| **Asignatura** | **Programación II** |
| --- | --- |
| **Carrera** | **Ing. en Informática** |
| **Plan** | **Ajuste 2023** |
| **Ciclo** | **2do** |
| **Cuatrimestre** | **1ero** |
| **Tema/Título** | **Practica 2** |
| **Profesor** | **Adrian Tozzi** |

**Grupo de Trabajo**

| **ID/Matrícula** | **APELLIDO, Nombres** | **Correo Electrónico** |
| --- | --- | --- |
| **000-20-2323** | Cannizzaro, Pablo | Pablom.cannizzaro@comunidad.ub.edu.ar |
| **000-20-1394** | Callizaya, Leandro | [leandroa.callizaya@comunidad.ub.edu.ar](mailto:leandro.callizaya@comunidad.ub.edu.ar) |
| **000-20-1953** | Conde Buades, Joaquín | Joaquin.conde@comunidad.ub.edu.ar |
| **000-20-1212** | Franco Dalla Via Oliveros | Franco.dalla@comunidad.ub.edu.ar |

**Grilla de calificación**

| **Concepto** | **Propuesta** | **Marco Teórico** | **Desarrollo propio** | **Conclusiones** | **Fuentes y Referencias** |
| --- | --- | --- | --- | --- | --- |
| **Sobresaliente (10)** |  |  |  |  |  |
| **Distinguido (9-8)** |  |  |  |  |  |
| **Bueno (7-6)** |  |  |  |  |  |
| **Aprobado (5-4)** |  |  |  |  |  |
| **Insuficiente (3-2-1)** |  |  |  |  |  |
| **Reprobado (0)** |  |  |  |  |  |
| **NOTA** |  | | | | |

| Comentario adicional del Profesor: |
| --- |

**PRACTICA 2**

**EJERCICIO 1:**

Desarrollar una aplicación que nos permita almacenar información sobre películas (en DVD). La idea es crear un catálogo de todos los DVD que tenemos, hemos visto o deseamos tener o ver.

La funcionalidad que queremos que nos brinde el catálogo debe incluir como mínimo lo siguiente:

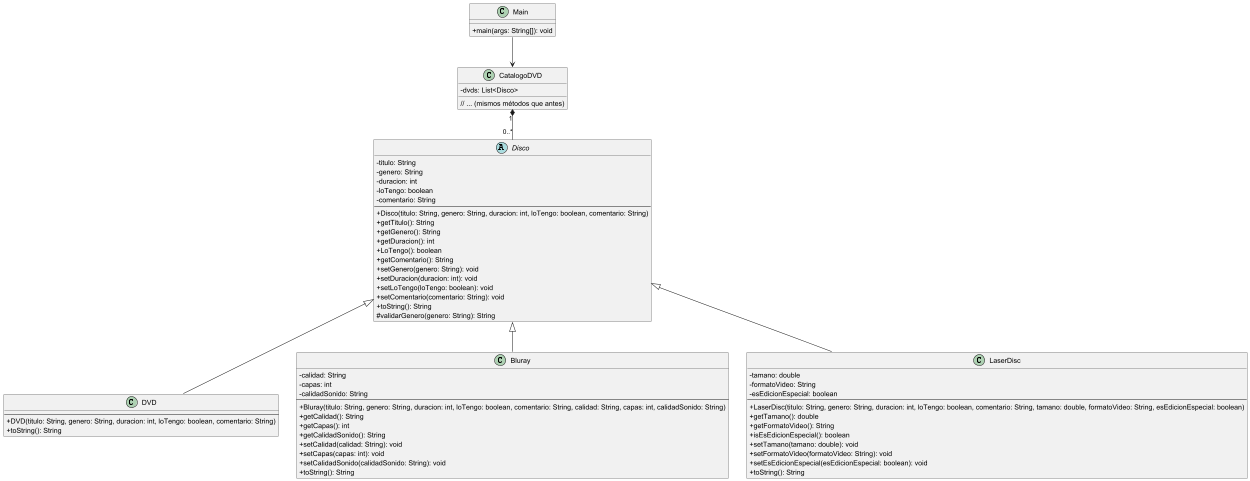
* Permitir ingresar información sobre los DVD.
* Permitir eliminar un DVD conociendo el título.
* Permitir modificar los valores de uno o más atributos conociendo el título.
* Listar:
  + todos los DVD
  + los DVD que tengo
  + los DVD que duran menos de un tiempo dado en minutos.
  + todos los DVD ordenados por título
* Informar:
  + la cantidad total de DVD.
  + la cantidad de DVD que tengo

Los detalles que queremos almacenar de cada DVD son:

* Título.
* Género.
* Tiempo de duración en minutos.
* Un atributo que indique si tenemos o no el DVD.
* Un comentario. (Excelente, Muy Buena, Buena, Regular, Mala)

