Guía de ejercicios

ANTES DE EMPEZAR...

Recuerda que en cada uno de los ejercicios necesito una demostración de que los entendiste. Para eso, en cada uno necesito que me especifiques dónde se cumple:

- Comportamiento
- Implementación
- Herencia
- Polimorfismo

Y justificar por qué se está cumpliendo ese concepto.

Ahora sí. ¡A programar!

Ejercicio 1:

Se desea diseñar un programa que simule un reproductor de música con distintas fuentes.

El sistema debe:

1. Definir una interfaz ReproductorMusica con los métodos:
    - reproducir()
    - pausar()
    - detener()
2. Implementar dos clases concretas:
    - ReproductorMP3: reproduce archivos locales en formato MP3.
    - ReproductorStreaming: reproduce música desde la nube (streaming).
3. En la clase principal (Main), el programa debe mostrar un menú en consola que permita al usuario:
    - Seleccionar el tipo de reproductor (MP3 o Streaming).
    - Elegir qué acción realizar: reproducir, pausar o detener.
    - Poder ejecutar varias acciones hasta que el usuario decida salir del programa.

Ejemplo de ejecución esperada:

Seleccione el tipo de reproductor:

1. MP3

2. Streaming

Opción: 1

Acciones disponibles:

1. Reproducir

2. Pausar

3. Detener

4. Salir

Opción: 1

Reproduciendo MP3...

Opción: 2

Pausando reproducción

Opción: 4

Saliendo del programa...

Ejercicio 2:

Se desea construir un sistema de pagos flexible que permita a una tienda aceptar múltiples métodos de pago.

Actualmente existen 3 formas de pago:

- Pago en efectivo
- Pago con tarjeta de crédito
- Pago por transferencia bancaria

El sistema debe:

1. Definir una interfaz Pago con los métodos:
    - realizarPago(double monto)
    - obtenerComprobante()
2. Implementar las clases:
    - PagoEfectivo
    - PagoTarjetaCredito

- PagoTransferencia

3. Crear una clase HistorialPagos que guarde todos los pagos realizados, de modo que el usuario pueda:
    - Registrar un nuevo pago.
    - Consultar los comprobantes de todos los pagos efectuados.
    - Calcular el total recaudado.
4. En la clase principal (Main), mostrar un menú en consola que permita al usuario:
    - Seleccionar el tipo de pago (efectivo, tarjeta o transferencia).
    - Ingresar el monto y, si corresponde, los datos adicionales (número de tarjeta o cuenta).
    - Guardar el pago en el historial.
    - Consultar en cualquier momento el listado de comprobantes y el total recaudado.
    - Salir del programa.

Ejemplo de ejecución esperada:

=== Sistema de Pagos ===

1. Pagar en efectivo

2. Pagar con tarjeta de crédito

3. Pagar por transferencia bancaria

4. Ver historial de pagos

5. Ver total recaudado

6. Salir

Opción: 1

Ingrese monto: 1500

Pago en efectivo realizado por $1500

Comprobante de pago en efectivo: $1500

Opción: 2

Ingrese monto: 2000

Ingrese número de tarjeta: 1234567890123456

Pago con tarjeta de crédito realizado por $2000 (Tarjeta terminada en 3456)

Comprobante de pago con tarjeta: $2000 (Tarjeta ****3456)

Opción: 4

--- Historial de pagos ---

Comprobante de pago en efectivo: $1500

Comprobante de pago con tarjeta: $2000 (Tarjeta ****3456)

Opción: 5

Total recaudado: $3500

Opción: 6

Saliendo del sistema...

Ejercicio 3:

Se desea diseñar un sistema para una empresa de desarrollo de software que permita gestionar a sus empleados.

Actualmente existen dos tipos de empleados:

- **Programadores**, que trabajan desarrollando en un lenguaje de programación.
- **Diseñadores**, que trabajan utilizando una herramienta de diseño.

El sistema debe cumplir con los siguientes requisitos:

1. Definir una clase abstracta Empleado que contenga:
    - Atributos básicos: nombre, salario.
    - Métodos:
        - trabajar() (abstracto, implementado por cada subclase).
        - calcularBono() que retorne un adicional al salario según el tipo de empleado.
2. Implementar las subclases:
    - Programador, con el atributo adicional lenguaje.
        - Su bono es el 20% del salario.

- Diseñador, con el atributo adicional herramienta.
    - Su bono es el 15% del salario.

