Materia:

Programación I

Nombre:

Juan Carlos Contreras

Matricula:

2022-0390

Grupo #7

Profesor:

Freidy Nuñez Perez

Fecha: 05-10-2023

Sistema de pagos bancarios.

1. Crear una clase abstracta "CuentaBancaria" que tenga los siguientes atributos: saldo, número de cuenta y nombre del titular.
2. Crear una clase "CuentaCorriente" que herede de "CuentaBancaria" y tenga un atributo adicional "límite de descubierto".
3. Crear una clase "CuentaDeAhorros" que también herede de "CuentaBancaria" y tenga un atributo adicional "tasa de interés".
4. Crear una interfaz "IMovimientos" que tenga los siguientes métodos: depositar y retirar.
5. Implementar la interfaz "IMovimientos" en ambas clases hijas.
6. En la clase "CuentaCorriente", sobrecargar el método retirar para tener en cuenta el límite de descubierto.
7. Crear una clase "Banco" que tenga una lista de cuentas bancarias y tenga los siguientes métodos: agregar cuenta, eliminar cuenta, y mostrar información de cuentas.
8. En la clase "Banco", crear un método que permita interactuar con los objetos de las clases "CuentaCorriente" y "CuentaDeAhorros" por medio de un menú que permita depositar, retirar, mostrar información y salir.
9. En la clase "Banco", utilizar la palabra reservada "super" para llamar los métodos de la clase padre.
10. Utilizar sobrecarga de constructores y métodos según sea necesario.

CuentaBancaria.java | CuentaCorriente.java | CuentaDeAhorros.java | IMovimientos.java | Banco.java

```java
package Banco;

public abstract class CuentaBancaria {
    private String saldo;
    private int numeroCuenta;
    private String nombreTitular;

    public CuentaBancaria(String titular, int numeroCuenta2, String saldoInicial) {
        this.saldo = titular;
        this.numeroCuenta = numeroCuenta2;
        this.nombreTitular = saldoInicial;
    }

    public CuentaBancaria(String string, double saldoInicial, String string2) {
        // TODO Auto-generated constructor stub
    }

    public CuentaBancaria(String titular, String numeroCuenta2, double saldoInicial) {
        // TODO Auto-generated constructor stub
    }

    public String getSaldo() {
        return saldo;
    }

    public void setSaldo(String saldo) {
        this.saldo = saldo;
    }

    public int getNumeroCuenta() {
        return numeroCuenta;
    }

    public void setNumeroCuenta(int numeroCuenta) {
        this.numeroCuenta = numeroCuenta;
    }

    public String getNombreTitular() {
```

CuentaBancaria.java | CuentaCorriente.java | CuentaDeAhorros.java | IMovimientos.java | Banco.java

```
    public int getNumeroCuenta() {
        return numeroCuenta;
    }

    public void setNumeroCuenta(int numeroCuenta) {
        this.numeroCuenta = numeroCuenta;
    }

    public String getNombreTitular() {
        return nombreTitular;
    }

    public void setNombreTitular(String nombreTitular) {
        this.nombreTitular = nombreTitular;
    }

    public abstract void depositar(double cantidad);

    public abstract boolean retirar(double cantidad);

    public String toString() {
        return "CuentaBancaria [saldo=" + saldo + ", numeroCuenta=" + numeroCuenta + ", nombreTitular=" + nombreTitular
                + "]";
    }

    protected static void add(CuentaDeAhorros nuevaCuentaDeAhorros) {
        // TODO Auto-generated method stub

    }

    protected static void add(CuentaCorriente nuevaCuentaCorriente) {
        // TODO Auto-generated method stub

    }

    protected static boolean isEmpty() {
        // TODO Auto-generated method stub
```

