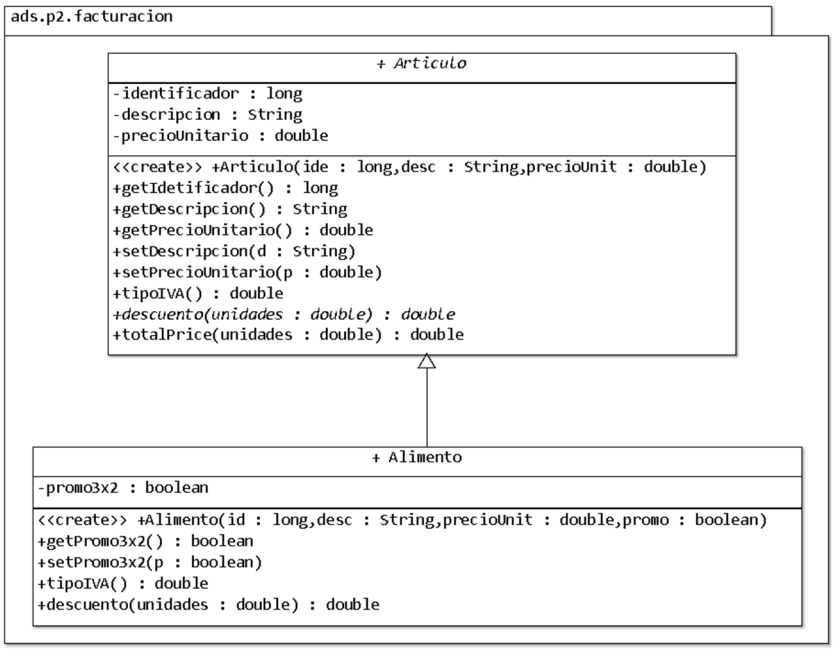
Práctica 2

Introducción al Diseño Orientado a Objetos

Inicio: A partir del 10 de febrero.
Duración: 2 semanas.
Entrega: En Moodle, una hora antes del comienzo de la siguiente práctica según grupos (semana del 24 de febrero)
Peso de la práctica: 15%

El objetivo de esta práctica es introducir los conceptos básicos de orientación a objetos, desde el punto de vista del diseño, así como de su correspondencia en Java, con particular énfasis en el diseño de clases con relaciones de herencia, asociación y agregación mediante diagramas de clases en UML.

Apartado 1 (4 puntos):



Sea una aplicación que almacena información los artículos de venta en grandes superficies. El siguiente diagrama de clases describe dos clases: una clase base **abstracta** `Articulo` que almacena el identificador, la descripción y el precio unitario de cada artículo (como **atributos privados**), y una subclase `Alimento` que modela los artículos de alimentación con un atributo privado adicional indicando si se les debe aplicar la promoción 3x2. Además del constructor y los métodos *getters* y *setters*, la clase `Articulo` tiene otros tres **métodos públicos**: el método *tipoIVA()* de la clase base `Articulo` devuelve el porcentaje de IVA que corresponderá aplicar sobre su precio según el tipo de artículo de que se trate; el método **abstracto** *descuento(unidades)* de la clase `Articulo` devuelve el descuento aplicable al precio sin impuestos del artículo; y el método *precioTotal(unidades)* calcula y devuelve el precio final a facturar por ese número de unidades del producto, incluyendo la aplicación del descuento y del IVA que le corresponda.

A continuación se muestra la implementación completa de dichas clases en Java. Como puedes ver, las clases se han incluido en el paquete "ads.p2.facturacion". Los paquetes sirven para organizar las clases que intervienen en programas grandes, con muchas clases y varios paquetes. De esta manera, clases más relacionadas entre sí se declararán en el mismo paquete. La estructura de los paquetes es jerárquica y se ha de corresponder con la estructura física de directorios (es decir, las clases han de almacenarse en un directorio "ads.p2.facturacion").
Los "constructores" tienen el mismo nombre que la clase, y son invocados mediante **new** para construir un objeto (crear una instancia de la clase) e inicializar sus atributos. Así, el constructor de `Articulo` toma tres parámetros e inicializa los atributos identificador, descripción y precio unitario. Fíjate también en cómo el constructor de la clase `Alimento`, que es **subclase** de `Articulo`, llama mediante **super(…)** al constructor de su **superclase** `Articulo` antes de inicializar el atributo promo3x2 específico de la subclase `Alimento`. Finalmente, observa cómo la clase `Alimento` sobrescribe el método *tipoIVA()* heredado de su superclase o clase padre, y cómo implementa el método abstracto *descuento(unidades)* que no tenía implementación en la clase `Articulo`.

