INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

ALUMNO : VALLEJO SERRANO EHECATZIN

MATERIA: POO

GRUPO: 2CV4

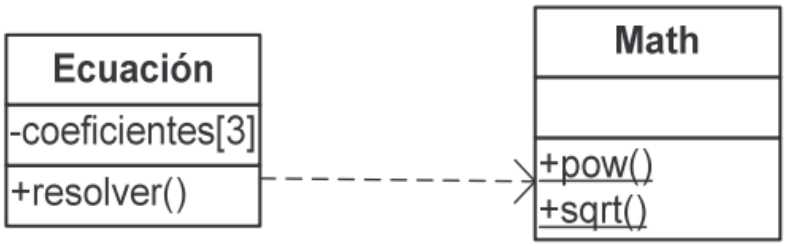
Práctica 4

Introducción

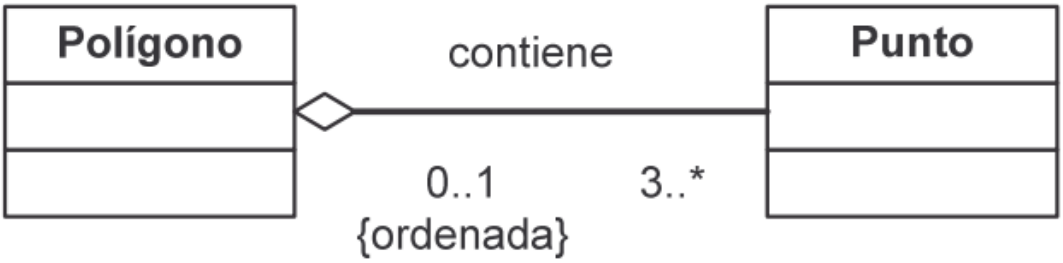
Relaciones entre clases

- Asociación:
  Representan relaciones estructurales entre las clases (la forma en que están relacionadas entre si las clases). Permite asociar objetos que colaboran entre si. Cabe destacar que no es una relación fuerte, es decir, el tiempo de vida de un objeto no depende del otro. Se representa con una línea continua entre las clases.

- Dependencia o uso:
  Es cuando una objeto es instanciado por otro objeto para su uso. Un objeto instancia (usa) a otro objeto para que le ayude a realizar una tarea, posteriormente si el objeto instanciado ya no es necesario el objeto instanciado puede ser eliminado. Se representa con una flecha punteada. En este ejemplo la ClaseA usa a la ClaseB.



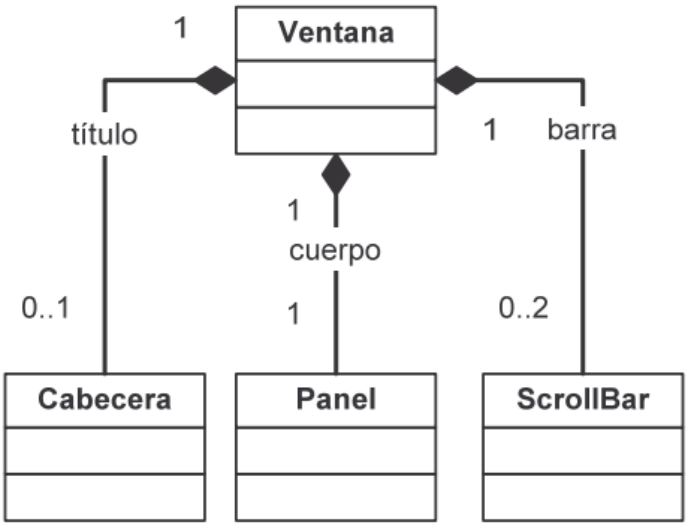
- Agregación:
  Es cuando un objeto esta compuesto de otros objetos, es decir, los objetos forman "parte de" para construir un objeto más complejo. En la agregación el tiempo de vida de los objetos que forman "parte de" no dependen del tiempo de vida del objeto contenedor, es decir, al desaparecer el objeto contenedor los otros objetos siguen existiendo y se pueden utilizar para que formen parte de otros objetos. Se representa con una línea con un rombo sin relleno en el extremo, él cual indica la clase contenedora de objetos de otras clases. En este ejemplo la clase Polígono contiene objetos de la clase Punto.