3. Crear una clase Empresa que contenga una lista de empleados y permita:
    - Agregar empleados a la nómina.
    - Listar empleados, mostrando nombre, salario y tipo.
    - Calcular el gasto total (suma de salarios + bonos).
4. En la clase principal (Main), implementar un menú interactivo en consola para que el usuario pueda:
    - Agregar un programador o diseñador.
    - Mostrar todos los empleados con sus roles.
    - Calcular el gasto total de la empresa.
    - Salir del programa.

Ejemplo de ejecución esperada:

=== Sistema de Gestión de Empleados ===

1. Agregar Programador

2. Agregar Diseñador

3. Mostrar Empleados

4. Calcular gasto total

5. Salir

Opción: 1

Ingrese nombre: Ana

Ingrese salario: 2000

Ingrese lenguaje: Java

Empleado agregado correctamente.

Opción: 2

Ingrese nombre: Pedro

Ingrese salario: 1800

Ingrese herramienta: Photoshop

Empleado agregado correctamente.

Opción: 3

--- Lista de empleados ---

Ana - Programador - Salario: $2000 - Lenguaje: Java

Pedro - Diseñador - Salario: $1800 - Herramienta: Photoshop

Opción: 4

El gasto total de la empresa (con bonos) es: $4370

Opción: 5

Saliendo del sistema...

Ejercicio 4

Se desea diseñar un sistema que permita gestionar distintas figuras geométricas y calcular sus propiedades principales.

Actualmente existen tres tipos de figuras:

- **Círculo**, definido por su radio.
- **Rectángulo**, definido por su base y altura.
- **Triángulo isósceles**, definido por su base y altura.

El sistema debe cumplir los siguientes requisitos:

1. La interfaz Figura debe definir dos métodos:
    - double area() → devuelve el área de la figura.
    - double perimetro() → devuelve el perímetro de la figura.
2. Cada clase (Circulo, Rectangulo, Triangulo) debe implementar la interfaz y definir sus fórmulas de área y perímetro.
3. Crear una clase GestorFiguras que contenga una lista de figuras y permita:
    - Agregar nuevas figuras.
    - Mostrar todas las figuras indicando su tipo, área y perímetro.
    - Calcular el área total ocupada por todas las figuras.

4. En la clase principal (MainFiguras), implementar un **menú en consola** para que el usuario pueda:
    - Agregar un círculo (pidiendo su radio).
    - Agregar un rectángulo (pidiendo base y altura).
    - Agregar un triángulo (pidiendo base y altura).
    - Mostrar todas las figuras registradas.
    - Calcular el área total ocupada.
    - Salir del programa.

Ejemplo de ejecución esperada:

=== Sistema de Figuras Geométricas ===

1. Agregar círculo

2. Agregar rectángulo

3. Agregar triángulo

4. Mostrar figuras

5. Calcular área total

6. Salir

Opción: 1

Ingrese radio: 5

Círculo agregado correctamente.

Opción: 2

Ingrese base: 4

Ingrese altura: 6

Rectángulo agregado correctamente.

Opción: 3

Ingrese base: 3

Ingrese altura: 8

Triángulo agregado correctamente.

Opción: 4

--- Figuras registradas ---

Círculo → Área: 78.54 | Perímetro: 31.42

Rectángulo → Área: 24.0 | Perímetro: 20.0

Triángulo → Área: 12.0 | Perímetro: 17.56

Opción: 5

Área total ocupada por todas las figuras: 114.54

Opción: 6

Saliendo del sistema...

Ejercicio 5:

Una empresa de transporte necesita un sistema que permita crear vehículos de distintos tipos y simular sus acciones.

El sistema debe cumplir con los siguientes requisitos:

1. Definir una interfaz Vehiculo con los métodos:
    - acelerar()
    - frenar()
2. Implementar las clases concretas:
    - **Auto**: imprime mensajes propios al acelerar y frenar.
    - **Moto**: imprime mensajes propios al acelerar y frenar.
    - **Camion** (nuevo): imprime mensajes propios al acelerar y frenar.
3. Implementar una clase VehiculoFactory que se encargue de crear vehículos a partir de un tipo ingresado por el usuario.
4. Crear una clase **GestorVehiculos** que contenga una lista de vehículos y permita:
    - Agregar vehículos creados por la fábrica.
    - Mostrar todos los vehículos registrados indicando su tipo.
    - Simular que todos los vehículos aceleran y frenan en conjunto.

5. En la clase principal (MainFactory), implementar un menú en consola que permita al usuario:
    - Crear un vehículo (auto, moto o camión).
    - Mostrar los vehículos registrados.
    - Hacer que todos los vehículos aceleren.
    - Hacer que todos los vehículos frenen.
    - Salir del programa.

