EXAMEN 2ª EVALUACION. Tipo B
Unidad Didáctica 6: Abstracción: clases, paquetes, subclases e interfaces.
Unidad Didáctica 7: Clases genéricas y control de excpeciones.

Nombre: _____

1. Vamos a realizar un programa para la gestión de un supermercado, para esto utiliza como *IDE IntelliJ* y en un proyecto denominado *examen* crea un package denominado *productos* que contenga el siguiente código:

   - Crea un *enum* para tipificar los tipos de IVA existentes, siendo estos GENERAL que aplica el 21%, REDUCIDO que aplica el 10% y SUPERREDUCIDO que se le aplica el 4%
   - Crea un *enum* para indicar los distintos tipos de productos alimentarios, y cuyos valores son: PRODUCTO_PANADERIA, ACEITE y LICOR, tienen IVA superreducido, reducido y general respectivamente.
   - Crea una clase *Helper* que contenga los siguientes métodos estáticos usando expresiones regulares:
     - Método para validar el número de referencia de un producto, siendo el formato *GE,RE o SRE* (que hace referencia al tipo de IVA) seguido de tres dígitos, no está permitido que la primera cifra sea un 0.
   - Crea una interface denominada *Producto* con los siguientes métodos:
     - Método denominado *precioFinal* que se implementará en las clases derivadas y calcula el precio del producto final, después de aplicar el IVA.
     - Método estático que nos devuelva una lista de productos ordenados de precio de menor a mayor, y se le pasará como parámetro ademas de una lista de productos, el tipo de producto alimentario, de manera que solo devuelve los productos que sean del tipo alimentario correspondiente. Se deberá hacer el método aplicando *Stream* en la colección, en caso que no lo sepas hacerlo, implementa el método de todas formas, pues debes usarlo posteriormente.
     - Método estático que nos devuelva el precio total de los productos de una lista de objetos *Producto*. Usando *wildcard (?)* para que solo puedan entrar objetos *Producto*, en caso que no lo sepas hacerlo, implementa el método de todas formas, pues debes usarlo posteriormente
   - Crea una clase, denominada *ProductoGenerico*, que implementen la interface anterior y que tenga los siguientes atributos y métodos:
     - Número de referencia de un producto.
     - Tipo de producto, que pueden ser de alimentación o no.
     - Constructor o constructores.
     - Getters.
     - Sobreescribe el método *toString* para que la salida sea del tipo:

       ```
       PRODUCTO: GE123, ALIMENTARIO: NO
       PRODUCTO: RE123, ALIMENTARIO: SI
       ```
   - Crea una clase, denominada *Alimento*, que extienda de la clase anterior y que tenga los siguientes atributos y métodos:
     - Tipo de alimento, que puede ser acuerdo al anterior enum que define los tipos de productos alimentarios
     - Constructor o constructores.

  - Getters.
  - Sobreescribe el método *toString* para que la salida sea del tipo:

    ```
    PRODUCTO: RE123, ALIMENTARIO: SI, ACEITES
    PRODUCTO: RE123, ALIMENTARIO: SI, LICORES
    ```

- Implementa dos clases derivadas, una *AceiteOliva* y otra *Cerveza* que tenga como atributos y métodos:
  - IVA a aplicar.
  - Marca.
  - Precio sin IVA.
  - Constructor.
  - Getters sobreescribe el *toString* para que el resultado sea:

    ```
    PRODUCTO: RE123, ALIMENTARIO: SI, ACEITES, Aceite Picual, 10 € sin IVA, 11 € con IVA
    PRODUCTO: RE123, ALIMENTARIO: SI, LICORES, Cerveza Air, 2 € sin IVA, 2.42 € con IVA
    ```