- Composición:

  Es cuando un objeto esta compuesto de otros objetos, es decir, los objetos forman "parte de" para construir un objeto más complejo. Es similar a la

relación de agregación, sin embargo, en la composición el tiempo de vida de los objetos que forman "parte de" si dependen del tiempo de vida del objeto contenedor, es decir, al desaparecer el objeto contenedor los otros objetos también desaparecen. Se representa con una línea con un rombo relleno en el extremo, él cual indica la clase contenedora de objetos de otras clases. En este ejemplo la clase Ventana contiene objetos de la clases Cabecera, Panel y ScrollBar.



Para especificar la multiplicidad de una asociación hay que indicar la multiplicidad mínima y la multiplicidad maxima , a continuacion se presenta una tabla con cada multiplicidad existente.

Multiplicidad	Significado
1	Uno y sólo uno
0..1	Cero o uno
N..M	Desde N hasta M
*	Cero o varios
0..*	Cero o varios
1..*	Uno o varios (al menos uno)

Análisis y diseño

Para esta cuarta practica , es necesario preguntarnos : ¿Qué me piden que haga?.

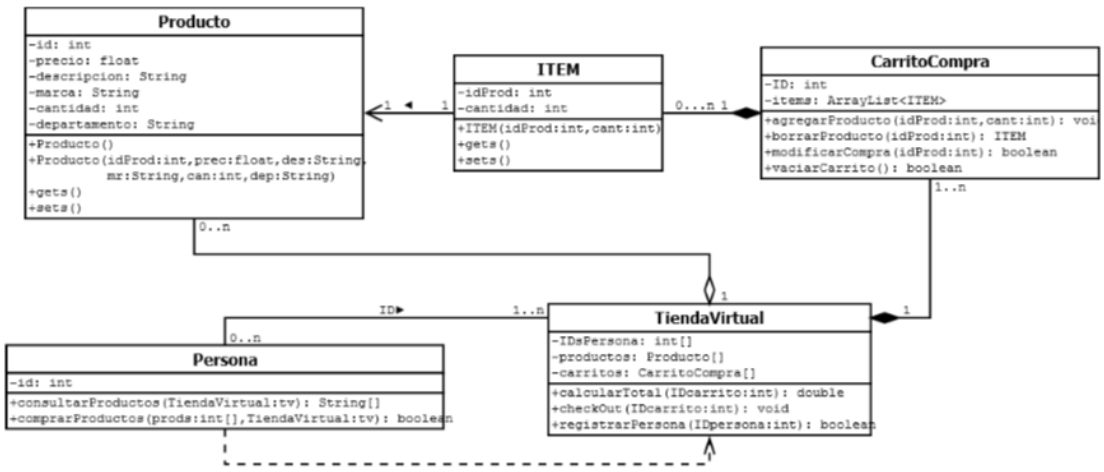
En este caso se necesitamos “echar a andar “ , por decirlo de alguna manera coloquial , una app que le permita a una cierta persona , realizar compras. Entonces en este caso , estamos hablando de una tienda virtual.

La persona de poder:

- Observar que artículos hay disponibles en cierta tienda.
- Registrarse en una tienda.
- Realizar compras.

Eso sería lo fundamental en cuanto a la Persona , la cual será cliente de una o más tiendas.

En cuanto al funcinamiento y diseño de toda al app, se nos proporciona el siguiente diagrama de clases:



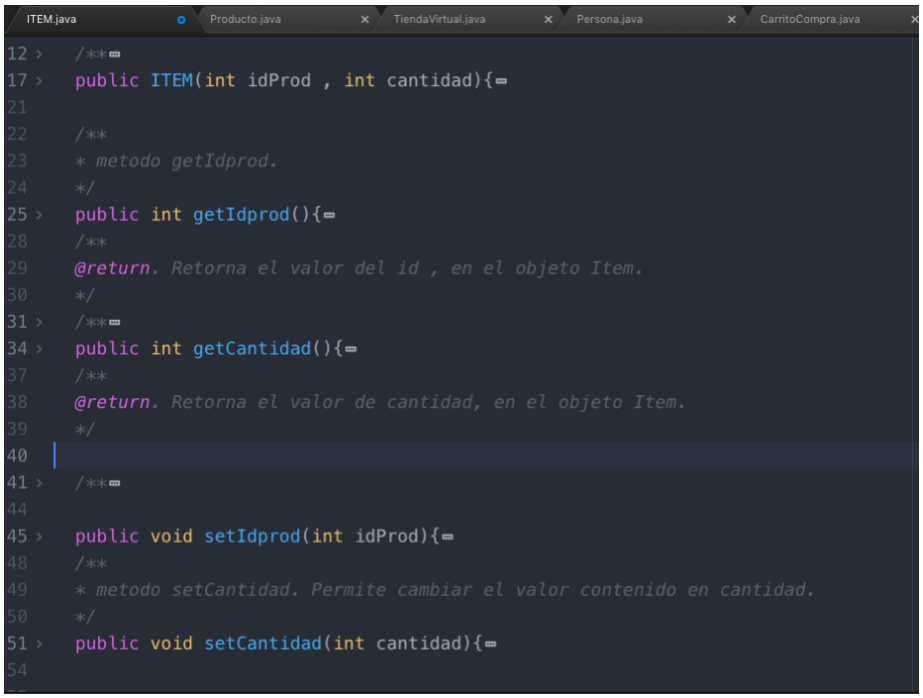
Aquí claramente podemos ver , que a cada una de las relaciones entre clases , ya tiene un tipo , asi como una multiplicidad. A demás se nos brindan algunos métodos que deben de tener cada clase. Y se nos pide que en base a esto , hagamos funcionar al App , cumpliendo con el diseño. Podemos agregar metodos. Y lo que haga falta , para cumplir con el proposito de la practica.

Entonces , lo realmente complicado para mi , en este caso , es hacer funcionar esto , con un modelo que ya esta establecido.

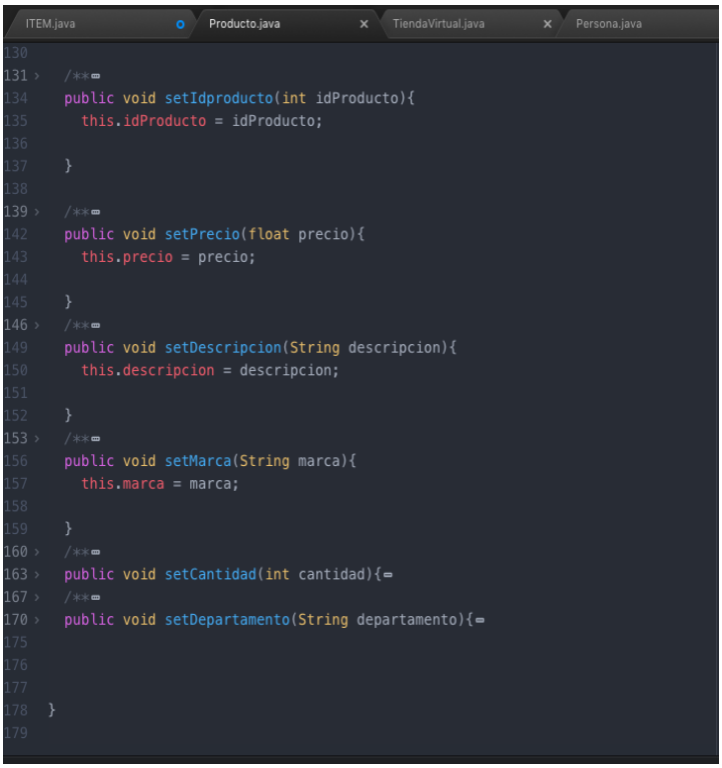
Lo primero que realicé para empezar esta practica, fue establecer y escribir los metodos que me estan dando como muestra: , además de implementar metodos fundamentales ,