*CuentaBancaria.java | CuentaCorriente.java | CuentaDeAhorros.java | IMovimientos.java | Banco.java

```
package Banco;

public class CuentaCorriente extends CuentaBancaria implements IMovimientos {
    private double limiteDescubierto;

    public CuentaCorriente(String titular, String numeroCuenta, double saldoInicial, double limiteDescubierto) {
        super(titular, numeroCuenta, saldoInicial);
        this.limiteDescubierto = limiteDescubierto;
    }

    public double getLimiteDescubierto() {
        return limiteDescubierto;
    }

    public void setLimiteDescubierto(double limiteDescubierto) {
        this.limiteDescubierto = limiteDescubierto;
    }

    @Override
    public void depositar(double cantidad) {
        setSaldo(getSaldo() + cantidad);
    }

    @Override
    public void retirar(double cantidad) {
        if (cantidad > getSaldo() + limiteDescubierto) {
            System.out.println("No se puede retirar esta cantidad. Límite de descubierto excedido.");
        } else {
            setSaldo(getSaldo() - cantidad);
        }
    }

    public String toString() {
        return "CuentaCorriente [saldo=" + getSaldo() + ", numeroCuenta=" + getNumeroCuenta() + ", nombreTitular="
                + getNombreTitular() + ", limiteDescubierto=" + limiteDescubierto + "]";
    }
}
```

```java
        this.tasaInteresAnual = tasaInteresAnual;
    }

    public double getTasaInteresAnual() {
        return tasaInteresAnual;
    }

    public void setTasaInteresAnual(double tasaInteresAnual) {
        this.tasaInteresAnual = tasaInteresAnual;
    }

    public void aplicarInteresMensual() {
        double interesMensual = getSaldo() * tasaInteresAnual / 12;
        setSaldo(getSaldo() + interesMensual);
    }

    @Override
    public void depositar(double cantidad) {
        setSaldo(getSaldo() + cantidad);
        aplicarInteresMensual();
    }

    @Override
    public void retirar(double cantidad) {
        if (cantidad > getSaldo()) {
            System.out.println("No se puede retirar esta cantidad. Saldo insuficiente.");
        } else {
            setSaldo(getSaldo() - cantidad);
            aplicarInteresMensual();
        }
    }

    public String toString() {
        return "CuentaDeAhorros [saldo=" + getSaldo() + ", numeroCuenta=" + getNumeroCuenta() + ", nombreTitular="
                + getNombreTitular() + ", tasaInteresAnual=" + tasaInteresAnual + "]";
    }
}
```

```java
package Banco;

public interface IMovimientos {
    void depositar(double cantidad);
    void retirar(double cantidad);
}
```

Creación de un sistema de biblioteca virtual.

1. Crear una clase abstracta "MaterialBibliográfico" que tenga los siguientes atributos: título, autor, año de publicación y número de páginas.
2. Crear una clase "Libro" que herede de "MaterialBibliográfico" y tenga un atributo adicional "editorial".
3. Crear una clase "Revista" que también herede de "MaterialBibliográfico" y tenga un atributo adicional "issn".
4. Crear una clase abstracta "Prestable" que herede de "MaterialBibliográfico" y tenga un atributo adicional "estado" (disponible o no disponible).

