EJERCICIOS JAVA-POO

Crear Clases y Objetos

1. Definir una clase "Libro" con atributos para el título, el autor y el año de publicación. Incluye un constructor para inicializar estos atributos y un método para mostrar los detalles del libro.

2. Definir una clase "CuentaBancaria" con atributos para el número de cuenta, el titular y el saldo. Incluye métodos para depositar, retirar dinero y mostrar el saldo actual.

Herencia y Polimorfismo

3. Definir una clase "Empleado" con atributos para el nombre y el salario. Crea una subclase "Gerente" que agrega un atributo para el departamento y sobrescribe un método para mostrar los detalles del empleado.

4. Definir una clase abstracta "Vehiculo" con un método abstracto "mover". Crea subclases "Coche" y "Bicicleta" que implementen el método "mover" de diferentes maneras.

Encapsulamiento y Abstracción

5. Modificar la clase "CuentaBancaria" para que todos los atributos sean privados. Proporciona métodos getter y setter para acceder y modificar los atributos.

6. Crear una interfaz "OperacionesMatematicas" con métodos para sumar, restar, multiplicar y dividir. Implementa esta interfaz en una clase "Calculadora".

POO Y ESTRUCTURAS DE DATOS

Ejercicio 1: Implementación de una Lista Enlazada

Implementa una lista enlazada simple en Java. Debe tener métodos para agregar, eliminar y buscar elementos.

1. Crear una clase `Nodo` que represente un nodo en la lista.

2. Crear una clase `ListaEnlazada` que contenga:

- Un método `agregar(T elemento)` para agregar un elemento al final de la lista.

- Un método `eliminar(T elemento)` para eliminar el primer elemento que coincida con el valor proporcionado.

- Un método `contiene(T elemento)` que devuelva `true` si el elemento está en la lista, de lo contrario `false`.

Ejercicio 2: Gestión de una Biblioteca

Diseña un sistema para gestionar una biblioteca. Debes utilizar clases para representar libros y miembros de la biblioteca, y una estructura de datos para gestionar el inventario y los préstamos.

1. Crear una clase `Libro` con atributos como `titulo`, `autor`, `ISBN`, y `disponible`.

2. Crear una clase `Miembro` con atributos como `nombre`, `idMiembro`, y una lista de libros prestados.

3. Crear una clase `Biblioteca` que contenga:

- Un método `agregarLibro(Libro libro)` para agregar un libro al inventario.

- Un método `prestarLibro(String ISBN, Miembro miembro)` que permita a un miembro pedir prestado un libro si está disponible.

- Un método `devolverLibro(String ISBN, Miembro miembro)` para devolver un libro prestado.

- Un método `listarLibrosDisponibles()` que devuelva una lista de libros disponibles.

Ejercicio 3: Pila (Stack) Personalizada

Implementa una estructura de datos tipo pila (stack) utilizando un array.

1. Crear una clase `PilaPersonalizada` que contenga:

- Un método `apilar(T elemento)` para agregar un elemento a la pila.

- Un método `desapilar()` para remover y devolver el elemento en la cima de la pila.

- Un método `cima()` que devuelva el elemento en la cima sin removerlo.

- Un método `estaVacia()` que devuelva `true` si la pila está vacía.

- Un método `tamano()` que devuelva el número de elementos en la pila.

Ejercicio 4: Sistema de Gestión de Tareas

Crea un sistema para gestionar tareas. Utiliza una cola de prioridad para manejar las tareas según su urgencia.

1. Crear una clase `Tarea` con atributos como `descripcion`, `fechaLimite`, y `prioridad`.

2. Crear una clase `GestorTareas` que utilice una cola de prioridad (PriorityQueue) para gestionar las tareas.

- Un método `agregarTarea(Tarea tarea)` para agregar una tarea.

- Un método `eliminarTarea(Tarea tarea)` para eliminar una tarea.

- Un método `obtenerSiguienteTarea()` que devuelva la siguiente tarea a realizar según la prioridad.

Ejercicio 5: Sistema de Calificaciones

Crea un sistema para gestionar las calificaciones de los estudiantes.

1. Crear una clase `Estudiante` con atributos como `nombre`, `idEstudiante`, y una lista de calificaciones.

2. Crear una clase `Curso` que contenga:

- Un método `agregarEstudiante(Estudiante estudiante)` para agregar un estudiante al curso.

- Un método `agregarCalificacion(String idEstudiante, double calificacion)` para agregar una calificación a un estudiante.

- Un método `obtenerPromedio(String idEstudiante)` que devuelva el promedio de calificaciones de un estudiante.

- Un método `listarEstudiantes()` que devuelva una lista de los estudiantes en el curso.