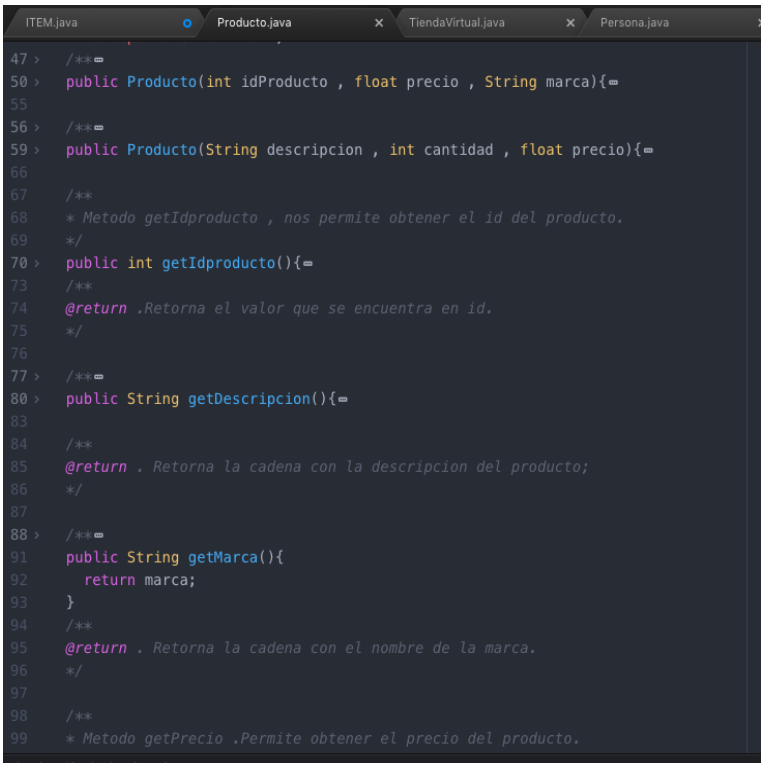
como los son los metodos SET y GET , para poder obtener y modificar los atributos de cada clase, si es que en algun punto de la practica lo requiero.

CLASE ITEM



.PRODUCTO





Es evidente que cada clase debe tener sus métodos SET Y GET , por lo tanto evitare poner capturas, ya que seria muy redundante. Me enfocaré en mostrar los métodos clave de cada clase. Que en este caso las clases mas divertidas , o de las cuales pude observar que recaían mas acciones para poder hacer funcionar la app , fueron : CarritoCompra y TiendaVirtual.

CarritoCompra

Algunas de las acciones fundamentales , al momento de comprar, algo en la tienda virtual ,es donde se va a almacenar. En este caso se establece la creación de una clase carritocompra , en la cual se tendrá un ArrayList de objetos ITEM. Entonces muchos de los métodos recaen en el manipular esta lista.

Métodos como :

- Buscar Item
  Si la persona quiere comprar algo , lo ideal va a ser buscarlo.

```java
public int buscarItem(int idProducto){
  int posicion = 0;
  int aux = 0;

  for( ITEM i : items){

    if(idProducto == i.getIdprod()){
      posicion  = aux;
      break;
    }
    else{
      posicion = -1;
    }
    aux++;
  }

  return posicion;

}
```

Además , puede cambiar de opinión y puede que desee cambiar la cantidad de artículos , así como quitarlo definitivamente .

- Modificar ITEM

```java
public boolean modificarItem(int idProducto,int cantidad){

  boolean estado = false;

  for(ITEM i: items){
    if(idProducto == i.getIdprod()){
      i.setCantidad(cantidad);
      estado = true;
    }
    else{
      estado = false;
    }

  }

    return estado;

}

/**
```

- Borrar Item

```
142    public boolean borrarItem(int idProd){
143
144      boolean estado = true;
145
146      for(ITEM i: items){
147        if(idProd == i.getIdprod()){
148          items.remove(i);
149          estado = true;
150        }
151        else{
152          estado = false;
153        }
154
155      }
156
157      return estado;
158    }
```

En cuanto a TiendaVirtual , como se establece que se debe contener Arrays, para almacenar Productos , así como Carritos e Ids . Aquí se presenta un problema al momento de dar de alta un nuevo producto. Entonces lo que hice fue establecer un tamaño definido para cada Array.

Métodos :

- Contar Productos

```
54    public int contarProductos(){
55      int i = 0;
56      int contador = 0;
57
58      while(productos[i]!= null){
59        i++;
60
61      }
62
63      return i;
64    }
```

Este método es de mucha ayuda , ya que me cuenta cuantos productos hay en el ArrayProductos.