Clase Articulo

```
package ads.p2.facturacion;

public abstract class Articulo {
  private long identificador;
  private String descripcion;
  private double precioUnitario;

  public Articulo(long id, String desc, double precio) {
       identificador = id; descripcion = desc; precioUnitario = precio;
  }
  public long getIdentificador() { return identificador; }
  public String getDescripcion() { return descripcion; }
  public double getPrecioUnitario() { return precioUnitario; }
  public void setDescripcion(String desc) { descripcion = desc; }
  public void setPrecioUnitario(double precio) { precioUnitario = precio; }

  // el tipo IVA general es 21% aplicable salvo que se redefina en una subclase
  public double tipoIVA() { return 0.21; }

  // cada subclase de articulo calculará el descuento que corresponda
  public abstract double descuento(double unidades);

  // el precio total siempre se calcula de la misma forma
  public double precioTotal(double unidades) {
       return ((precioUnitario * unidades) - descuento(unidades))
                     * (1.0 + tipoIVA());
  }
}
```

Clase Alimento

```
package ads.p2.facturacion;

public class Alimento extends Articulo {
  private boolean promo3x2;
  public Alimento(long id, String desc, double precio, boolean promo) {
       super(id, desc, precio);
       promo3x2 = promo;
  }
  public boolean getPromo3x2() { return promo3x2; }
  public void setPromo3x2(boolean promo) { promo3x2 = promo; }
  public double tipoIVA() { return 0.10; }
  public double descuento(double unidades) {
        if (promo3x2) return getPrecioUnitario() * (int) (unidades / 3);
        else return 0.0;
  }
}
```

El siguiente programa ejemplo utiliza las dos clases anteriores para calcular e imprimir el precio final de facturación de 7 unidades de chocolate a la taza con un precio unitario de 2.5 y en promoción 3x2, así como el de 4 unidades de yogur líquido a 1.25 la unidad, sin promoción. Como puedes ver, los objetos se crean mediante **new** seguido del nombre de la clase concreta con los parámetros del constructor (más adelante veremos que una clase puede tener más de un constructor). En cambio, el tipo usado en la declaración de la variable puede ser una clase abstracta o concreta. El compilador comprueba que la clase del objeto asignado a una variable sea compatible con la clase usada en la declaración de esa variable.

Programa ejemplo

```
import ads.p2.facturacion.*;

public class Ejemplo1 {
  public static void main(String args[]) {
       Articulo a1 = new Alimento(990034, "Chocolate a la taza", 2.5, true);
        System.out.println("Precio total: "
                    + a1.precioTotal(7)); // 13.75 = (7 * 2.5 - 2 * 2.5) * 1.10
```

```
        Alimento a2 = new Alimento(990268, "Yogur liquido", 1.25, false);
        System.out.println("Precio total: "
                    + a2.precioTotal(4)); // 5.5 = (4 * 1.25 - 0.0) * 1.10

    }
}
```

La salida del programa anterior es la siguiente:

```
Precio total: 13.750000000000002
Precio total: 5.5
```

Se pide:

Añade dos clases al diagrama UML y escribe su código Java para modelar los **libros** y el **tabaco** como otras dos formas de artículos en venta cumpliendo los siguientes requisitos. Cada libro tiene un atributo que identifica a qué categoría pertenece: "LibroDeTexto", "Coleccion", "Best-sellers", etc. Se debe poder cambiar la categoría de un libro una vez creado. A los libros se les aplica un tipo de IVA reducido (4%) y su descuento se calcula según su categoría: 15% para libros de texto, y 50% para la tercera unidad y sucesivas de los libros de colección. Respecto al tabaco, el precio final se calcula sin descuento y aplicando el tipo de IVA general. Completa la clase `Articulo` con un método toString() de forma que, con el código de tus dos nuevas clases, la ejecución de este programa

