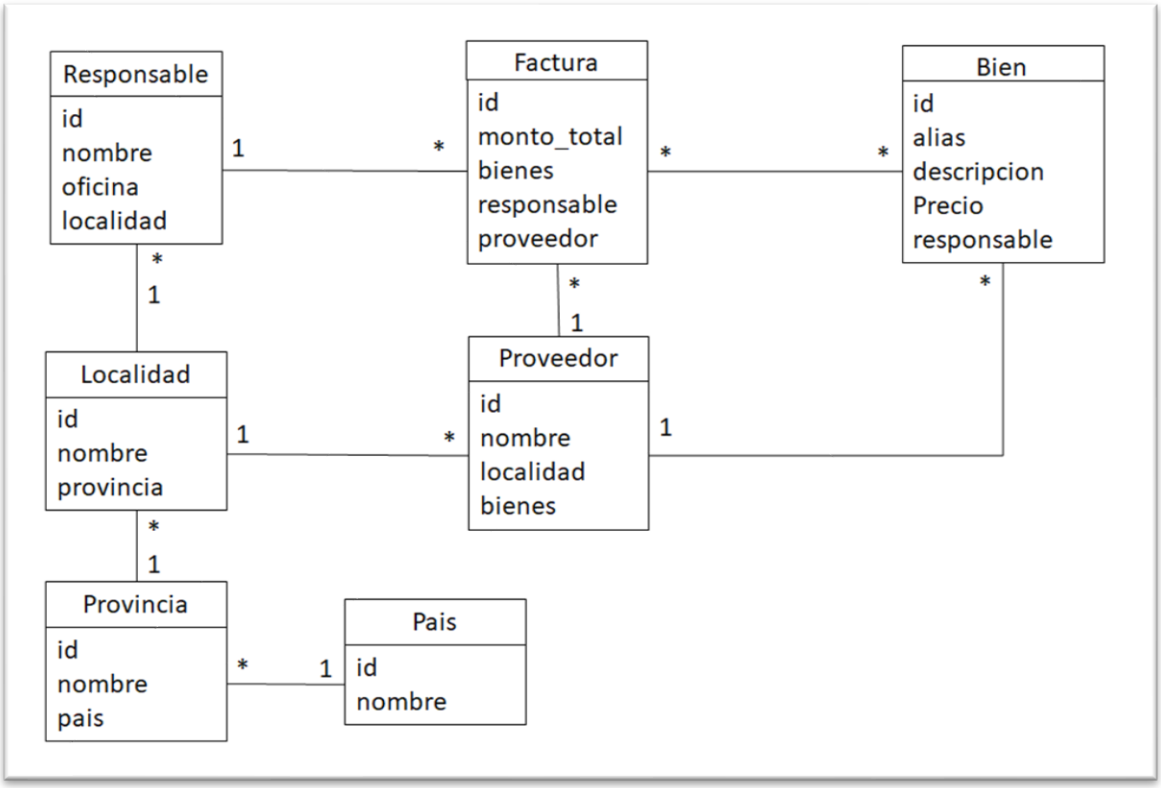
Ejercicios de clase

1. Una empresa de turismo necesita un sistema de gestión de pasajes para el cual se requiere implementar una clase que modele un Pasaje de avión el cual cuenta con la siguiente información:
    - Código de pasaje
    - Numero de vuelo
    - Ciudad origen
    - Ciudad destino
    - Fecha y hora de partida
    - Fecha y hora de arribo
    - Fecha máxima de cancelación (solo fecha y debe ser anterior por 24 horas a la fecha de partida)
    - Asiento asignado
    - Pasajero (nombre, apellido, correo electrónico, nro de pasaporte)
    - Monto pagado.
    - Tipo de servicio (primera clase, business, economy)
    - Aerolínea que brinda el servicio.
- Cree las clases necesarias para modelar esta situación.
2. Implementar el método equals teniendo en cuenta que
    a. 2 pasajes son iguales si tienen el mismo código de pasaje la misma ciudad origen y la misma ciudad destino
    b. 2 personas son iguales si tienen el mismo correo electrónico y numero de pasaporte.
3. Agregar al pasaje el método “duración” que retorna la duración del viaje (puede usar el método https://docs.oracle.com/javase/10/docs/api/java/time/temporal/TemporalUnit.html#between(java.time.temporal.Temporal,java.time.temporal.Temporal))
4. Implementar la interface comparable en la clase viaje para que compare los viajes por fecha de partida.
5. Crear una clase llamada “RegistroViajes” donde se tenga una colección de viajes.
    a. Crear un método para agregar un viaje a la colección (no pueden registrarse viajes duplicados)
    b. Crear un método que liste todos los viajes oredenados por monto pagado de manera descendente (del mas caro al mas barato)
    c. Usando el API de stream, crear un método que permita buscar pasajes por origen y/o por destino. Este método recibe dos argumentos, un origen y un destino y retorna todos los pasajes que tienen dicho origen O dicho destino.
    d. Crear un método que dado retorne el pasaje que se corresponde con un código de pasaje y un pasajero. Este método deberá retornar un optional.
    e. Crear un método que, usando el api de Stream, calcule el costo promedio de pasajes para un pasajero recibido por parámetro. (para calcular el promedio puede utilizar https://docs.oracle.com/en/java/javase/12/docs/api/java.base/java/util/stream/DoubleStream.html#average())