** Nota : Realice los mismos métodos para los Arrays restantes.

Una vez teniendo este método , es fácil solucionar el problema de dar de alta un producto , solamente requiero saber el tamaño del Arreglo ,contando cuantos artículos tiene y metiendo el nuevo Producto en el espacio que le corresponde , de la siguiente manera:

```
112     public void darAltaProducto(Producto x){
113
114       int existentes = contarProductos();
115
116       if(productos[0] == null){
117          productos[0] = x;
118       }
119       else{
120
121         productos[(existentes-1)+1] = x;
122       }
123
124     }
125
```

** Nota : Se realizan los mismos métodos para dar de alta a una persona y a un carrito.

- Buscar Carrito

  Es fundamenta una vez que se asigna y se da de alta a un nuevo cliente , además de dar de alta a su carrito asociado, buscar este carrito dentro del Array , para poder , posteriormente realizar métodos que involucrarán obtener información acerca del carrito .

```java
public int buscarCarrito(int idc){

  int i = 0;
  int j = contarCarritos();
  int aux2 = 0;
  int aux = 0;
  int posicion = 0;


  for(i = 0; i < j; i++){

    aux = carritos[i].getIdCarrito();
    System.out.println("Idcarrito analizado :"+aux);

    if(idc == aux){

      posicion = i;

    }

    else{
      posicion = -1;
    }

  }

  return posicion;
```

En realidad, estos son para mi , la clave de solucionar el problema que se establece en esta practica. Desafortunadamente tuve un problema en cuanto a realizar búsquedas de un determinado producto , ya sea en el caso de la clase CarritoCompra, así como en la clase TiendaVirtual.

Por lo cual al momento de establecer mi Main , lo hice la siguiente manera:

```java
public class Main2{

  public static void main(String[]args){

    TiendaVirtual a = new TiendaVirtual("Cuidado con el Catzin");
    CarritoCompra nuevo = new CarritoCompra(2203);
    CarritoCompra nuevo2 = new CarritoCompra(2205);
    int[]idsprod = new int[50];
    int cantidad = 0;
    Scanner entrada = new Scanner(System.in);
    Persona cliente = new Persona("Daniel",2459494);
    Producto auxiliar = new Producto(123,250,"Gorra negro mate","Nike",20,"Deportes");
    Producto auxiliar2 = new Producto(567,500,"Tennis deportivos LunarLon","Nike",20,"Deportes");
    Producto auxiliar3 = new Producto(723,500,"Pantalon Skynny mezclilla","Nike",20,"Deportes");
    Producto auxiliar4 = new Producto(893,500,"Pantalon Skynny negro","Nike",20,"Deportes");
    a.darAltaProducto(auxiliar);
    a.darAltaProducto(auxiliar2);
    a.darAltaProducto(auxiliar3);
    a.darAltaProducto(auxiliar4);
    /*qSystem.out.println(cliente.getNombre()+cliente.getIdpersona());
    cliente.mostrarDescripciones(cliente.consultarProductos(a),a);*/
    /*nuevo.agregarItems(auxiliar.getIdproducto(),2);
    nuevo.modificarItem(123,10);
    /*System.out.println(nuevo.getItems().get(0).getCantidad());*/
    a.darAltaCarrito(nuevo);
    a.darAltaPersona(cliente);
```

Hasta lo que pude realizar , antes de tener problemas con la actual versión de Java , mi practica me permite, siguiendo el diseño establecido:

- Crear una Tienda Virtual , que recibe un determinado nombre.
- Crear Carritos de compra , que reciben un determinado ID.
- Crear objetos Persona , que en este caso son los clientes.
- Dar de Alta diferente numero de productos.
- Dar de Alta diferente numero de carritos.
- Dar de alta diferente numero de personas.
- Permitir a una persona visualizar los productos.

Conclusiones:

Está practica me permitió aprender a entender , que es necesario el trabajo en equipo. Ya que realmente era un código mucho más grande que los anteriores. Además también aprendí a codificar una determinada aplicación con un modelo determinado, puede observar como se codifica y se utilizan las relaciones , respetando su multiplicidad.

En este caso no puse capturas , ya que lo métodos que logre realizar , no pueden dar ejemplo de un funcionamiento esperado de la app. Pero de los errores se aprende y mejorare mi desempeño en cuanto a practicas futuras.