Se pide:

1. Dibujar los diagramas de clase en UML.
2. Dibujar el Diagrama de Secuencia de 2 Métodos
3. Programar la clase DVD e ir probando cada método.
4. Programar la clase Catálogo.
5. Realizar pruebas positivas y pruebas negativas.
6. Desarrollar pruebas automatizadas (Deseable).
7. Documentar la aplicación usando javadoc.
8. Agregar a la aplicación anterior una interfaz (de texto o gráfica AWS o Swing) de usuario.
9. Respetar el Modelo MVC.



package P2\_1;

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.*in*);

CatalogoDVD catalogo = new CatalogoDVD();

while (true) {

try {

System.*out*.println("\n--- Menú de opciones ---");

System.*out*.println("1. Agregar DVD/BluRay/LaserDisc");

System.*out*.println("2. Eliminar DVD/BluRay/LaserDisc");

System.*out*.println("3. Modificar DVD/BluRay/LaserDisc");

System.*out*.println("4. Listar todos los DVDs/BluRays/LaserDisc");

System.*out*.println("5. Listar DVDs/BluRays/LaserDisc que tengo");

System.*out*.println("6. Listar DVDs/BluRays/LaserDisc por duración");

System.*out*.println("7. Listar DVDs/BluRays/LaserDisc ordenados por título");

System.*out*.println("8. Salir");

System.*out*.print("Seleccione una opción: ");

int opcion = Integer.*parseInt*(scanner.nextLine());

switch (opcion) {

case 1:

catalogo.crearDVDConValidaciones(scanner);

break;

case 2:

System.*out*.print("Ingrese el título del DVD a eliminar: ");

String tituloEliminar = scanner.nextLine();

if (catalogo.eliminarDVD(tituloEliminar)) {

System.*out*.println("DVD eliminado exitosamente.");

} else {

System.*out*.println("No se encontró el DVD con ese título.");

}

break;

case 3:

System.*out*.print("Ingrese el título del DVD a modificar: ");

String tituloModificar = scanner.nextLine();

if (catalogo.modificarDVDConValidacion(tituloModificar, scanner)) {

System.*out*.println("DVD modificado exitosamente.");

} else {

System.*out*.println("No se encontró el DVD con ese título.");

}

break;

case 4:

catalogo.listarTodos();

break;

case 5:

catalogo.listarTengo();

break;

case 6:

System.*out*.print("Ingrese la duración máxima (en minutos): ");

int maxDuracion = Integer.*parseInt*(scanner.nextLine());

catalogo.listarPorDuracion(maxDuracion);

break;

case 7:

catalogo.listarOrdenadoPorTitulo();

break;

case 8:

System.*out*.println("¡Hasta luego!");

scanner.close();

return;

default:

System.*out*.println("Opción no válida.");

}

} catch (Exception e) {

System.*out*.println("⚠️ Error: " + e.getMessage());

}

}

}

}

package P2\_1;

abstract class Disco {

private String titulo;

private String genero;

private int duracion; // en minutos

private boolean loTengo;

private String comentario;

public Disco(String titulo, String genero, int duracion, boolean loTengo, String comentario) {

this.titulo = titulo;

this.genero = genero;

this.duracion = duracion;

this.loTengo = loTengo;

this.comentario = comentario;

}

public String getTitulo() {

return titulo;

}

public void setTitulo(String titulo) {

this.titulo = titulo;

}

public String getGenero() {

return genero;

}

public void setGenero(String genero) {

this.genero = genero;

}

public int getDuracion() {

return duracion;

}

public void setDuracion(int duracion) {

this.duracion = duracion;

}

public boolean loTengo() {

return loTengo;

}

public void setLoTengo(boolean loTengo) {

this.loTengo = loTengo;

}

public String getComentario() {

return comentario;

}

public void setComentario(String comentario) {

if (comentario.length() > 100) {

this.comentario = comentario.substring(0, 100);

} else {

this.comentario = comentario;

}

}

@Override

public String toString() {

return String.*format*("Título: %s | Género: %s | Duración: %d min | Lo tengo: %s | Comentario: %s",

titulo, genero, duracion, loTengo ? "Sí" : "No", comentario);

}

}

package P2\_1;

import java.util.\*;

public class CatalogoDVD {

private final List<Disco> dvds;

public CatalogoDVD() {

dvds = new ArrayList<>();

}

public void crearDVDConValidaciones(Scanner scanner) {

try {

System.*out*.print("Tipo (D: DVD, B: BluRay, L: LaserDisc): ");

String tipo = scanner.nextLine().trim().toUpperCase();

if (!tipo.equals("D") && !tipo.equals("B") && !tipo.equals("L")) {

throw new IllegalArgumentException("Tipo inválido. Debe ser D, B o L.");

