Nombre: Heriberto Cárdenas Tamez

Matricula: AL07099215

Materia: Computación en java

Maestro: Jose Alfredo Jimenez Hernandez

Fecha: 22 de septiembre del 2025

Implementación de una Baraja de Póker en Java

1. Introducción

El presente documento tiene como objetivo explicar el desarrollo de un programa en Java que simula el funcionamiento de una baraja de póker. Se implementaron clases y métodos que permiten manipular la baraja, siguiendo las reglas reales de las cartas de póker. Este reporte incluye una explicación de la lógica implementada, ejemplos de ejecución y evidencia del correcto funcionamiento del programa.

2. Definición de la Baraja de Póker

La baraja de póker utilizada en este proyecto está compuesta por un total de 52 cartas. Estas cartas se dividen en cuatro palos: tréboles, corazones, picas y diamantes. Cada palo cuenta con 13 valores diferentes, que van desde el número 2 hasta el 10, además de las cartas A (as), J (jack), Q (queen) y K (king). Los palos de tréboles y picas corresponden al color negro, mientras que los palos de corazones y diamantes corresponden al color rojo.

3. Diseño del Programa

El programa fue diseñado utilizando el lenguaje de programación Java y se compone de tres clases principales:
1. Card: representa una carta individual con tres atributos: palo, color y valor.
2. Deck: representa el conjunto completo de cartas. Contiene la lógica para inicializar las 52 cartas de la baraja y proporciona métodos para manipularlas.
3. Main: clase principal que ejecuta el programa y prueba los métodos definidos en la clase Deck.

4. Métodos implementados

La clase Deck cuenta con cuatro métodos principales que simulan las acciones que se pueden realizar con una baraja:

- shuffle(): mezcla aleatoriamente todas las cartas de la baraja utilizando el método Collections.shuffle(). Este método imprime en pantalla el mensaje 'Se mezcló el Deck'.

- head(): muestra la primera carta del deck y la elimina de la colección. Además, indica cuántas cartas quedan disponibles después de la operación.

- pick(): selecciona una carta al azar dentro de la baraja y la elimina. También muestra en

pantalla el palo, color y valor de la carta seleccionada, junto con el número de cartas restantes.

- hand(): devuelve una mano compuesta por 5 cartas, las cuales son eliminadas del deck. Se muestran en pantalla las cinco cartas junto con el número total de cartas restantes después de la operación.

5. Evidencia de funcionamiento

A continuación, se presentan ejemplos de la salida obtenida al ejecutar el programa:

➞ Ejemplo método shuffle():
Se mezcló el Deck.

➞ Ejemplo método head():
Corazones,Rojo,K
Quedan 51 cartas en deck

➞ Ejemplo método pick():
Tréboles,Negro,7
Quedan 50 cartas en deck

➞ Ejemplo método hand():
Diamantes,Rojo,3
Picas,Negro,A
Corazones,Rojo,10
Tréboles,Negro,5
Diamantes,Rojo,K
Quedan 45 cartas en deck

```
@hct47 ➞/workspaces/actividad-3 (main) $ javac *.java && java Main
Se mezcló el Deck.
Corazones,Rojo,Q
Quedan 51 cartas en deck
Tréboles,Negro,Q
Quedan 50 cartas en deck
Diamantes,Rojo,3
Diamantes,Rojo,4
Corazones,Rojo,6
Diamantes,Rojo,7
Picas,Negro,8
Quedan 45 cartas en deck
@hct47 ➞/workspaces/actividad-3 (main) $
```

[Vista previa] README.md | Main.java U | Deck.java U | Car

Main.java

```java
public class Main {
    public static void main(String[] args) {
        Deck deck = new Deck();

        // Pruebas de funcionamiento
        deck.shuffle();  // Mezclar baraja
        deck.head();     // Mostrar primera carta
        deck.pick();     // Mostrar carta aleatoria
        deck.hand();     // Mostrar mano de 5 cartas
    }
}

```

[Vista previa] README.md | Main.java U | Deck.java U | Card.java U

Deck.java

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

public class Deck {
    private ArrayList<Card> cards;

    // Constructor: inicializa las 52 cartas
    public Deck() {
        cards = new ArrayList<>();
        String[] palos = {"Corazones", "Diamantes", "Tréboles", "Picas"};
        String[] colores = {"Rojo", "Rojo", "Negro", "Negro"};
        String[] valores = {"A", "2", "3", "4", "5", "6", "7", "8", "9", "10", "J", "Q", "K"};

        for (int i = 0; i < palos.length; i++) {
            for (String valor : valores) {
                cards.add(new Card(palos[i], colores[i], valor));
            }
        }
    }

    // a) Mezcla aleatoriamente el deck
    public void shuffle() {
        Collections.shuffle(cards);
        System.out.println("Se mezcló el Deck.");
    }

    // b) Muestra la primera carta y la elimina
    public void head() {
        if (cards.isEmpty()) {
            System.out.println("Ya no hay cartas en el deck.");
            return;
        }
        Card primera = cards.remove(0);
        System.out.println(primera);
        System.out.println("Quedan " + cards.size() + " cartas en deck");
    }

    // c) Selecciona una carta al azar y la elimina
    public void pick() {
        if (cards.isEmpty()) {
            System.out.println("Ya no hay cartas en el deck.");
```

```java
    public void pick() {
        if (cards.isEmpty()) {
            System.out.println("Ya no hay cartas en el deck.");
            return;
        }
        Random random = new Random();
        int index = random.nextInt(cards.size());
        Card seleccionada = cards.remove(index);
        System.out.println(seleccionada);
        System.out.println("Quedan " + cards.size() + " cartas en deck");
    }

    // d) Devuelve una mano de 5 cartas y las elimina del deck
    public void hand() {
        if (cards.size() < 5) {
            System.out.println("No hay suficientes cartas para repartir una mano.");
            return;
        }
        for (int i = 0; i < 5; i++) {
            Card carta = cards.remove(0);
            System.out.println(carta);
        }
        System.out.println("Quedan " + cards.size() + " cartas en deck");
    }
}

```

Card.java

```java
public class Card {
    private String palo;
    private String color;
    private String valor;

    // Constructor
    public Card(String palo, String color, String valor) {
        this.palo = palo;
        this.color = color;
        this.valor = valor;
    }

    // Método para mostrar la carta en formato solicitado
    @Override
    public String toString() {
        return palo + "," + color + "," + valor;
    }
}

```

6. Repositorio remoto

El código fuente del proyecto ha sido subido a un repositorio en línea, el cual permite tener control de versiones y facilita la colaboración con otros desarrolladores. Además, asegura la disponibilidad del proyecto en cualquier momento.

👉 Enlace al repositorio:

7. Conclusión

El programa cumple con los requisitos establecidos, ya que simula correctamente una baraja de póker utilizando programación orientada a objetos en Java. Hace uso del Collection Framework para manejar las cartas y permite realizar las operaciones de mezclar, mostrar y eliminar cartas de forma dinámica. Además, respeta las reglas de color y valor de una baraja real, mostrando en consola los resultados de manera clara y ordenada.