Problema Integrador

La empresa SENERISIMA solicita un sistema que gestione el inventario de todos sus bienes. El sistema de gestión de inventarios va a componer:



[Nota]; El gráfico es una representativa de todos los modelos, no necesariamente estará toda la información, por eso es muy importante leer a continuación y respetar todos los atributos

Bien:

- id: Identificador incremental y única, no debe haber repetidos en el registro. Recomendado: definirlo como una variable static que irá incrementando cada vez que se crea un nuevo objeto que instancia a la clase Bien.
- alias: Una cadena String que registra el alias de un bien, por ejemplo: “Samsung Galaxy A52”, “Notebook i3 7ma generación”, etc.
- descripción: Una cadena String que detalla la información sobre el bien.
- precio: valor Double
- responsable: objeto Responsable

Proveedor:

- id: <idem anterior>
- nombre: cadena String
- localidad: objeto Localidad
- bienes: lista de objetos de Bien que el proveedor trabaja

Responsable:

- id: <idem anterior>

- nombre: cadena String
- localidad: objeto Localidad
- oficina: cadena String, por ejemplo: “Departamento de informática”, “Departamento de Recursos Humanos”, etc.

Factura:

- id: <idem anterior>
- monto_total: valor Double, total de todos los bienes
- bienes: lista de objetos de Bien
- responsable: objeto Responsable que solicita el Bien
- proveedor: objeto Proveedor que provee el Bien
- **fecha: objeto Date**

Localidad:

- id: <idem anterior>
- nombre: cadena String
- provincia: objeto Provincia

Provincia:

- id: <idem anterior>
- nombre: cadena String
- país: objeto País

País:

- id: <idem anterior>
- nombre: cadena String

En este sistema, se solicita crear clases abstractas:

- Persona: va a coincidir con las clases Responsable y Proveedor.
- Ubicación: va a coincidir con las clases Localidad, Provincia y País.

Dada la entrevista con la empresa, se solicita que el sistema trabaje con los siguientes requerimientos.

Si el proveedor está en el mismo país, provincia y localidad que el responsable el monto total de todos los costos se calcula la suma de todos los precios + 10% del total por el envío. Si el proveedor está en el mismo país y provincia, pero no en la misma localidad, que el responsable el monto va a ser + 30% del total por el envío. En el caso del mismo país y distintos el resto, el monto va a ser +70% del total por el envío. Ningún proveedor proveerá ningún bien a los responsables si están en diferentes países.

Todos los proveedores hacen un descuento del 10% del monto total, si la cantidad de los bienes es mayor a 5, un 20% si la cantidad fuere mayor a 10. Ninguna factura debe permitirse más de 20 bienes.

Implementar una interface CriterioBusqueda<T> con un método que permita determinar si la clase que contiene el objeto localidad es igual a la localidad dada por parámetro y sus provincias y paises son iguales (son iguales cuando los nombres son iguales, definir el equalsTo a cada uno de ellos). El método debe ser public Boolean esDeLocalidad(Localidad l). La clase Persona implementa esta interfaz.

En todo caso, los objetos de las clases Responsable, Proveedor, Bien y Factura se registrarán dentro de una clase Empresa, donde va a contener todos los métodos asociados a los requerimientos recién mencionados. También la clase misma, debe poder capturar todas las excepciones. La clase Empresa debe implementar varios métodos como se solicita a continuación:

ArrayList<Factura> listarFacturaByProveedor(Proveedor p) Se debe poder listar todas las facturas de un proveedor especificado.

ArrayList<Factura> listarFacturaByResponsable(Responsable r) Se debe poder listar todas las facturas de un responsable especificado.

ArrayList<String> listarNombresProveedores(): Se debe poder listar todos los nombres de los proveedores por una determinada Localidad, ordenado de forma ascendente por nombre. Utilizar el *stream filter*, sorted, map y collect. Filter con el método esDeLocalidad(…), sorted ordenado ascendente por nombre, map tomando el nombre de cada uno de los proveedores.

ArrayList<Bien> listarBienesByLocalidad(Localidad l): Se debe poder listar todos los bienes solicitados por los responsables que reside en una Localidad especificada.

ArrayList<Bien> listarBienesByPrecioMayor(Double precioMayor): Se debe poder listar todos los bienes con un precio mayor a un valor pasado por parámetros.

ArrayList<Factura> listarFacturasByMontoMayor(Double montoMayor): Se debe poder listar todas las facturas con un monto total mayor a un valor pasado por parámetros.

ArrayList<String> listarFactuadasPorProveedor(Proveedor p) Se debe poder listar cadenas de String que debe componer de la siguiente manera: “En la fecha <Fecha>, <NombreProveedor> facturó con un total de $<montoTotal> con <bienes.size()>”. Es decir, se debe poder listar todas las facturas realizadas por un proveedor p, ordenado por fecha de forma Descendente (de nuevo a antiguo) en formato DIA/MES/AÑO, implementar la clase Comparable y ordenarlas con Collection.sort.

Por ejemplo:

En la fecha 07/05/2022, Martín Domínguez facturó con un total de $132102 con 15 bienes

En la fecha 07/05/2022, Leandro Amarrillo facturó con un total de $332102 con 23 bienes