	DAM – Entornos de Desarrollo	
	Examen 2 Evaluación	**CURSO : 2022-23**

Junit, Javadoc y log4j

(5 pts) EJERCICIO 1 (Testing Junit)

En aules se encuentra un fichero **carritoCompra.zip** que se corresponde con un proyecto **Java Maven** que consta de 3 clases:

- **Producto**: Tiene 2 miembros, **nombreProducto** y **precio**, representa un producto de un hipotético carrito de la compra.
- **CarritoCompra**: Tiene 1 miembro, **listaProductos** que se corresponde con la lista de productos de un carrito de la compra. Se implementa mediante un ArrayList<Producto>.
- **ProductNotFoundException**: Una excepción definida en el mismo programa. Esta excepción se ejecuta en el momento en que se intenta eliminar del carrito un producto no presente dentro del mismo.

Debes crear un conjunto de tests unitarios mediante el uso de la librería **JUNIT**(versión 5).

- Para la clase **CarritoCompra** debes implementar los tests que comprueban los siguientes requisitos (un test por cada requisito):

1. Cuando se crea un **carritoCompra**(ejecutando el constructor) el carrito tiene 0 productos.
   El total de productos debe calcularse ejecutando el método **getCantidadProductos()**.

2. Cuando un **carritoCompra** se vacía después de ejecutar el método **vaciarCarrito()**, el carrito tiene 0 productos.
   El total de productos debe calcularse ejecutando el método **getCantidadProductos()**

3. Cuando se agrega un nuevo **producto** después de ejecutar el método **agregaProducto**(), el número total de productos se debe incrementar en uno.
   El total de productos debe calcularse ejecutando el método **getCantidadProductos().**

4. Cuando se agrega un nuevo **producto** después de ejecutar el método **agregaProducto(),** el nuevo saldo debe ser la suma del saldo anterior más el coste del nuevo **producto**.
   El saldo se debe calcular ejecutando el método **getPrecioTotal()**.

5. Cuando se elimina un **producto** del carrito ejecutando el método **borraProducto(),** la cantidad de productos debe reducirse en uno.

   El total de productos debe calcularse ejecutando el método **getCantidadProductos()**

6. Cuando se intenta eliminar un producto que no está en el carrito ejecutando el método **borraProducto()** debe lanzarse la excepción **ProductNotFoundException.**

- Para la clase **Producto** debes implementar los tests que comprueban los siguientes requisitos (un test por cada requisito):

1. Realizar un test **parametrizado** para el **constructor** de la clase **Producto** introduciendo 3 pares de valores **{String nombreProducto, double precio}** de modo que se ejecute 3 veces el constructor. **Es obligatorio que el valor del atributo precio sea negativo**.
   La idea es testear 3 veces que al crear un objeto de la clase **Producto** con precio negativo se ejecuta la excepción **IllegalArgumentException.**
   Debe emplearse la anotación **@CsvSource**

2. Realizar un test parametrizado para el método **equals** de la clase **Producto.** Este método define que 2 objetos tipo **Producto** son **iguales(devuelve true)** si tienen el mismo valor en el campo **nombreProducto,** sino son **distintos(devuelve false)**.

   En este test parametrizado se van a realizar dos pruebas, se debe comprobar que 2 objetos **Producto** son **iguales** y que 2 objetos **Producto** son **distintos**. Es decir, siendo **p1** y **p2** dos objetos de la clase **Producto** se debe comprobar que:

   - **p1.equals(p2)** devuelve **true** cuando el **nombreProducto** de ambos objetos es el **mismo**.
   - **p1.equals(p2)** devuelve **false** cuando el **nombreProducto** de ambos objetos es **diferente**.

   El método del test debe tener la siguiente estructura:

   public void testEquals(Producto a, Producto b, boolean resultadoEsperado)

   Se debe emplear la anotación **@MethodSource,** de modo que la fuente de los datos del test sera un método que debe ser definido por ti también.

(2.5) EJERCICIO 2 (Documentación JavaDoc)

Documenta las 3 clases del proyecto carrito de la compra mediante el uso de **JavaDoc** y genera toda la documentación del proyecto en formato **HTML**. Se deben documentar todas las clases junto con sus métodos. Deben aparecer las siguientes anotaciones **Javadoc** cuando sea posible**.**

- **@see**
- **{@link}**
- **@since**
- **@deprecated**
- **@author**
- **@version**
- **@param**
- **@return**
- **@throws**

Los valores que acompañan a las anotaciones te los puedes inventar.