}

System.*out*.print("Título: ");

String titulo = scanner.nextLine().trim();

if (buscarPorTitulo(titulo) != null) {

System.*out*.println("⚠️ Ya existe un DVD con ese título.");

return;

}

System.*out*.print("Género (S)uspenso, (C)omedia o (T)error: ");

String genero = scanner.nextLine().trim().toUpperCase();

if (!genero.equals("S") && !genero.equals("C") && !genero.equals("T")) {

throw new IllegalArgumentException("Género inválido. Debe ser S, C o T.");

}

System.*out*.print("Duración (1-600): ");

int duracion = Integer.*parseInt*(scanner.nextLine());

if (duracion < 1 || duracion > 600) {

throw new IllegalArgumentException("La duración debe estar entre 1 y 600 minutos.");

}

System.*out*.print("¿Lo tienes? (s/n): ");

String tengo = scanner.nextLine().trim().toLowerCase();

boolean loTengo = tengo.equals("s");

System.*out*.print("Comentario (máx 100 caracteres): ");

String comentario = scanner.nextLine();

if (comentario.length() > 100) {

throw new IllegalArgumentException("El comentario no puede exceder los 100 caracteres.");

}

// Crear el DVD o tipo correspondiente

if (tipo.equals("B")) {

System.*out*.print("Calidad (HD, Full HD, 4K): ");

String calidad = scanner.nextLine().trim();

System.*out*.print("Capas (1-4): ");

int capas = Integer.*parseInt*(scanner.nextLine());

System.*out*.print("Calidad de sonido (Estéreo, Dolby, DTS): ");

String sonido = scanner.nextLine();

agregarDVD(new Bluray(titulo, genero, duracion, loTengo, comentario, calidad, capas, sonido));

} else if (tipo.equals("L")) {

System.*out*.print("Tamaño (pulgadas, usual 12): ");

double tamano = Double.*parseDouble*(scanner.nextLine());

System.*out*.print("Formato de video (NTSC/PAL/SECAM): ");

String formatoVideo = scanner.nextLine().trim();

System.*out*.print("¿Es edición especial? (s/n): ");

boolean especial = scanner.nextLine().trim().equalsIgnoreCase("s");

agregarDVD(new LaserDisc(titulo, genero, duracion, loTengo, comentario, tamano, formatoVideo, especial));

} else {

agregarDVD(new DVD(titulo, genero, duracion, loTengo, comentario));

}

} catch (NumberFormatException e) {

System.*out*.println("⚠️ Error: Entrada inválida para un número. Por favor, ingrese un valor válido.");

} catch (Exception e) {

System.*out*.println("⚠️ Error: " + e.getMessage());

}

}

public Disco buscarPorTitulo(String titulo) {

return dvds.stream()

.filter(d -> d.getTitulo().equalsIgnoreCase(titulo))

.findFirst()

.orElse(null);

}

public void agregarDVD(Disco disco) {

if (buscarPorTitulo(disco.getTitulo()) != null) {

throw new IllegalArgumentException("Ya existe un disco con ese título.");

}

dvds.add(disco);

}

public boolean eliminarDVD(String titulo) {

return dvds.removeIf(d -> d.getTitulo().equalsIgnoreCase(titulo));

}

public boolean modificarDVDConValidacion(String titulo, Scanner scanner) {

Disco d = buscarPorTitulo(titulo);

if (d == null) return false;

try {

System.*out*.print("Nuevo género (S)uspenso, (C)omedia o (T)error: ");

d.setGenero(scanner.nextLine());

System.*out*.print("Nueva duración (1-600): ");

d.setDuracion(Integer.*parseInt*(scanner.nextLine()));

System.*out*.print("¿Lo tienes? (s/n): ");

String loTengo = scanner.nextLine().trim().toLowerCase();

d.setLoTengo(loTengo.equals("s"));

System.*out*.print("Nuevo comentario (máx 100 caracteres): ");

d.setComentario(scanner.nextLine());

if (d instanceof Bluray bluray) {

System.*out*.print("Nueva calidad (HD, Full HD, 4K): ");

bluray.setCalidad(scanner.nextLine());

System.*out*.print("Capas (1-4): ");

bluray.setCapas(Integer.*parseInt*(scanner.nextLine()));

System.*out*.print("Calidad de sonido (Estéreo, Dolby, DTS): ");

bluray.setCalidadSonido(scanner.nextLine());

} else if (d instanceof LaserDisc ld) {

System.*out*.print("Nuevo tamaño (en pulgadas): ");

ld.setTamano(Double.*parseDouble*(scanner.nextLine()));

