**Java Assignment 4**

**Sahil Goyal**

**AIML-B1**

**22070126094**

**2022-26**

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.

Should have methods:

a) createDeck() //Can also add this method as constructor

b) printDeck()

c) printCard()

d) sameCard() //Card which is from same suit

e) compareCard() //Card having same rank or number

f) findCard() //Search for particular card

g) dealCard() //Print 5 random cards

h) shuffleDeck() //Randomize the deck

Code:

// Card.java

public class Card {

private char value;

private String suit;

// Constructor to initialize card with value and suit

public Card(char value, String suit) {

this.value = value;

this.suit = suit;

}

// Getter method to retrieve the value of the card

public int getValue() {

return value;

}

// Getter method to retrieve the suit of the card

public String getSuit() {

return suit;

}

// Setter method to set the suit of the card

public void setSuit(String suit) {

this.suit = suit;

}

// Setter method to set the value of the card

public void setValue(char value) {

this.value = value;

}

// toString method to represent the card as a string

@Override

public String toString() {

return value + " of " + suit;

}

}

//Deck.java

import java.util.\*;

public class Deck {

private ArrayList<Card> cards;

// Constructor to initialize the deck with 52 cards and shuffle them

public Deck() {

this.cards = new ArrayList<Card>();

// Define possible values and suits for cards

String[] values = { "A", "2", "3", "4", "5", "6", "7", "8", "9", "10", "J", "Q", "K" };

String[] suits = { "Hearts", "Diamonds", "Clubs", "Spades" };

// Create cards for each combination of value and suit

for (String suit : suits) {

for (String value : values) {

char charValue;

if (value.equals("A") || value.equals("J") || value.equals("Q") || value.equals("K")) {

charValue = value.charAt(0);

} else {

charValue = value.charAt(0);

}

this.cards.add(new Card(charValue, suit));

}

}

// Shuffle the deck

Collections.shuffle(this.cards);

}

// Method to retrieve the entire deck and print all cards in a line

public void printDeck() {

StringBuilder deckString = new StringBuilder();

for (Card card : cards) {

deckString.append(card.toString()).append(", ");

}

System.out.println(deckString.toString());

}

// Method to get the top card of the deck

public Card getCard() {

return cards.get(0);

}

// Method to find cards of the same suit as provided

public ArrayList<Card> sameCard(String suit) {

ArrayList<Card> sameSuitCards = new ArrayList<>();

for (Card card : cards) {

if (card.getSuit().equalsIgnoreCase(suit)) {

sameSuitCards.add(card);

}

}

return sameSuitCards;

}

// Method to compare cards having the same rank or number as provided

public ArrayList<Card> compareCard(char value) {

ArrayList<Card> sameRankCards = new ArrayList<>();

for (Card card : cards) {

if (card.getValue() == value) {

sameRankCards.add(card);

}

}

return sameRankCards;

}

// Method to search for a particular card with given suit and value

public Card findCard(String suit, char value) {

for (Card card : cards) {

if (card.getSuit().equalsIgnoreCase(suit) && card.getValue() == value) {

return card;

}

}

return null;

}

// Method to deal 5 random cards from the deck

public ArrayList<Card> dealCard() {

ArrayList<Card> dealtCards = new ArrayList<>();

Random rand = new Random();

for (int i = 0; i < 5; i++) {

int index = rand.nextInt(cards.size());

dealtCards.add(cards.remove(index));

}

return dealtCards;

}

// Method to shuffle the deck

public void shuffleDeck() {

Collections.shuffle(cards);

}

}

// Main.java

// Sahil Goyal

// AIML-B1

// 2022-26

// 22070126094

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in); // Create a Scanner object to read user input

Deck deck = new Deck(); // Create a new deck of cards

int choice; // Variable to store the user's choice

do {

// Display the menu options to the user

System.out.println("\nMenu:");

System.out.println("1. Display the entire deck");

System.out.println("2. Find cards of the same suit");

System.out.println("3. Compare cards with the same rank or number");

System.out.println("4. Find a particular card");

System.out.println("5. Deal 5 random cards");

System.out.println("6. Shuffle the deck");

System.out.println("7. Exit");

System.out.print("Enter your choice: ");

choice = scanner.nextInt(); // Read the user's choice

scanner.nextLine(); // Consume newline character

// Perform actions based on the user's choice

switch (choice) {

case 1:

System.out.println("Entire Deck:");

deck.printDeck(); // Display the entire deck

break;

case 2:

System.out.print("Enter suit to find cards: ");

String suit = scanner.nextLine(); // Read the suit from the user

System.out.println(deck.sameCard(suit)); // Display cards with the same suit

break;

case 3:

System.out.print("Enter value to compare cards: ");

char value = scanner.next().charAt(0); // Read the value to compare

System.out.println(deck.compareCard(value)); // Display cards with the same value

break;

case 4:

