Connected to the target VM, address: '127.0.0.1:54532', transport: 'socket'

Menu:
1. Print the deck
2. Print a card
3. Sort the deck
4. Find a card
5. Deal 5 cards
6. Exit
Enter your choice: 1
Printing the deck...
Ace of Clubs
2 of Clubs
3 of Clubs
4 of Clubs
5 of Clubs
6 of Clubs
7 of Clubs
8 of Clubs
9 of Clubs
10 of Clubs
Jack of Clubs
Queen of Clubs
King of Clubs
Ace of Diamonds
2 of Diamonds
3 of Diamonds
```



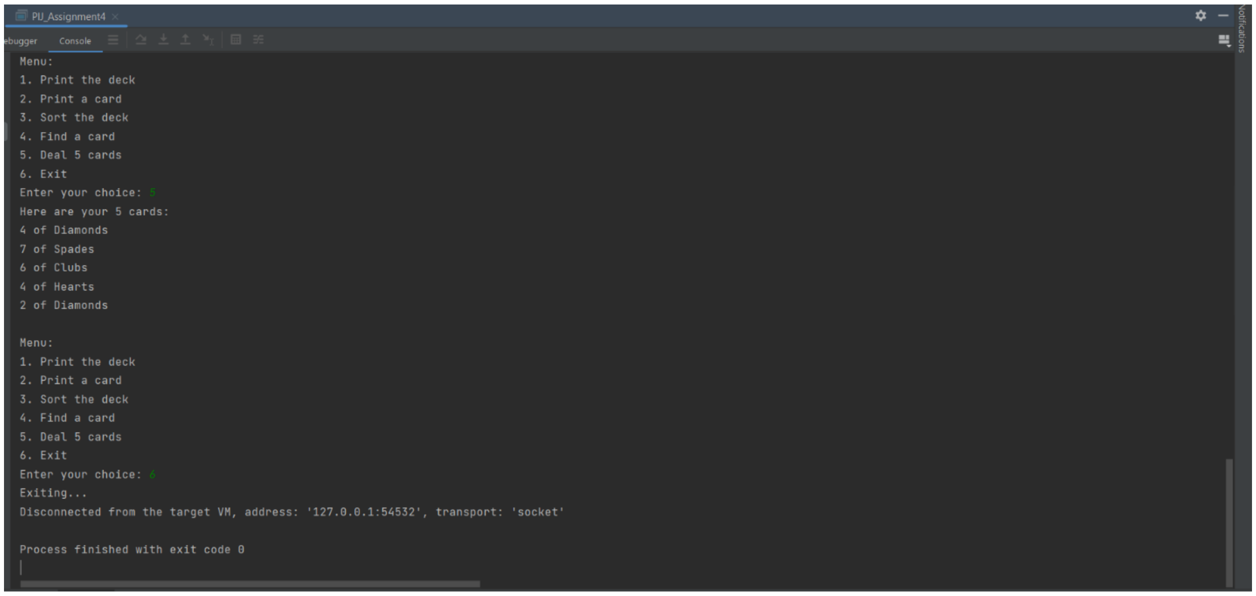
```
King of Clubs
Ace of Diamonds
2 of Diamonds
3 of Diamonds
4 of Diamonds
5 of Diamonds
6 of Diamonds
7 of Diamonds
8 of Diamonds
9 of Diamonds
10 of Diamonds
Jack of Diamonds
Queen of Diamonds
King of Diamonds
Ace of Hearts
2 of Hearts
3 of Hearts
4 of Hearts
5 of Hearts
6 of Hearts
7 of Hearts
8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



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

Menu:
1. Print the deck
2. Print a card
3. Sort the deck
4. Find a card
5. Deal 5 cards
6. Exit
Enter your choice: 2
Enter the rank of the card (0-12): 5
Enter the suit of the card (0-3): 2
The card is: 6 of Hearts

Menu:
1. Print the deck
```



```
Menu:
1. Print the deck
2. Print a card
3. Sort the deck
4. Find a card
5. Deal 5 cards
6. Exit
Enter your choice: 4
Enter the rank of the card to find (0-12): 6
Enter the suit of the card to find (0-3): 3
Card found at index 45.

Menu:
1. Print the deck
2. Print a card
3. Sort the deck
4. Find a card
5. Deal 5 cards
6. Exit
Enter your choice: 5
Here are your 5 cards:
4 of Diamonds
7 of Spades
6 of Clubs
4 of Hearts
2 of Diamonds

Menu:
```



```
Menu:
1. Print the deck
2. Print a card
3. Sort the deck
4. Find a card
5. Deal 5 cards
6. Exit
Enter your choice: 5
Here are your 5 cards:
4 of Diamonds
7 of Spades
6 of Clubs
4 of Hearts
2 of Diamonds

Menu:
1. Print the deck
2. Print a card
3. Sort the deck
4. Find a card
5. Deal 5 cards
6. Exit
Enter your choice: 6
Exiting...
Disconnected from the target VM, address: '127.0.0.1:54532', transport: 'socket'

Process finished with exit code 0
```