System.*out*.print("Nuevo formato de video (NTSC/PAL/SECAM): ");

ld.setFormatoVideo(scanner.nextLine());

System.*out*.print("¿Es edición especial? (s/n): ");

ld.setEsEdicionEspecial(scanner.nextLine().trim().equalsIgnoreCase("s"));

}

return true;

} catch (NumberFormatException e) {

System.*out*.println("⚠️ Error: Entrada inválida para un número.");

return false;

} catch (Exception e) {

System.*out*.println("⚠️ Error: " + e.getMessage());

return false;

}

}

public void listarTodos() {

dvds.forEach(System.*out*::println);

}

public void listarTengo() {

dvds.stream().filter(Disco::loTengo).forEach(System.*out*::println);

}

public void listarPorDuracion(int maxDuracion) {

dvds.stream().filter(d -> d.getDuracion() < maxDuracion).forEach(System.*out*::println);

}

public void listarOrdenadoPorTitulo() {

dvds.stream()

.sorted(Comparator.*comparing*(Disco::getTitulo)) // Usar Disco::getTitulo en lugar de DVD::getTitulo

.forEach(System.*out*::println);

}

public int cantidadTotal() {

return dvds.size();

}

public long cantidadTengo() {

return dvds.stream().filter(Disco::loTengo).count();

}

}

package P2\_1;

public class LaserDisc extends Disco {

private double tamano;

private String formatoVideo;

private boolean esEdicionEspecial;

public LaserDisc(String titulo, String genero, int duracion, boolean loTengo, String comentario,

double tamano, String formatoVideo, boolean esEdicionEspecial) {

super(titulo, genero, duracion, loTengo, comentario);

setTamano(tamano);

setFormatoVideo(formatoVideo);

this.esEdicionEspecial = esEdicionEspecial;

}

public double getTamano() { return tamano; }

public String getFormatoVideo() { return formatoVideo; }

public boolean isEsEdicionEspecial() { return esEdicionEspecial; }

public void setTamano(double tamano) {

if (tamano <= 0)

throw new IllegalArgumentException("El tamaño debe ser mayor que 0.");

this.tamano = tamano;

}

public void setFormatoVideo(String formatoVideo) {

if (!formatoVideo.equalsIgnoreCase("NTSC") &&

!formatoVideo.equalsIgnoreCase("PAL") &&

!formatoVideo.equalsIgnoreCase("SECAM"))

throw new IllegalArgumentException("Formato de video no válido (NTSC, PAL, SECAM).");

this.formatoVideo = formatoVideo;

}

public void setEsEdicionEspecial(boolean esEdicionEspecial) {

this.esEdicionEspecial = esEdicionEspecial;

}

@Override

public String toString() {

return "[LaserDisc] " + super.toString() +

" | Tamaño: " + tamano + " pulgadas" +

" | Formato video: " + formatoVideo +

" | Ed. Especial: " + (esEdicionEspecial ? "Sí" : "No");

}

}

package P2\_1;

public class DVD extends Disco {

public DVD(String titulo, String genero, int duracion, boolean loTengo, String comentario) {

super(titulo, genero, duracion, loTengo, comentario);

}

@Override

public String toString() {

return "[DVD] " + super.toString();

}

}

package P2\_1;

public class Bluray extends Disco {

private String calidad;

private int capas;

private String calidadSonido;

public Bluray(String titulo, String genero, int duracion, boolean loTengo, String comentario,

String calidad, int capas, String calidadSonido) {

super(titulo, genero, duracion, loTengo, comentario);

setCalidad(calidad);

setCapas(capas);

setCalidadSonido(calidadSonido);

}

public String getCalidad() { return calidad; }

public int getCapas() { return capas; }

public String getCalidadSonido() { return calidadSonido; }

public void setCalidad(String calidad) {

if (!calidad.equalsIgnoreCase("HD") &&

!calidad.equalsIgnoreCase("Full HD") &&

!calidad.equalsIgnoreCase("4K"))

throw new IllegalArgumentException("Calidad inválida. Debe ser HD, Full HD o 4K.");

this.calidad = calidad;

}

public void setCapas(int capas) {

if (capas < 1 || capas > 4)

throw new IllegalArgumentException("Capas debe estar entre 1 y 4.");

this.capas = capas;

}

public void setCalidadSonido(String calidadSonido) {

if (!calidadSonido.equalsIgnoreCase("Estéreo") &&

!calidadSonido.equalsIgnoreCase("Dolby") &&

!calidadSonido.equalsIgnoreCase("DTS"))

throw new IllegalArgumentException("Calidad de sonido inválida. Use Estéreo, Dolby o DTS.");

this.calidadSonido = calidadSonido;

}

@Override

public String toString() {

return "[BluRay] " + super.toString() + ", Calidad: " + calidad + ", Capas: " + capas +