System.out.print("Enter suit of the card: ");

String cardSuit = scanner.nextLine(); // Read the suit of the card

// Consume the newline character left in the input buffer

System.out.print("Enter value of the card: ");

char cardValue = scanner.next().charAt(0); // Read the value of the card

Card foundCard = deck.findCard(cardSuit, cardValue); // Find the card

if (foundCard != null) {

System.out.println("Card found: " + foundCard); // Display the found card

}

else {

System.out.println("Card not found."); // Display message if card not found

}

break;

case 5:

System.out.println("Dealt cards: " + deck.dealCard()); // Deal 5 random cards

break;

case 6:

deck.shuffleDeck(); // Shuffle the deck

System.out.println("Deck shuffled.");

break;

case 7:

System.out.println("Exiting..."); // Exit the program

break;

default:

System.out.println("Invalid choice!"); // Display message for invalid choice

}

} while (choice != 7); // Repeat until the user chooses to exit

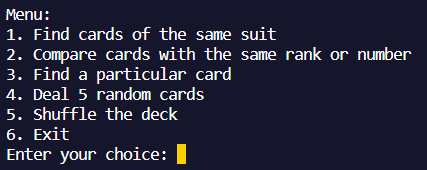
scanner.close(); // Close the scanner to prevent resource leak

}

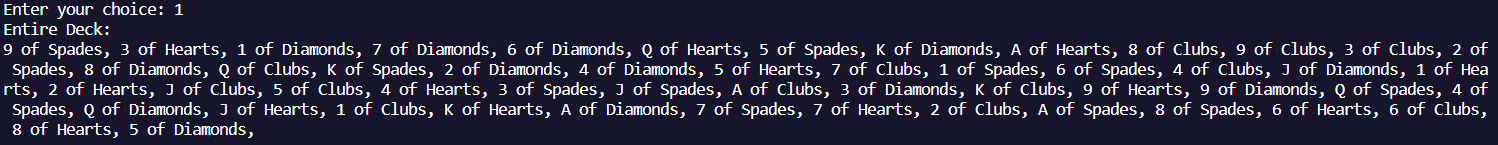
}

Output:

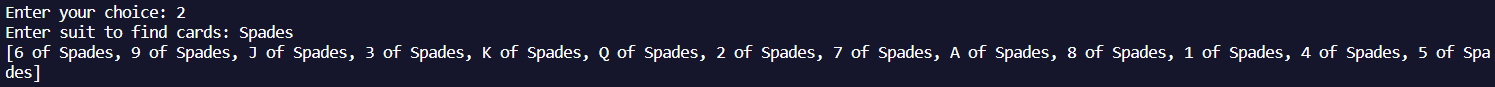
Menu



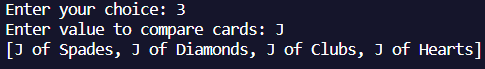
Choice 1:



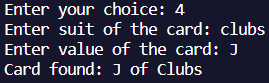
Choice 2:



Choice 3:



Choice 4:

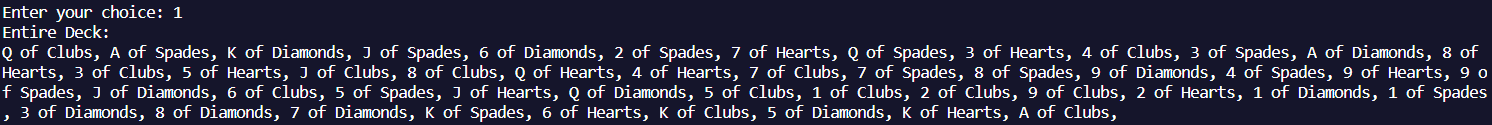


Choice 5:



Choice 6:

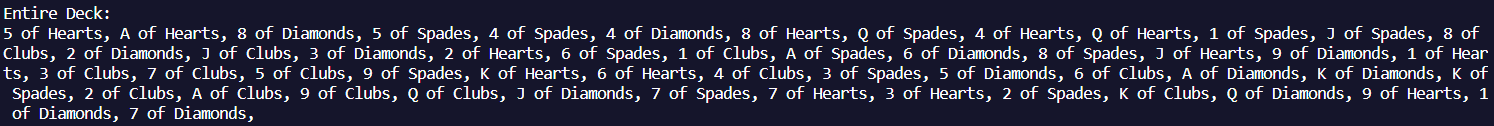
Previous Deck;



Shuffling;



New Deck;



Choice 7:



Check out my repository on Github:

[https://github.com/sahilgoyal7214/programming-in-java/tree/main/Assignment\_4](https://github.com/sahilgoyal7214/programming-in-java/tree/main/Assignment_4%20)