5. Crear una clase "DVD" que herede de "Prestable" y tenga un atributo adicional "duración".
6. Crear una interfaz "IPrestable" que tenga los siguientes métodos: prestar y devolver.
7. Implementar la interfaz "IPrestable" en la clase "Prestable".
8. En la clase "Prestable", sobrecargar el método prestar para tener en cuenta el estado de disponibilidad del material.
9. Crear una clase "BibliotecaVirtual" que tenga una lista de materiales bibliográficos y tenga los siguientes métodos: agregar material, eliminar material, y mostrar información de materiales.
10. En la clase "BibliotecaVirtual", crear un método que permita interactuar con los objetos de las clases "Libro", "Revista", "DVD" por medio de un menú que permita prestamo, devolución, mostrar información y salir.
11. En la clase "BibliotecaVirtual", utilizar la palabra reservada "super" para llamar los métodos de la clase padre.
12. Utilizar sobrecarga de constructores y métodos según sea necesario.

```
// Clase abstracta "MaterialBibliográfico"
public abstract class MaterialBibliográfico {
    protected String título;
    protected String autor;
    protected int añoPublicación;
    protected int númeroPáginas;

    public MaterialBibliográfico(String título, String autor, int añoPublicación, int
númeroPáginas) {
        this.título = título;
        this.autor = autor;
        this.añoPublicación = añoPublicación;
        this.númeroPáginas = númeroPáginas;
    }
```

```
    public abstract void mostrarInformación();
}

// Clase "Libro" que hereda de "MaterialBibliográfico"
public class Libro extends MaterialBibliográfico {
    private String editorial;

    public Libro(String título, String autor, int añoPublicación, int númeroPáginas, String
editorial) {
        super(título, autor, añoPublicación, númeroPáginas);
        this.editorial = editorial;
    }

    @Override
    public void mostrarInformación() {
        System.out.println("Libro:");
        System.out.println("Título: " + título);
        System.out.println("Autor: " + autor);
        System.out.println("Año de Publicación: " + añoPublicación);
        System.out.println("Número de Páginas: " + númeroPáginas);
        System.out.println("Editorial: " + editorial);
    }
}

// Clase "Revista" que hereda de "MaterialBibliográfico"
public class Revista extends MaterialBibliográfico {
    private String issn;

    public Revista(String título, String autor, int añoPublicación, int númeroPáginas,
String issn) {
```

```
        super(título, autor, añoPublicación, númeroPáginas);
        this.issn = issn;
    }

    @Override
    public void mostrarInformación() {
        System.out.println("Revista:");
        System.out.println("Título: " + título);
        System.out.println("Autor: " + autor);
        System.out.println("Año de Publicación: " + añoPublicación);
        System.out.println("Número de Páginas: " + númeroPáginas);
        System.out.println("ISSN: " + issn);
    }
}
// Clase abstracta "MaterialBibliográfico"
public abstract class MaterialBibliográfico {
    protected String título;
    protected String autor;
    protected int añoPublicación;
    protected int númeroPáginas;

    public MaterialBibliográfico(String título, String autor, int añoPublicación, int
númeroPáginas) {
        this.título = título;
        this.autor = autor;
        this.añoPublicación = añoPublicación;
        this.númeroPáginas = númeroPáginas;
    }

    public abstract void mostrarInformación();
}
```

```
// Clase abstracta "Prestable" que hereda de "MaterialBibliográfico"
public abstract class Prestable extends MaterialBibliográfico {
    protected boolean disponible;

    public Prestable(String título, String autor, int añoPublicación, int númeroPáginas,
boolean disponible) {
        super(título, autor, añoPublicación, númeroPáginas);
        this.disponible = disponible;
    }

    public abstract void prestar();
    public abstract void devolver();

    public boolean isDisponible() {
        return disponible;
    }
}

// Clase "DVD" que hereda de "Prestable"
public class DVD extends Prestable {
    private int duración;

    public DVD(String título, String autor, int añoPublicación, int númeroPáginas, boolean
disponible, int duración) {
        super(título, autor, añoPublicación, númeroPáginas, disponible);
        this.duración = duración;
    }

    @Override
    public void mostrarInformación() {
```

```
        System.out.println("DVD:");
        System.out.println("Título: " + título);
        System.out.println("Autor: " + autor);
        System.out.println("Año de Publicación: " + añoPublicación);
        System.out.println("Número de Páginas: " + númeroPáginas);
        System.out.println("Duración (minutos): " + duración);
        System.out.println("Estado: " + (disponible ? "Disponible" : "No disponible"));
    }

    @Override
    public void prestar() {
        if (disponible) {
            disponible = false;
            System.out.println("El DVD ha sido prestado.");
        } else {
            System.out.println("El DVD no está disponible en este momento.");
        }
    }

    @Override
    public void devolver() {
        if (!disponible) {
            disponible = true;
            System.out.println("El DVD ha sido devuelto.");
        } else {
            System.out.println("El DVD ya está disponible.");
        }
    }
}

// Interfaz "IPrestable" que define los métodos prestar y devolver
```

```
public interface IPrestable {
    void prestar();
    void devolver();
}
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

public class BibliotecaVirtual {
    private List<Prestable> materiales;

    public BibliotecaVirtual() {
        materiales = new ArrayList<>();
    }

    public void agregarMaterial(Prestable material) {