Ejemplo de ejecución esperada:

=== Sistema de Vehículos ===

1. Crear vehículo

2. Mostrar vehículos

3. Acelerar todos

4. Frenar todos

5. Salir

Opción: 1

Ingrese tipo de vehículo (auto/moto/camion): auto

Vehículo creado: Auto

Opción: 1

Ingrese tipo de vehículo (auto/moto/camion): moto

Vehículo creado: Moto

Opción: 2

--- Vehículos registrados ---

Auto

Moto

Opción: 3

El auto acelera

La moto acelera

Opción: 4

El auto frena

La moto frena

Opción: 5

Saliendo del sistema...

Ejercicio 6:

Una empresa de transporte necesita un sistema que permita crear vehículos de distintos tipos y simular sus acciones.

El sistema debe cumplir con los siguientes requisitos:

1. Definir una interfaz Vehiculo con los métodos:
    - acelerar()
    - frenar()
2. Implementar las clases concretas:
    - Auto: imprime mensajes propios al acelerar y frenar.
    - Moto: imprime mensajes propios al acelerar y frenar.
    - Camion (nuevo): imprime mensajes propios al acelerar y frenar.
3. Implementar una clase VehiculoFactory que se encargue de crear vehículos a partir de un tipo ingresado por el usuario.
4. Crear una clase GestorVehiculos que contenga una lista de vehículos y permita:
    - Agregar vehículos creados por la fábrica.
    - Mostrar todos los vehículos registrados indicando su tipo.
    - Simular que todos los vehículos aceleran y frenan en conjunto.
5. En la clase principal (MainFactory), implementar un menú en consola que permita al usuario:
    - Crear un vehículo (auto, moto o camión).
    - Mostrar los vehículos registrados.
    - Hacer que todos los vehículos aceleren.

- Hacer que todos los vehículos frenen.
- Salir del programa.

Ejemplo de ejecución esperada:

=== Sistema de Vehículos ===

1. Crear vehículo

2. Mostrar vehículos

3. Acelerar todos

4. Frenar todos

5. Salir

Opción: 1

Ingrese tipo de vehículo (auto/moto/camion): auto

Vehículo creado: Auto

Opción: 1

Ingrese tipo de vehículo (auto/moto/camion): moto

Vehículo creado: Moto

Opción: 2

--- Vehículos registrados ---

Auto

Moto

Opción: 3

El auto acelera

La moto acelera

Opción: 4

El auto frena

La moto frena

Opción: 5

Saliendo del sistema...

Ejercicio 7:

Se desea implementar un sistema interactivo de biblioteca que permita gestionar libros y sus préstamos.

Requisitos del sistema:

1. Almacenamiento de libros:
    - Los libros deben estar precargados (mínimo 3), por ejemplo:
        - "El Quijote"
        - "Cien años de soledad"
        - "La Odisea"
    - Los libros deben almacenarse en una estructura Map, donde:
        - La **clave** sea el título del libro en minúsculas.
        - El **valor** sea el objeto Libro.
2. Clase Libro:
    - Debe tener atributos: titulo y prestado (booleano).
    - Debe tener métodos para:
        - Prestar el libro (prestar()).
        - Devolver el libro (devolver()).
        - Consultar si está prestado (isPrestado()).
3. Clase Biblioteca:
    - Debe permitir:
        - Agregar libros al Map.
        - Mostrar todos los libros con su estado (prestado o disponible).
        - Prestar un libro: si está disponible, marcarlo como prestado; si no, mostrar un mensaje de error.
        - Devolver un libro: si estaba prestado, marcarlo como disponible; si no, mostrar un mensaje de error.
4. Interacción con el usuario (consola):

- El usuario podrá elegir entre:

1. Mostrar libros.
2. Prestar un libro.
3. Devolver un libro.
4. Salir del sistema.

- Cada acción debe mostrar mensajes claros indicando si se pudo realizar o si hubo un error.

Ejemplo de ejecución esperada:

--- Biblioteca ---

1. Mostrar libros

2. Prestar libro

3. Devolver libro

4. Salir

Opción: 1

El Quijote (Disponible)

Cien años de soledad (Disponible)

La Odisea (Disponible)

Opción: 2

Ingrese el título del libro a prestar: El Quijote

Prestado: El Quijote

Opción: 1

El Quijote (Prestado)

Cien años de soledad (Disponible)

La Odisea (Disponible)

Opción: 2

Ingrese el título del libro a prestar: El Quijote

No disponible: El Quijote

Opción: 3

Ingrese el título del libro a devolver: El Quijote

Devuelto: El Quijote

Opción: 1

El Quijote (Disponible)

Cien años de soledad (Disponible)

La Odisea (Disponible)

Opción: 4

Saliendo del sistema...