", Sonido: " + calidadSonido;

}

}

**EJERCICIO 2:**

package Practica2;

import java.util.\*;

import java.time.LocalDate;

// Enumeraciones

enum TipoNovela {

*CIENCIA\_FICCION*, *ROMANCE*, *MISTERIO*, *JUVENIL*, *POLICIAL*

}

enum TipoTecnico {

*INGENIERIA*, *CIENCIAS\_NATURALES*, *CIENCIAS\_SOCIALES*

}

enum Formato {

*TAPAS\_DURAS*, *EDICION\_ECONOMICA*

}

// Clase Editorial

class Editorial {

private String nombre;

public Editorial(String nombre) {

this.nombre = nombre;

}

public String getNombre() {

return nombre;

}

}

// Clase Proveedor

class Proveedor {

private String nombre;

private List<Editorial> editoriales;

public Proveedor(String nombre) {

this.nombre = nombre;

this.editoriales = new ArrayList<>();

}

public void agregarEditorial(Editorial editorial) {

editoriales.add(editorial);

}

public boolean representaEditorial(Editorial editorial) {

return editoriales.contains(editorial);

}

public String getNombre() {

return nombre;

}

}

// Clase abstracta Libro

abstract class Libro {

protected String titulo;

protected List<String> autores;

protected Editorial editorial;

protected int añoEdicion;

protected Formato formato;

protected String isbn;

protected int stock;

public Libro(String titulo, List<String> autores, Editorial editorial, int añoEdicion, Formato formato, String isbn, int stock) {

this.titulo = titulo;

this.autores = autores;

this.editorial = editorial;

this.añoEdicion = añoEdicion;

this.formato = formato;

this.isbn = isbn;

this.stock = stock;

}

public void vender() {

if (stock > 0) {

stock--;

} else {

throw new IllegalStateException("No hay stock disponible");

}

}

public int verificarStock() {

return stock;

}

public String getIsbn() {

return isbn;

}

public Editorial getEditorial() {

return editorial;

}

public String getTitulo() {

return titulo;

}

public String toString() {

return titulo + " (" + isbn + ") - Stock: " + stock;

}

}

// Clase Novela

class Novela extends Libro {

private TipoNovela tipo;

public Novela(String titulo, List<String> autores, Editorial editorial, int añoEdicion, Formato formato, String isbn, int stock, TipoNovela tipo) {

super(titulo, autores, editorial, añoEdicion, formato, isbn, stock);

this.tipo = tipo;

}

}

// Clase LibroTecnico

class LibroTecnico extends Libro {

private TipoTecnico tipo;

public LibroTecnico(String titulo, List<String> autores, Editorial editorial, int añoEdicion, Formato formato, String isbn, int stock, TipoTecnico tipo) {

super(titulo, autores, editorial, añoEdicion, formato, isbn, stock);

this.tipo = tipo;

}

}

// Clase Cliente

class Cliente {

private String nombre;

private List<Encargo> encargos = new ArrayList<>();

public Cliente(String nombre) {

this.nombre = nombre;

}

public void agregarEncargo(Encargo encargo) {

encargos.add(encargo);

}

public String getNombre() {

return nombre;

}

public List<Encargo> getEncargos() {

return encargos;

}

}

// Clase Encargo

class Encargo {

private Libro libro;

private Cliente cliente;

private LocalDate fecha;

public Encargo(Libro libro, Cliente cliente) {

this.libro = libro;

this.cliente = cliente;

this.fecha = LocalDate.*now*();

}

public Libro getLibro() {

return libro;

}

public Cliente getCliente() {

return cliente;

}

public String toString() {

return "Encargo de '" + libro.getTitulo() + "' para " + cliente.getNombre() + " el " + fecha;

}

}

// Clase Libreria

class Libreria {

private List<Libro> libros = new ArrayList<>();

private List<Proveedor> proveedores = new ArrayList<>();

public void agregarLibro(Libro libro) {

libros.add(libro);

}

public void agregarProveedor(Proveedor proveedor) {

proveedores.add(proveedor);

}

public Libro buscarLibroPorTitulo(String titulo) {

for (Libro libro : libros) {

if (libro.getTitulo().equalsIgnoreCase(titulo)) {

return libro;

}

}

return null;

}

public Libro buscarLibroPorIsbn(String isbn) {

for (Libro libro : libros) {

if (libro.getIsbn().equals(isbn)) {

return libro;

}

}

return null;

}

public void venderLibro(String isbn) {

Libro libro = buscarLibroPorIsbn(isbn);

if (libro != null) {

try {

libro.vender();

System.*out*.println("Libro vendido correctamente: " + libro.getTitulo());

} catch (IllegalStateException e) {

System.*out*.println(e.getMessage());

