CICLO FORMATIVO DE GRADO SUPERIOR DESARROLLO DE APLICACIONES WEB *UT6 – POO AVANZADA*	EFA MORATALAZ Profesor: **DGC**	Asignatura: **PROGRAMACIÓN** Fecha:	**Nota**:
Alumno: _____			

Ejercicio 1. Tienda de libros

Se desea modelar una aplicación para una tienda de libros que tiene distintos tipos de productos, como libros físicos y libros digitales. Además, cada producto tiene un precio y un autor.

Cree una clase "Producto" que tenga los siguientes atributos:

- precio (double)
- autor (String)

Cree dos subclases de "Producto" llamadas "LibroFisico" y "LibroDigital". La clase "LibroFisico" debe tener un atributo extra para el peso en gramos y la clase "LibroDigital" debe tener un atributo extra para el formato de archivo.

Cree una clase "TiendaDeLibros" que tenga un atributo de lista de productos. La clase "TiendaDeLibros" debe tener un método para agregar productos a la lista y un método para calcular el precio total de todos los productos en la lista.

En el método principal de la aplicación, cree una instancia de la clase "TiendaDeLibros" y agregue algunos productos de ambas subclases. Luego, muestre el precio total de todos los productos en la lista.

Ejercicio 2. Empresa de transporte

Se desea modelar una aplicación para una empresa de transporte que tiene distintos tipos de vehículos, como camiones, buses y autos. Cada vehículo tiene un identificador único, una marca, un modelo, un año de fabricación y una lista de componentes.

Además, cada vehículo tiene un tipo de combustible y un consumo de combustible por kilómetro recorrido.

Cree dos subclases de "Vehiculo" llamadas "Camion" y "Bus". La clase "Camion" debe tener un atributo extra para el peso en toneladas y la clase "Bus" debe tener un atributo extra para la cantidad de asientos.

Cree una tercera subclase de "Vehiculo" llamada "Auto" que tenga un atributo extra para la cantidad de puertas y la clase "Auto" debe tener una lista de componentes diferente a la de las otras subclases.

Cree una clase "EmpresaTransporte" que tenga un atributo de lista de vehículos. La clase "EmpresaTransporte" debe tener un método para agregar vehículos a la lista y un método para calcular el consumo total de combustible de todos los vehículos en la lista.

En el método principal de la aplicación, cree una instancia de la clase "EmpresaTransporte" y agregue algunos vehículos de las subclases "Camion", "Bus" y "Auto". Luego, muestre el consumo total de combustible de todos los vehículos en la lista.

Además, cree un método en la clase "EmpresaTransporte" que permita filtrar los vehículos por año de fabricación y mostrar solo los vehículos que cumplan con el criterio de filtrado. Este método debe ser genérico y aceptar cualquier subclase de "Vehiculo". Finalmente, pruebe el método de filtrado con algunos vehículos de las subclases.

Ejercicio 3. Seguros

Se desea modelar una aplicación para una compañía de seguros que tiene distintos tipos de pólizas, como pólizas de vida, pólizas de auto y pólizas de hogar. Cada póliza tiene un número de póliza único, una fecha de inicio, una fecha de finalización y un valor asegurado. Además, cada póliza tiene una prima anual que se basa en el valor asegurado y la fecha de inicio de la póliza.

Cree dos subclases de "Poliza" llamadas "PolizaVida" y "PolizaAuto". La clase "PolizaVida" debe tener un atributo extra para el beneficiario de la póliza y la clase "PolizaAuto" debe tener un atributo extra para la marca y modelo del automóvil asegurado.

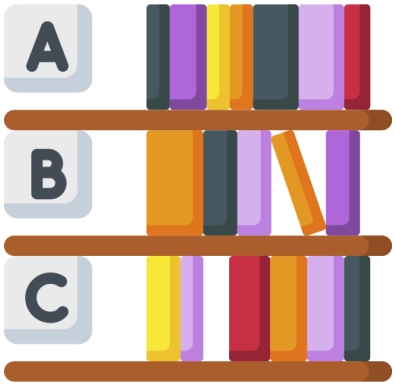
Cree una tercera subclase de "Poliza" llamada "PolizaHogar" que tenga un atributo extra para la dirección del hogar asegurado y una lista de riesgos cubiertos.

Cree una clase "CompaniaSeguros" que tenga un atributo de lista de pólizas. La clase "CompaniaSeguros" debe tener un método para agregar pólizas a la lista y un método para calcular la prima total de todas las pólizas en la lista.

En el método principal de la aplicación, cree una instancia de la clase "CompaniaSeguros" y agregue algunos tipos de pólizas de las subclases "PolizaVida", "PolizaAuto" y "PolizaHogar". Luego, muestre la prima total de todas las pólizas en la lista.

Además, cree un método en la clase "CompaniaSeguros" que permita filtrar las pólizas por fecha de inicio y mostrar solo las pólizas que cumplan con el criterio de filtrado. Este método debe ser genérico y aceptar cualquier subclase de "Poliza". Finalmente, pruebe el método de filtrado con algunas pólizas de las subclases.