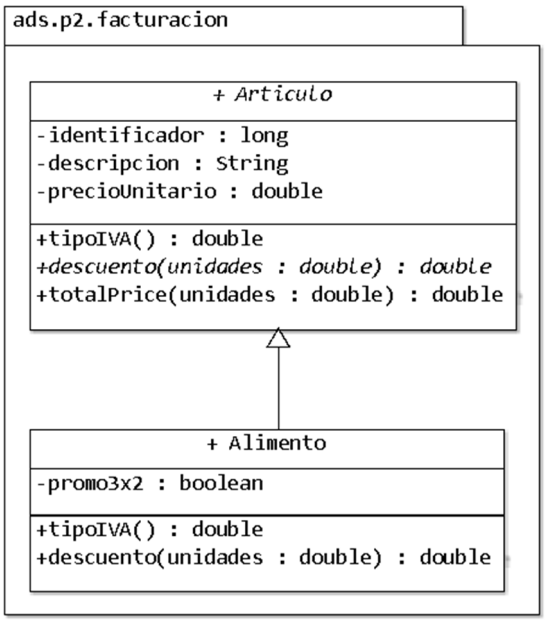
```
import ads.p2.facturacion.*;

public class Ejemplo2 {
  public static void main(String args[]) {
    Articulo a3 = new Libro(940111, "Dibujo Técnico", 15, "LibroDeTexto");
    Articulo a4 = new Libro(940607, "Grandes Clasicos", 10, "Coleccion");
    Libro a5 = new Libro(940607, "Fifty fifty", 3.25, "SinClasificar");
    Tabaco a6 = new Tabaco(690023, "Fortuna", 1.75);

    int cant = 2;
    System.out.println(cant + " unidades de " + a3 + " Precio final: " + a3.precioTotal(cant));
    cant = 5;
    System.out.println(cant + " unidades de " + a4 + " Precio final: " + a4.precioTotal(cant));
    cant = 4;
    System.out.println(cant + " unidades de " + a5 + " Precio final: " + a5.precioTotal(cant));
    cant = 1;
    System.out.println(cant + " unidades de " + a6 + " Precio final: " + a6.precioTotal(cant));
  }
}
```

produzca la siguiente salida:

```
2 unidades de Id:940111 Dibujo Técnico Precio: 15.0 Precio final: 26.52
5 unidades de Id:940607 Grandes Clasicos Precio: 10.0 Precio final: 36.4
4 unidades de Id:940607 Fifty fifty Precio: 3.25 Precio final: 13.52
1 unidades de Id:690023 Fortuna Precio: 1.75 Precio final: 2.1174999999999997
```

Nota:

En realidad, un diagrama de clases es una abstracción del código final, así que normalmente y para facilitar su comprensión no se suelen incluir todos los detalles necesarios para la codificación. De esta manera se suelen omitir los métodos "setters" y "getters", y los constructores. El siguiente diagrama muestra el diagrama de clases anterior, usando este mayor nivel de abstracción.



Por si quieres saber más...:

El método *precioTotal()* de la clase `Articulo` es un ejemplo del patrón de diseño "Template Method", que consiste en escribir código en una clase padre que llama a métodos (quizá abstractos) que se implementan en la subclase o clase hija. Tienes más información sobre patrones de diseño en:

Patrones de diseño elementos de software orientado a objetos reutilizable. Gamma, E., Helm, R., Johnson, R., Vlissides, J. INF/681.3.06/PAT. Addison-Wesley, 2003.

Apartado 2 (4 puntos):

Tienes que diseñar el módulo de una aplicación de venta de electrodomésticos que se encarga de calcular el importe final que debe pagar el cliente en función del precio del producto, de si se debe realizar la entrega a domicilio y de si en el momento de la compra se entrega un electrodoméstico viejo. En la tienda se venden tres tipos de electrodomésticos de los que se deben almacenar los siguientes datos:

- *Televisión*: marca, modelo, precio base, tamaño de pantalla y clase energética (una letra entre A y D, ver más abajo).
- *Lavadora*: marca, modelo, precio base, dimensiones, clase energética y peso.
- *Frigorífico*: marca, modelo, precio base, dimensiones, clases energética y peso.