- Implementa los métodos *hashCode* e *equals* para indicar que dos productos son iguales si tienen la misma referencia de serie.
- Implementa el código necesario para que las clases *AceiteOliva* y *Cerveza* no puedan tener clases derivadas.
- Crea la documentación para la clase *AceiteOliva*
- Crea una excepción propia, denominada *NumeroSerieException* que se lance cuando se intente crear un objeto anterior con un número de referencia con formato incorrecto.
  ´
- Clase *Main*, qué tenga la siguiente estructura:

```
//COMPROBACIÓN DEL MÉTODO DEL Helper
//USA LOS SIGUIENTES VALORES:
// RE120 devuelve True, RE012 devuelve False y RE1200 devuelve False

//CREAMOS TRES OBJETOS AceiteOliva y TRES OBJETOS Cerveza
//CAMBIA EL PRECIO DE UNO DE LOS OBJETOS
//CREA UN NUEVO OBJETO Aceite CON EL MISMO nº de serie QUE ALGUNO DE LOS ANTERIORES
//Y RESTO DE ATRIBUTOS DISTINTOS. COMPRUEBA EL MÉTODO equals

//INTENTAMOS CREAR UN Objeto numero serie CON FORMATO ERRÓNEO
//ATRAPAMOS LA EXCEPCION Y MOSTRAMOS MENSAJE EN CONSOLA

//AÑADIMOS TODOS LOS DISPOSITIVOS A UNA LISTA Producto
//MOSTRAMOS LOS DATOS EN CONSOLA, RECORRIENDO LA LISTA
//OBTENEMOS UNA LISTA DE Producto DE TIPO Licores, USA EL MÉTODO DE LA INTERFACE
//MOSTRAMOS LOS DATOS EN CONSOLA, RECORRIENDO LA LISTA
//OBTENEMOS EL PRECIO DE TOTAL DE LOS PRODUCTOS DE LA LISTA INICIAL,
//USANDO EL MÉTODO DE LA INTERFACE Y MOSTRAMOS EN CONSOLA
```

- Incluye los comentarios a la clase *Main*

CALIFICACIÓN DE RESULTADOS DE APRENDIZAJE Y CRITERIOS DE EVALUACIÓN

(RA3)

Escribe y depura código, analizando y utilizando las estructuras de control del lenguaje.

d) Se ha escrito código utilizando control de excepciones. (15) ☐

h) Se han creado excepciones. (14) ☐

(RA4)

Desarrolla programas organizados en clases analizando y aplicando los principios de la programación orientada a objetos.

g) Se han definido y utilizado clases heredadas. (13) ☐

i) Se han creado y utilizado conjuntos y librerías de clases. (13) ☐

(RA6)

Escribe y prueba programas sencillos, reconociendo y aplicando los fundamentos de la programación orientada a objetos.

f) Se han creado clases y métodos genéricos. (16) ☐

g) Se han utilizado expresiones regulares en la búsqueda de patrones en cadenas de texto. (16) ☐

j) Se han utilizado operaciones agregadas para el manejo de información almacenada en colecciones. (16) ☐

(RA7)

Desarrolla programas aplicando características avanzadas de los lenguajes orientados a objetos y del entorno de programación.

a) Se han identificado los conceptos de herencia, superclase y subclase. (10) ☐

b) Se han utilizado modificadores para bloquear y forzar la herencia de clases y métodos. (10) ☐

c) Se ha reconocido la incidencia de los constructores en la herencia. (10) ☐

d) Se han creado clases heredadas que sobrescriban la implementación de métodos de la superclase. (10) ☐

e) Se han diseñado y aplicado jerarquías de clases. (10) ☐

f) Se han probado y depurado las jerarquías de clases. (10) ☐

g) Se han realizado programas que implementen y utilicen jerarquías de clases. (10) ☐

h) Se ha comentado y documentado el código. (10) ☐

i) Se han identificado y evaluado los escenarios de uso de interfaces. (10) ☐

CALIFICACIONES PARCIALES RA

RA3 (29/110) ☐

RA4 (26/120) ☐

RA6 (48/160) ☐

RA7 (100/100) ☐

ENTREGA DE EXAMEN

Se entrega el documento de papel con tu nombre, y un fichero comprimido de la carpeta del proyecto de *IntellJ* con el formato .zip o APELLIDOS_NOMBRE.rar, subiendo éste a la plataforma mas una copia de seguridad que se entrega al profesor.