Además de las anotaciones en cada método y en cada clase debe aparecer un comentario indicando para que se usa (si no entiendes exactamente lo que hace una clase o un método puedes inventarte el comentario).

(2.5 pts) EJERCICIO 3 (Log con log4j)

El fichero **clasesParaLog.zip** contiene 5 clases Java de un pequeño programa informático. Además de las 5 clases JAVA el **zip** tambíen contiene un fichero **log4j2.xml** con el esqueleto de un fichero de configuración de log4j.

Se debe crear un proyecto nuevo (puede ser proyecto **Maven** o proyecto **Java** estandard) que incluya las librerías de **log4j versión 5**.

En este nuevo proyecto las clases deben distribuirse en los siguientes paquetes:

- En el paquete **com.empresa.almazora.main** debe situarse la clase:
  - **AplicacionJuego.java**
- En el paquete **com.empresa.almazora.auxiliar** debe situarse la clase:
  - **ClaseAuxiliarLoteria.java**
- En el paquete **com.empresa.almazora.loterias** deben situarse las clases:
  - **LoteriaNacional.java**
  - **Primitiva.java**
  - **Quiniela.java**

Las clases de este programa no tienen ningún tipo de log definido, se deben realizar las siguientes tareas:

- Crear en cada clase un objeto que nos permita realizar el logging de sus métodos haciendo uso las clases **Logger** y **LogManager** del framework **log4j.**

- Al principio de los métodos de cada clase se debe introducir una sentencia de **logging** para cada uno de los siguientes niveles: **trace**, **info, warn y error**. El contenido de los mensajes de estas sentencias de logging debe ser el mismo para todos los niveles: “**Inicio del método {xxxx}**”, donde {xxxx} representa el nombre del método.

- Generar un fichero de configuración **log4j2.xml** con las siguientes características:

    1. Debe contener 2 variables, de nombre **autor** y **rutaLogs**, cuyos valores deben ser respectivamente “**Steve Jobs**” y “**misLogs**”.

    2. Debe contener un **appender** de tipo **Console** y de nombre **miConsola**.

       El formato(layout) en el que se mostraran las sentencias de log será el siguiente y en este orden:

        - año-mes-dia hora:minutos:segundos.milisegundos
        - Nivel de Log.
        - Valor de la variable autor entre corchetes.
        - El texto **Linea:** y a continuación el número de linea de la sentencia de log.
        - Nombre del paquete junto con el nombre de la clase que ha generado la sentencia de log.
        - Carácter guion(-) y a continuación el mensaje de log.
        - Salto de linea.

        - Ejemplo linea de Log con el formato que se pide:

            - 2023-03-03 09:58:30.913 INFO [Steve Jobs] Linea:22 com.empresa.almazora.main.AplicacionJuego - Inicio del método main

       El appender **miConsola** debe emplearse únicamente por las clases se encuentren bajo el paquete **com.empresa.almazora.main.**

       Para ello, en el fichero de configuración se debe crear un **logger**, el cual debe estar configurado para que solo se muestren los mensajes de los niveles **warn, info y error**. Además este **logger** debe hacer uso del atributo **additivity** para que las sentencias de log no se muestren repetidas.

    3. Debe contener un **appender** de tipo **File** y de nombre **fichero**.

        - El formato(layout) en el que se mostraran las sentencias de log será el que tu creas más conveniente.
        - El log se debe guardar en la ruta **fichero/miLog.log**.
        - Este **appender** será empleado por el logger raiz **rootLogger.**
        - Cada vez que se ejecuta el programa el contenido del archivo **fichero/miLog.log** se borra, de modo que las sentencias de logging grabadas anteriormente se pierden.

    4. Debe contener un **appender** de tipo **RollingFile** y de nombre **holaBebe**.

        - El formato(layout) en el que se mostraran las sentencias de log será el que tu creas más conveniente.
        - El log se debe guardar en la ruta **misLogs/fichRotativoTiempo/{dd-MM-hh-mm}/logTiempo.log,** en la definición de la ruta debes emplear la variable **rutaLogs** que ha sido creada anteriormente.
        - Se debe configurar el appender para que se genere un nuevo fichero de log cada **5 minutos**.
        - Este **appender** será empleado por el logger raiz **rootLogger.**

    5. El logger raiz **rootLogger** debe estar configurado para que solo muestre los mensajes de los niveles **warn y error**.