Ejercicio 4. Biblioteca



Se desea modelar una aplicación para una biblioteca que tiene distintos tipos de libros y una lista de préstamos de libros. Cada libro tiene un identificador único, un título, un autor, una fecha de publicación y un número de páginas. Además, cada préstamo tiene un libro prestado, un usuario que lo ha tomado prestado y una fecha de devolución.

Cree una clase "Libro" que tenga los siguientes atributos:

- id (int)
- titulo (String)
- autor (String)
- fechaPublicacion (LocalDate)
- numPaginas (int)

Cree una clase "Prestamo" que tenga los siguientes atributos:

- libroPrestado (Libro)
- usuario (String)
- fechaDevolucion (LocalDate)

Cree una clase "Biblioteca" que tenga un atributo de lista de libros y un atributo de lista de préstamos. La clase "Biblioteca" debe tener un método para agregar libros y un método para agregar préstamos. Además, debe tener un método que permita consultar la lista de préstamos de un libro dado.

Cree una clase "Main" que tenga un método principal que permita agregar algunos libros y préstamos a la biblioteca y luego consultar la lista de préstamos de algunos libros.

Además, cree una clase "LibroDeReferencia" que es una subclase de "Libro" y tiene un atributo extra para el tema del libro. Cree una clase "BibliotecaDeReferencia" que es una subclase de "Biblioteca" y tiene un atributo extra para la lista de libros de referencia y un método extra para agregar libros de referencia. Los préstamos de libros de referencia no se permiten en esta biblioteca, por lo que el método de agregar préstamos debe lanzar una excepción si se intenta agregar un préstamo de un libro de referencia.

En el método principal de la aplicación, cree una instancia de la clase "BibliotecaDeReferencia" y agregue algunos libros y préstamos a la biblioteca y luego intente agregar un préstamo de un libro de referencia.

Ejercicio 5. Tienda de música

Se desea modelar una aplicación para una tienda de música que tiene distintos tipos de productos, como CDs, vinilos y descargas digitales. Cada producto tiene un identificador único, un título, un artista, un género, una fecha de lanzamiento y un precio de venta. Además, cada producto tiene una lista de pistas (para CDs y descargas digitales) o canciones (para vinilos).

Cree una clase abstracta "Producto" que tenga los siguientes atributos:

- id (int)
- titulo (String)
- artista (String)
- genero (String)
- fechaLanzamiento (LocalDate)
- precioVenta (double)

Cree tres subclases de "Producto" llamadas "CD", "Vinilo" y "DescargaDigital". La clase "CD" debe tener un atributo extra para la duración total del CD (en minutos) y una lista de pistas. La clase "Vinilo" debe tener un atributo extra para la cantidad de discos que tiene el vinilo y una lista de canciones. La clase "DescargaDigital" debe tener un atributo extra para el tamaño del archivo de descarga (en megabytes) y una lista de pistas.

Cree una clase "TiendaMusica" que tenga un atributo de lista de productos. La clase "TiendaMusica" debe tener un método para agregar productos a la lista y un método para calcular el precio total de todos los productos en la lista.

Además, cree una clase "CarritoCompra" que tenga un atributo de lista de productos. La clase "CarritoCompra" debe tener un método para agregar productos a la lista, un método para eliminar productos de la lista y un método para calcular el precio total de todos los productos en la lista.

En el método principal de la aplicación, cree una instancia de la clase "TiendaMusica" y agregue algunos productos de las subclases "CD", "Vinilo" y "DescargaDigital" a la lista. Luego, muestre el precio total de todos los productos en la lista. Luego, cree una instancia de la clase "CarritoCompra" y agregue algunos productos a la lista. Luego, muestre el precio total de todos los productos en el carrito de compras y elimine un producto de la lista.

Ejercicio 6. EFAteca

Construir un programa para una biblioteca que contiene libros y revistas. Ambas publicaciones tienen un código, un año de publicación y un atributo prestado.

Cuando se crea la publicación, el atributo prestado estará falso porque la publicación no está prestada. Las revistas tienen un número y los libros un nombre del autor. Ambos tipos de publicaciones deben tener métodos de consulta de todos sus atributos. Para prevenir posibles cambios en el programa se tiene que implementar una interfaz que contenga los métodos prestar(), devolver() y prestado(). La clase publicación implementará dicha interfaz.

El programa deberá obtener el número de publicaciones prestadas. Para ello, la clase principal contendrá dos métodos:

- Un método que cuente las publicaciones prestadas. Dicho método recibirá como parámetro una lista de las publicaciones existentes en la biblioteca y devolverá cuantas hay prestadas.
- El método main, que creará una matriz de referencias polimórficas de tipo publicación, invocará al método con el que se obtiene el número de publicaciones prestadas e imprimirá el número de publicaciones prestadas. También se deberá imprimir los datos de todas las publicaciones. Como, por ejemplo, crear una matriz de 2 libros y 2 revistas y considerar que se prestan un libro y una revista.

Se debe:

1. Diseñar el diagrama de clases en notación UML indicando las relaciones entre las mismas.
2. Implementar en Java las clases y/o interfaces resultantes
3. Construir un programa principal en el que se realicen las operaciones descritas anteriormente (cálculo de publicaciones prestadas usando un método específico para tal fin E impresión en el método main del número de publicaciones prestadas).