Si el cliente recoge el electrodoméstico en mano no se añade al precio ningún importe extra. Si se debe realizar la entrega a domicilio se aplican las siguientes reglas para calcular el coste de los portes. Este coste se debe añadir al importe a pagar por el cliente:

- Para las televisiones;
  Si el tamaño de la pantalla es menor o igual a 40'' y el precio de venta supera los 500€ la entrega será gratuita.
  Si el tamaño de la pantalla es menor o igual a 40'' y el precio de venta no supera los 500€ (sin tener en cuenta los posibles descuentos) el coste del porte serán 35€.
  En cualquier otro caso, el coste del porte serán 35€ más 1€ por cada pulgada que la dimensión de la pantalla supere las 40''. Por ejemplo, el coste de llevar al domicilio del cliente una televisión de 45'' será 40€.
- Para las lavadoras:
  Si el peso es inferior o igual a 50 kg el coste del porte serán 35€.
  Si el peso supera los 50 kg el coste del porte serán 35€ más 0.50€ por cada fracción de 1 kg que el peso supere los 50 kg. Por ejemplo, el coste de llevar una lavadora de 56.5 kg al domicilio del cliente será 38.50€.
- El coste por entregar en el domicilio del cliente un frigorífico se calculará redondeando a valor entero el resultado de multiplicar su volumen (en m^3) por 70€. Por ejemplo, el precio del porte de un frigorífico de 59.5 cm de ancho, 71.9 cm de fondo y 201 cm de alto será 60€.

Todos los electrodomésticos en stock en la tienda están catalogados con un valor de eficiencia energética en función de su consumo. La clase energética del electrodoméstico se representa con una letra entre la A y la D, siendo los electrodomésticos de clase A los de menor consumo (mayor clase energética) y los de clase D los que más consumen (menor clase energética). Si en el momento de la compra el cliente entrega un electrodoméstico viejo, se le aplicará un descuento en el importe a pagar en función de la clase energética del electrodoméstico que compra y del que entrega (independientemente del tipo de ambos electrodomésticos). Las reglas para calcular la cuantía de este descuento son:

- Si el electrodoméstico viejo es de la misma clase energética del que se adquiere, se aplicará un descuento de 25€ al precio final de la compra.
- Si es de una clase energética menor, a los 25€ de descuento se le añadirán 15€ por cada clase energética de diferencia respecto al nuevo electrodoméstico. Por ejemplo, si se compra una lavadora de clase energética B y se entrega una televisión de clase D el descuento a aplicar serán 55€.
- Si es de mayor clase energética, a los 25€ de descuento se le quitarán 5€ por cada clase energética de diferencia. Por ejemplo, si se compra una lavadora de clase energética D y se entrega una televisión de clase B el descuento será de 15€.
- Si se desconoce la clase energética del electrodoméstico viejo, se aplicará un descuento de 10€.

Se pide:

Realizar un diagrama de clases UML para el nuevo módulo de la aplicación.
Nota: En el diagrama de clases en UML no es necesario incluir constructores, ni métodos *getters* y *setters*. Tampoco se necesario entregar su implementación en Java.

Apartado 3 (2 puntos):

Se va a construir una aplicación para la gestión de personal de las universidades públicas de Madrid, aunque en este apartado nos centraremos solo en la parte del diagrama de clases que refleja los siguientes aspectos.

El personal *funcionario* se identifica mediante un número de registro de personal y además tiene como características principal su fecha de ingreso en el cuerpo de funcionario y su categoría salarial (un entero que sirve para calcular su sueldo base). Algunos funcionarios ocupan puestos de responsabilidad administrativa, como es el de *administrador de centro*. Para cada administrador de centro debemos conocer la universidad a la que está adscrito y el centro (facultad o escuela) del que es administrador, así como un complemento salarial que se debe sumar al sueldo para calcular su sueldo final.

Por otro lado, se debe almacenar información relativa al personal *docente* de las universidades. Cada docente se identifica mediante su número de DNI y pertenece a un departamento (el departamento es unidad independiente de universidad y el centro, y puede haber departamentos relacionados con varias universidades o varios centros); además para cada docente debemos saber si es doctor o no (ya que esto influye en el cómputo del sueldo de todos los docentes) y debemos mantener un histórico de las actividades docentes que ha tenido asignadas, en el que para cada año y cuatrimestre se registre las asignaturas que ha impartido, con su tipo de docencia (teoría, práctica, o proyecto) y el número de horas por semanas que ha impartido en cada una de esas asignaturas en ese cuatrimestre.