}

} else {

System.*out*.println("Libro no encontrado.");

}

}

public void encargarLibro(String isbn, Cliente cliente) {

Libro libro = buscarLibroPorIsbn(isbn);

if (libro == null) {

System.*out*.println("Libro no encontrado.");

return;

}

if (libro.verificarStock() > 0) {

System.*out*.println("Hay stock disponible, no se requiere encargo.");

return;

}

for (Proveedor proveedor : proveedores) {

if (proveedor.representaEditorial(libro.getEditorial())) {

Encargo encargo = new Encargo(libro, cliente);

cliente.agregarEncargo(encargo);

System.*out*.println("Encargo realizado al proveedor: " + proveedor.getNombre());

return;

}

}

System.*out*.println("No se encontró proveedor para la editorial del libro.");

}

}

// Clase principal de ejemplo

public class Main {

public static void main(String[] args) {

Editorial ed1 = new Editorial("Planeta");

Editorial ed2 = new Editorial("Siglo XXI");

Proveedor proveedor1 = new Proveedor("Distribuidora Sur");

proveedor1.agregarEditorial(ed1);

Proveedor proveedor2 = new Proveedor("Distribuidora Norte");

proveedor2.agregarEditorial(ed2);

Libreria libreria = new Libreria();

libreria.agregarProveedor(proveedor1);

libreria.agregarProveedor(proveedor2);

Libro novela1 = new Novela("Viaje a Marte", Arrays.*asList*("J. Pérez"), ed1, 2021, Formato.*TAPAS\_DURAS*, "ISBN123", 2, TipoNovela.*CIENCIA\_FICCION*);

Libro tecnico1 = new LibroTecnico("Introducción a la Ingeniería", Arrays.*asList*("M. Gómez"), ed2, 2019, Formato.*EDICION\_ECONOMICA*, "ISBN456", 0, TipoTecnico.*INGENIERIA*);

libreria.agregarLibro(novela1);

libreria.agregarLibro(tecnico1);

Cliente cliente = new Cliente("Ana López");

libreria.venderLibro("ISBN123"); // Vender un libro con stock

libreria.venderLibro("ISBN123"); // Vender el último en stock

libreria.venderLibro("ISBN123"); // Ya no hay stock

libreria.encargarLibro("ISBN456", cliente); // Encargar libro sin stock

libreria.encargarLibro("ISBN123", cliente); // No requiere encargo

System.*out*.println("\nEncargos realizados por " + cliente.getNombre() + ":");

for (Encargo e : cliente.getEncargos()) {

System.*out*.println(e);

}

}

}

**Consola:**

Libro vendido correctamente: Viaje a Marte

Libro vendido correctamente: Viaje a Marte

No hay stock disponible

Encargo realizado al proveedor: Distribuidora Norte

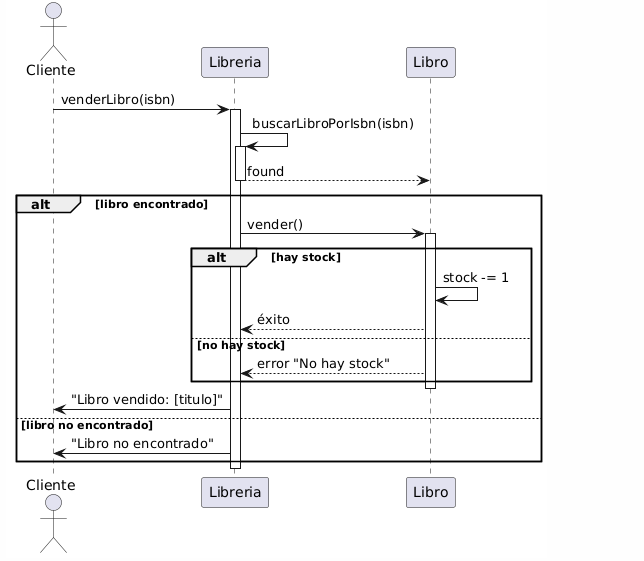
Encargo realizado al proveedor: Distribuidora Sur