        materiales.add(material);
    }

    public void eliminarMaterial(Prestable material) {
        materiales.remove(material);
    }

    public void mostrarInformacionMateriales() {
        for (Prestable material : materiales) {
            material.mostrarInformación();
        }
    }

    public void interactuarConMateriales() {
        Scanner scanner = new Scanner(System.in);
```

```
    int opcion;

    do {
        System.out.println("Menú de opciones:");
        System.out.println("1. Prestamo");
        System.out.println("2. Devolución");
        System.out.println("3. Mostrar información de materiales");
        System.out.println("4. Salir");
        System.out.print("Seleccione una opción: ");
        opcion = scanner.nextInt();

        switch (opcion) {
            case 1:
                prestarMaterial();
                break;
            case 2:
                devolverMaterial();
                break;
            case 3:
                mostrarInformacionMateriales();
                break;
            case 4:
                System.out.println("Saliendo del programa.");
                break;
            default:
                System.out.println("Opción no válida. Intente nuevamente.");
        }
    } while (opcion != 4);
}

private void prestarMaterial() {
```

```
    Scanner scanner = new Scanner(System.in);
    System.out.print("Ingrese el título del material a prestar: ");
    String titulo = scanner.nextLine();

    for (Prestable material : materiales) {
        if (material.título.equals(titulo) && material.isDisponible()) {
            material.prestar();
            return;
        }
    }

    System.out.println("Material no encontrado o no disponible.");
}

private void devolverMaterial() {
    Scanner scanner = new Scanner(System.in);
    System.out.print("Ingrese el título del material a devolver: ");
    String titulo = scanner.nextLine();

    for (Prestable material : materiales) {
        if (material.título.equals(titulo) && !material.isDisponible()) {
            material.devolver();
            return;
        }
    }

    System.out.println("Material no encontrado o ya está disponible.");
}

public static void main(String[] args) {
    BibliotecaVirtual biblioteca = new BibliotecaVirtual();
```

```
        Libro libro = new Libro("La Sombra del Viento", "Carlos Ruiz Zafón", 2001, 500, "Editorial Planeta");
        Revista revista = new Revista("National Geographic", "Varios", 2023, 100, "ISSN1234");
        DVD dvd = new DVD("Pulp Fiction", "Quentin Tarantino", 1994, 154, true, 154);

        biblioteca.agregarMaterial(libro);
        biblioteca.agregarMaterial(revista);
        biblioteca.agregarMaterial(dvd);

        biblioteca.interactuarConMateriales();
    }
}
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

public class BibliotecaVirtual {
    private List<Prestable> materiales;

    public BibliotecaVirtual() {
        materiales = new ArrayList<>();
    }

    public void agregarMaterial(Prestable material) {
        materiales.add(material);
    }

    public void eliminarMaterial(Prestable material) {
        materiales.remove(material);
```

```
}

public void mostrarInformacionMateriales() {
    for (Prestable material : materiales) {
        material.mostrarInformación();
    }
}

public void interactuarConMateriales() {
    Scanner scanner = new Scanner(System.in);
    int opcion;

    do {
        System.out.println("Menú de opciones:");
        System.out.println("1. Prestamo");
        System.out.println("2. Devolución");
        System.out.println("3. Mostrar información de materiales");
        System.out.println("4. Salir");
        System.out.print("Seleccione una opción: ");
        opcion = scanner.nextInt();

        switch (opcion) {
            case 1:
                prestarMaterial();
                break;
            case 2:
                devolverMaterial();
                break;
            case 3:
                mostrarInformacionMateriales();
                break;
```

```
            case 4:
                System.out.println("Saliendo del programa.");
                break;
            default:
                System.out.println("Opción no válida. Intente nuevamente.");
        }
    } while (opcion != 4);
}

private void prestarMaterial() {
    Scanner scanner = new Scanner(System.in);
    System.out.print("Ingrese el título del material a prestar: ");
    String titulo = scanner.nextLine();

    for (Prestable material : materiales) {
        if (material.título.equals(titulo) && material.isDisponible()) {
            material.prestar();
            return;
        }
    }

    System.out.println("Material no encontrado o no disponible.");
}

private void devolverMaterial() {
    Scanner scanner = new Scanner(System.in);
    System.out.print("Ingrese el título del material a devolver: ");
    String titulo = scanner.nextLine();

    for (Prestable material : materiales) {
        if (material.título.equals(titulo) && !material.isDisponible()) {
```

```
                material.devolver();
                return;
            }
        }

        System.out.println("Material no encontrado o ya está disponible.");
    }

    public static void main(String[] args) {
        BibliotecaVirtual biblioteca = new BibliotecaVirtual();

        Libro libro = new Libro("La Sombra del Viento", "Carlos Ruiz Zafón", 2001, 500,
"Editorial Planeta");
        Revista revista = new Revista("National Geographic", "Varios", 2023, 100,
"ISSN1234");
        DVD dvd = new DVD("Pulp Fiction", "Quentin Tarantino", 1994, 154, true, 154);

        biblioteca.agregarMaterial(libro);
        biblioteca.agregarMaterial(revista);
        biblioteca.agregarMaterial(dvd);

        biblioteca.interactuarConMateriales();
    }
}
```