2° Modelo del PRIMER PARCIAL – PROGRAMACIÓN II – 2024

De acuerdo a las descripciones de las siguientes clases, se pide:
Modelar en UML, crear el código correspondiente en JAVA (reutilizando código).

Generar un proyecto en JAVA nombrado como: ***Entidades.Apellido.Nombre***, que sea de tipo *Biblioteca de Clases,* el cual tendrá las siguientes clases (todas en el paquete ***apellido.nombre***):

1.- Crear una clase llamada “Barco” que posea los siguientes atributos:
(Sólo deberán ser visibles desde las clases que hereden de ella).

- cantMaxPasajeros (entero)
- motorEncendido (booleano)
- destino (string)

Y los siguientes métodos:

- Un getter abstracto de tipo List<Pasajero> getPasajeros.
- Un getter getDestino() que retorne el valor que hay en el atributo destino para ser mostrado por consola. Este método deberá ser sobrescrito.
- ingresar(Pasajero) que reciba un objeto de tipo Pasajero y lo guarde en el atributo que corresponda, siempre y cuando la cantidad máxima de pasajeros no sea superada, en tal caso, informar por consola. Este método será abstracto.
- bool EncenderMotor() que devolverá TRUE sólo si el motor se encuentra encendido.
  En caso contrario deberá encenderlo.

2.- Crear una interface ServicioComedor que posea el método:

- void servirComida (string) el cual deberá recibir la comida a servir. Se mostrará por consola.

3.- Crear la clase Pasajero con los atributos: apellido (String), nombre y rango (enumerado Rangos {Camarero, Cocinero, Capitan, Cliente}). Los modificadores de visibilidad se dejarán a su elección.

4.- Crear una clase llamada “Crucero” que herede de “Barco” y que posea el siguiente atributo:

- pasajeros (List<Pasajero>): lista genérica de tipo Pasajero. Asociar este atributo con el getter getPasajeros.

5.- Crear una clase llamada “Transatlántico” que herede de “Barco”, que implemente “ServicioComedor” y que posea el siguiente atributo:

- pasajeros (List<Pasajero>): lista genérica de tipo Pasajero. Asociar este atributo con el getter getPasajeros.

6.- Realizar constructores para permitir inicializar el destino y la cantidad máxima de pasajeros (en Barco y sus derivadas) y los nombres, apellidos y rango (en Pasajero).

7.- Generar el ***.jar*** del proyecto de tipo Biblioteca de Clases. Crear un nuevo proyecto de tipo Aplicación de Consola (Test.Apellido.Nombre), agregar el ***.jar*** y:
a) Crear un crucero y un transatlántico, ambos con una capacidad máxima de 3 pasajeros y como destinos ‘Brasil’ y ‘Australia’ respectivamente.
b) Crear 3 objetos de la clase “Pasajero” y agregarlos al crucero.
c) Crear 4 objetos más de la clase “Pasajero” y agregarlos al transatlántico.
d) Mostrar los destinos y el listado de pasajeros de ambos barcos. Sobrescribir el método toString().

ACLARACIÓN: todos los métodos y propiedades deberán estar documentados, los atributos y métodos nombrados correctamente respetando tanto el nombre como la forma.

Bonus track!

Una Empresa quiere desarrollar una parte de su sistema de facturación con el siguiente diseño:

Factura
Campos (protegidos): numero(long), fecha(LocalDateTime) y datosImprenta (string)
Métodos:

- Constructor()
- void calcularTotal() (abstracto)
- mostrar() (será sobrescrito)

FacturaC (deriva de Factura)
Campos (privados): items (List<Item>) y total (Double)
Métodos:

- Constructor()
- void calcularTotal() (Override)
- void agregarItem()
- String mostrar() (Override)

Item
Campos (privados): codigo(int), cantidad(int), detalle(string) y precio (double).
Getters para todos los campos.
Métodos:

- Constructor()
- String mostrarDetalles()

El procedimiento es el siguiente: Se crea un objeto FacturaC que por lo menos tendrá dos ítems distintos (se crearán dos objetos de tipo Item distintos).
A la factura se le agregarán los dos ítems (a la lista genérica Items), utilizando el método agregarItem(), pasándole los parámetros que usted crea conveniente.
El método mostrar() expondrá los datos de la factura, asimismo, el método mostrarDetalles() mostrará todos los datos del ítem en cuestión.
El método calcularTotal(), calculará el costo de todos los ítems de la factura (precio * cantidad).

Nota: Los métodos mostrar() y mostrarDetalles() deben utilizar el objeto 'StringBuilder'.
De acuerdo a las descripciones de las anteriores clases, se pide:
Modelar en UML, crear el código correspondiente en JAVA (reutilizando código).