Sistema de Gestión de Biblioteca:

Contexto: Estás desarrollando un sistema de gestión para una biblioteca que necesita manejar diferentes tipos de materiales como libros, revistas y medios digitales. Cada material tiene características comunes, pero también atributos y métodos específicos.

Consigna:

Define una clase base Material con los siguientes atributos y métodos:

Atributos:

title (título del material, de tipo String)

author (autor del material, de tipo String)

year (año de publicación, de tipo int)

Métodos:

getDetails(): método abstracto que retorna un String y será implementado por las subclases.

Crea las subclases Book, Magazine y DigitalMedia que hereden de Material. Implementa los métodos específicos en cada subclase:

En Book:

Atributos adicionales: genre (género del libro, de tipo String)

getDetails(): retorna una cadena con todos los detalles del libro.

En Magazine:

Atributos adicionales: issueNumber (número de edición, de tipo int)

getDetails(): retorna una cadena con todos los detalles de la revista.

En DigitalMedia:

Atributos adicionales: format (formato del medio digital, de tipo String - por ejemplo, "PDF", "EPUB")

getDetails(): retorna una cadena con todos los detalles del medio digital.

En una clase Main, crea instancias de Book, Magazine y DigitalMedia, y llama a sus métodos getDetails() para mostrar los detalles de cada material.

Sistema de Gestión de Hospital:

Contexto: Un hospital necesita un sistema para gestionar a sus empleados y pacientes. Hay diferentes tipos de empleados como médicos, enfermeras y administrativos, así como diferentes tipos de pacientes como internos y ambulatorios. El sistema debe ser capaz de gestionar y almacenar la información específica de cada tipo de empleado y paciente.

Consigna:

Define una clase base Person con los siguientes atributos y métodos:

Atributos:

name (nombre de la persona, de tipo String)

age (edad de la persona, de tipo int)

Métodos:

getInfo(): método abstracto que retorna un String y será implementado por las subclases.

Crea las subclases Employee y Patient que hereden de Person. Implementa los métodos específicos en cada subclase:

En Employee:

Atributos adicionales: employeeID (ID del empleado, de tipo String), position (posición del empleado, de tipo String)

Métodos:

getInfo(): retorna una cadena con toda la información del empleado.

calculateSalary(): método abstracto que será implementado por subclases.

En Patient:

Atributos adicionales: patientID (ID del paciente, de tipo String), ailment (enfermedad del paciente, de tipo String)

Métodos:

getInfo(): retorna una cadena con toda la información del paciente.

Crea las subclases Doctor, Nurse y Admin que hereden de Employee e implementen calculateSalary() de manera diferente según el tipo de empleado:

En Doctor:

calculateSalary(): retorna el salario base más un bono específico.

En Nurse:

calculateSalary(): retorna el salario base más horas extra.

En Admin:

calculateSalary(): retorna el salario base.

Crea las subclases InPatient y OutPatient que hereden de Patient:

En InPatient:

Atributos adicionales: roomNumber (número de habitación, de tipo int)

getInfo(): retorna una cadena con toda la información del paciente interno.

En OutPatient:

Atributos adicionales: appointmentDate (fecha de cita, de tipo String)

getInfo(): retorna una cadena con toda la información del paciente ambulatorio.

En una clase Main, crea instancias de Doctor, Nurse, Admin, InPatient y OutPatient, y llama a sus métodos para demostrar cómo se gestionan las diferentes personas en el hospital.

Sistema de Ventas y Pedidos:

Contexto: Una tienda en línea necesita un sistema para gestionar sus productos y pedidos. Hay diferentes tipos de productos como electrónicos, ropa y alimentos. Cada tipo de producto tiene atributos y métodos específicos. Además, el sistema debe gestionar los pedidos y calcular el costo total de los mismos.

Consigna:

Define una clase base Product con los siguientes atributos y métodos:

Atributos:

productID (ID del producto, de tipo String)

name (nombre del producto, de tipo String)

price (precio del producto, de tipo double)

Métodos:

getDetails(): método abstracto que retorna un String y será implementado por las subclases.

Crea las subclases Electronics, Clothing y Food que hereden de Product. Implementa los métodos específicos en cada subclase:

En Electronics:

Atributos adicionales: warranty (garantía del producto, de tipo String)

getDetails(): retorna una cadena con todos los detalles del producto electrónico.

En Clothing:

Atributos adicionales: size (tamaño de la prenda, de tipo String)

getDetails(): retorna una cadena con todos los detalles de la prenda.

En Food:

Atributos adicionales: expiryDate (fecha de caducidad, de tipo String)

getDetails(): retorna una cadena con todos los detalles del alimento.

Define una clase Order con los siguientes atributos y métodos:

Atributos:

orderID (ID del pedido, de tipo String)

products (lista de productos en el pedido, de tipo List<Product>)

Métodos:

addProduct(Product product): método para agregar un producto al pedido.

calculateTotal(): método que calcula y retorna el costo total del pedido.

En una clase Main, crea instancias de Electronics, Clothing y Food, y agrégalas a una instancia de Order. Llama a los métodos para agregar productos y calcular el costo total del pedido.