Los *profesores titulares* son docentes que además pertenecen al cuerpo de funcionarios, y como características adicionales debemos conocer el número de quinquenios de evaluación docente positiva y el número de sexenios de evaluación investigadora positiva, ya que ambos aportan complementos salariales.

Finalmente, existen *profesores asociados*, que no son funcionarios sino trabajan en empresas privadas y colaboran (normalmente con dedicación a tiempo parcial) en las tareas docentes de las universidades. Para éstos debemos conocer el nombre de su empresa, su número de seguridad social, y el porcentaje de su dedicación parcial a la docencia, ya que de este porcentaje depende en parte su sueldo.

Se pide:
a) Realiza un diagrama de clases UML para dicha aplicación. No es necesario incluir constructores, ni métodos *getters* y *setters*. Tampoco se necesario entregar su implementación en Java.

b) Comenta razonadamente si convendría hacer algún cambio en ese diagrama de clases en el supuesto de que se fuese a implementar la aplicación en Java, y en caso afirmativo entrega también el nuevo diagrama.

Apartado 4 (Opcional, 1 punto):

Realiza un nuevo diagrama de clases UML, partiendo del entregado en el apartado 2 pero aparte de él, para cumplir con las siguientes características adicionales de la aplicación relacionadas con la ampliación del negocio para proporcionar un servicio técnico y de solución de incidencias por teléfono y un servicio de reparación de determinados electrodomésticos.

El servicio técnico y de solución de incidencias sólo se ofertará para algunos modelos de televisión (no todos) que deberán identificarse a través de la aplicación. Al identificar los modelos de televisión para los que se va a ofrecer el servicio se indicará la url de la documentación técnica de la televisión para que el personal de soporte pueda acceder a ella y consultarla en caso de necesidad. Además de para televisiones, también se ofrecerá servicio técnico y solución de incidencias de tablets y teléfonos móviles, aunque se contempla la posibilidad de en un futuro también hacerlo para ordenadores sobremesa y portátiles. Para las tablets y los teléfonos móviles se deberá almacenar la siguiente información: marca, modelo y url de la documentación técnica. Este servicio sólo se proporcionará a aquellos clientes que lo contraten de forma específica pagando una tarifa anual que les dará derecho a realizar todas las consultas que deseen durante el periodo de vigencia del contrato. El cobro de esta tarifa y la gestión de los clientes que tienen acceso al servicio la realizará la aplicación de facturación de la tienda.

Inicialmente se realizarán reparaciones de todo tipo de televisores y lavadoras. Todos los modelos de televisión y lavadora catalogados tendrán un precio específico por hora de reparación (es decir, que cada uno ellos podrá tener un precio diferente). La aplicación deberá calcular el coste de la reparación aplicando las siguientes reglas:

- Si el electrodoméstico se ha reparado satisfactoriamente, el coste se calculará multiplicando el número de horas empleado por el coste por hora de reparación del modelo de televisión o lavadora específico. Tanto el resultado de la reparación (satisfactoria/no satisfactoria), como el número de horas empleadas en ella se proporcionarán manualmente en el momento de la devolución del electrodoméstico al cliente.
- Si el electrodoméstico no ha podido repararse, no se cobrará nada al cliente.

Normas de Entrega:

- Se deberán entregar los apartados 1, 2 y 3 (y opcionalmente el 4). Crea un directorio por cada apartado.
- La entrega la realizará uno de los alumnos de la pareja, a través de Moodle.
- Si el ejercicio pide código Java, se ha de entregar además la documentación generada con *javadoc*. Si el ejercicio pide un diagrama de diseño, se deberá entregar en PDF junto con una breve explicación (dos o tres párrafos a lo sumo).
- Se debe entregar un único fichero ZIP / RAR con todo lo solicitado, que deberá llamarse de la siguiente manera: `GR<numero_grupo>_<nombre_estudiantes>.zip`. Por ejemplo Marisa y Pedro, del grupo 2261, entregarían el fichero: `GR2261_MarisaPedro.zip`.