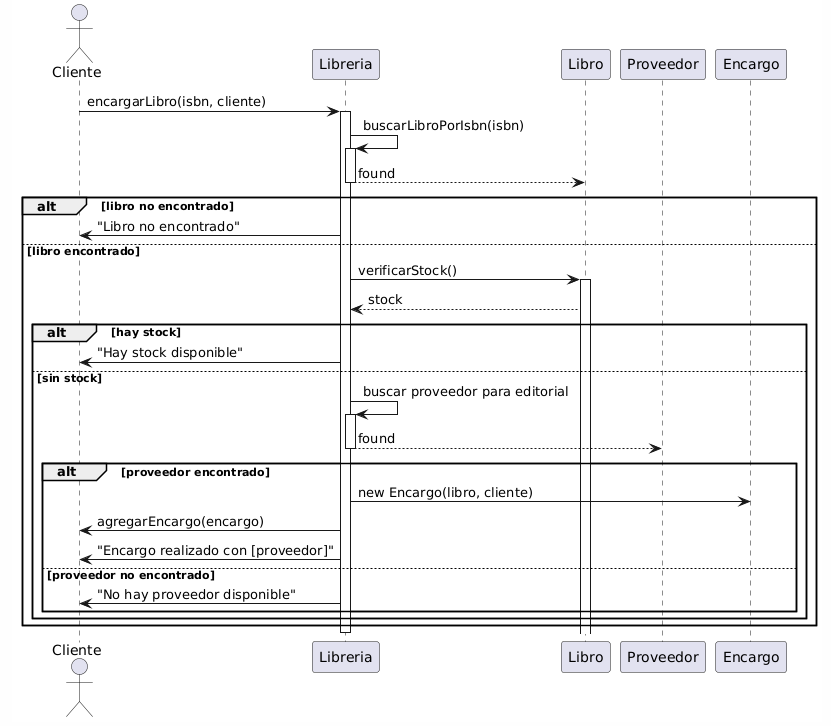
Encargos realizados por Ana López:

Encargo de 'Introducción a la Ingeniería' para Ana López el 2025-05-29

Encargo de 'Viaje a Marte' para Ana López el 2025-05-29







**Ejercicio 4:**

import java.util.ArrayList;

import java.util.Scanner;

public class CursoUniversitario {

private ArrayList<Estudiante> estudiantes;

public CursoUniversitario() {

this.estudiantes = new ArrayList<>();

}

public static void main(String[] args) {

CursoUniversitario curso = new CursoUniversitario();

Scanner scanner = new Scanner(System.*in*);

int opcion;

do {

System.*out*.println("\n--- MENÚ DE GESTIÓN DE CURSO ---");

System.*out*.println("1. Añadir estudiante");

System.*out*.println("2. Buscar estudiante");

System.*out*.println("3. Eliminar estudiante");

System.*out*.println("4. Cargar notas de estudiantes");

System.*out*.println("5. Calcular promedio del curso");

System.*out*.println("6. Mostrar estudiantes aprobados");

System.*out*.println("7. Salir");

System.*out*.print("Seleccione una opción: ");

opcion = scanner.nextInt();

scanner.nextLine(); // Limpiar buffer

switch (opcion) {

case 1:

curso.agregarEstudiante(scanner);

break;

case 2:

curso.buscarEstudiante(scanner);

break;

case 3:

curso.eliminarEstudiante(scanner);

break;

case 4:

curso.cargarNotas(scanner);

break;

case 5:

curso.calcularPromedioCurso();

break;

case 6:

curso.mostrarAprobados();

break;

case 7:

System.*out*.println("Saliendo del sistema...");

break;

default:

System.*out*.println("Opción no válida. Intente nuevamente.");

}

} while (opcion != 7);

scanner.close();

}

// Clase interna para representar a un estudiante

private static class Estudiante {

private String nombre;

private String apellidos;

private String codigo;

private int semestre;

private double notaFinal;

public Estudiante(String nombre, String apellidos, String codigo, int semestre) {

this.nombre = nombre;

this.apellidos = apellidos;

this.codigo = codigo;

this.semestre = semestre;

this.notaFinal = 0.0;

}

// Getters y setters

public String getCodigo() {

return codigo;

}

public double getNotaFinal() {

return notaFinal;

}

public void setNotaFinal(double notaFinal) {

this.notaFinal = notaFinal;

}

@Override

public String toString() {

return String.*format*("Código: %s\nNombre: %s %s\nSemestre: %d\nNota Final: %.2f",

codigo, nombre, apellidos, semestre, notaFinal);

}

}

// Metodo para agregar un nuevo estudiante

public void agregarEstudiante(Scanner scanner) {

System.*out*.println("\n--- AGREGAR NUEVO ESTUDIANTE ---");

System.*out*.print("Ingrese el código del estudiante: ");

String codigo = scanner.nextLine();

// Verificar si el código ya existe

if (buscarPorCodigo(codigo) != null) {

System.*out*.println("Error: Ya existe un estudiante con este código.");

return;

}

System.*out*.print("Ingrese el nombre del estudiante: ");

String nombre = scanner.nextLine();

System.*out*.print("Ingrese los apellidos del estudiante: ");

String apellidos = scanner.nextLine();

System.*out*.print("Ingrese el semestre del estudiante: ");

int semestre = scanner.nextInt();

scanner.nextLine(); // Limpiar buffer

Estudiante nuevoEstudiante = new Estudiante(nombre, apellidos, codigo, semestre);

estudiantes.add(nuevoEstudiante);

System.*out*.println("Estudiante agregado exitosamente!");

}

// Metodo para buscar un estudiante por código

public void buscarEstudiante(Scanner scanner) {

System.*out*.println("\n--- BUSCAR ESTUDIANTE ---");

System.*out*.print("Ingrese el código del estudiante a buscar: ");

String codigo = scanner.nextLine();

Estudiante encontrado = buscarPorCodigo(codigo);

if (encontrado != null) {

System.*out*.println("\nESTUDIANTE ENCONTRADO:");

System.*out*.println(encontrado);

} else {

System.*out*.println("No se encontró ningún estudiante con ese código.");

}

}

// Metodo para eliminar un estudiante

public void eliminarEstudiante(Scanner scanner) {

System.*out*.println("\n--- ELIMINAR ESTUDIANTE ---");

System.*out*.print("Ingrese el código del estudiante a eliminar: ");

String codigo = scanner.nextLine();

Estudiante encontrado = buscarPorCodigo(codigo);

if (encontrado != null) {

System.*out*.println("\nESTUDIANTE A ELIMINAR:");

System.*out*.println(encontrado);

System.*out*.print("\n¿Está seguro que desea eliminar este estudiante? (S/N): ");

String confirmacion = scanner.nextLine();

if (confirmacion.equalsIgnoreCase("S")) {

estudiantes.remove(encontrado);

System.*out*.println("Estudiante eliminado exitosamente.");

} else {

System.*out*.println("Operación cancelada.");

}

} else {

System.*out*.println("No se encontró ningún estudiante con ese código.");

}

}

// Metodo para cargar notas de estudiantes

public void cargarNotas(Scanner scanner) {

System.*out*.println("\n--- CARGAR NOTAS DE ESTUDIANTES ---");

if (estudiantes.isEmpty()) {

System.*out*.println("No hay estudiantes registrados en el curso.");

return;

}

System.*out*.println("Lista de estudiantes:");

for (Estudiante est : estudiantes) {

System.*out*.println(est.getCodigo() + " - " + est.nombre + " " + est.apellidos);

}

System.*out*.print("\nIngrese el código del estudiante a calificar: ");

String codigo = scanner.nextLine();

Estudiante encontrado = buscarPorCodigo(codigo);

if (encontrado != null) {

System.*out*.print("Ingrese la nota final del estudiante (0.0 - 5.0): ");

double nota = scanner.nextDouble();

scanner.nextLine(); // Limpiar buffer

if (nota >= 0.0 && nota <= 5.0) {

encontrado.setNotaFinal(nota);

System.*out*.println("Nota asignada correctamente.");

} else {

System.*out*.println("Error: La nota debe estar entre 0.0 y 5.0");

}

} else {

System.*out*.println("No se encontró ningún estudiante con ese código.");

}

}

// Metodo para calcular el promedio del curso

public void calcularPromedioCurso() {

if (estudiantes.isEmpty()) {

System.*out*.println("\nNo hay estudiantes registrados en el curso.");

return;

}

double sumaNotas = 0.0;

int contador = 0;

for (Estudiante est : estudiantes) {

if (est.getNotaFinal() > 0.0) { // Solo contar estudiantes con nota asignada

sumaNotas += est.getNotaFinal();

contador++;

}

}

if (contador == 0) {

System.*out*.println("\nNo hay notas asignadas para calcular el promedio.");

} else {

double promedio = sumaNotas / contador;

System.*out*.printf("\nPromedio del curso: %.2f (basado en %d estudiantes)%n", promedio, contador);

}

}

// Metodo para mostrar estudiantes aprobados

public void mostrarAprobados() {

if (estudiantes.isEmpty()) {

System.*out*.println("\nNo hay estudiantes registrados en el curso.");

return;

}

int aprobados = 0;

int totalConNota = 0;

for (Estudiante est : estudiantes) {

if (est.getNotaFinal() > 0.0) { // Solo contar estudiantes con nota asignada

totalConNota++;

if (est.getNotaFinal() >= 4.0) {

aprobados++;

}

}

}

if (totalConNota == 0) {

System.*out*.println("\nNo hay notas asignadas para calcular aprobados.");

} else {

double porcentaje = (double) aprobados / totalConNota \* 100;

System.*out*.printf("\nEstudiantes aprobados: %d de %d (%.2f%%)%n",

aprobados, totalConNota, porcentaje);

}

}

// Metodo auxiliar para buscar estudiante por código

private Estudiante buscarPorCodigo(String codigo) {

for (Estudiante est : estudiantes) {

if (est.getCodigo().equalsIgnoreCase(codigo)) {

return est;

}

}

return null;

}

}