TP 3 – EDGARDO LORENZO (2C)
FABRICA DE PRODUCTOS ELECTRONICOS

Implementacion del Tema 15 (EXCEPCIONES)

Se crea la Clase OutOfStockException.cs para controlar la cantidad de materiales disponibles para poder fabricar un producto ya sea de tipo Keyboard o Notebook.
La excepcion se lanza, en el caso de que aplique, desde la clase Factory.cs desde el metodo StockAvailable, la atrapa dentro de la misma clase por el set de la propiedad Create (llamadora de StockAvailable) y la vuelve a lanzar para ser finalmente atrapada y mostrada por la clase BuildFrm.cs dentro de su manejador btnBuild_Click (Donde se utiliza la propiedad Factory.Create = new TIPO PRODUCTO).

Implementacion del Tema 16 (TEST UNITARIOS)

Se crea el proyecto UnitTestEntidades, dentro del mismo la clase Product_Test.cs en la cual se evalua con un metodo por TRUE y otro por FALSE la comparacion entre 2 productos en base a su tipo, numero de serie y modelo.

Implementacion del Tema 17 (GENERICS)

1. Dentro de la clase Materials.cs se implementa el metodo LoadMaterialsNeeded parametrizado como <T> que recibe un objeto T. Dicho metodo se utiliza para cargar un diccionario <string,int> de materiales por cada tipo de objeto, sin necesidad de tener una lista hardcodeada de entrada.
   Funciona de la siguiente manera:

- Cada objeto tambien tiene un atributo de tipo diccionario de materiales
- Se llama al metodo y se le pasa como parametro el objeto (Keyboard, Notebook, el que fuese)
- Se recorre las keys de materiales del objeto y si los materiales NO estan dentro del diccionario de materiales de la clase Materials, se agrega el nombre del material y un valor random.

2. Dentro de la clase Factory.cs se implementa el metodo StockAvailable parametrizado como <T> que recibe un objeto T. Dicho metodo se utiliza para corroborar que en el diccionario de materiales haya stock suficiente para poder crear lo que cuesta crear un elemento de tipo T. Al ser Generics puede ser una instancia de MechanicalKeyboard, Thinkpad o si en el futuro se decide agregar otro tipo de producto mas, tambien.
   Funciona de la siguiente manera:

- Como mencionamos en el pto. 1 c/objeto tiene un atributo de tipo diccionario de materiales
- Se recorren las claves del diccionario de materiales del objeto
- Se recorren las claves del diccionario de materiales de Materials (desde Factory, ya que esta tiene un atributo de tipo Materials)
- Se comparan los values de dichas claves
- Si el value del stock de Materials es >= al value del stock del objeto se setea bool en true. Se realiza un break y se procede a evaluar otro material.
- Caso contrario no existen suficientes materiales para crear un objeto de dicho tipo y se procede a lanzar una OutOfStockException

3. Dentro de la clase Factory.cs se implementa el metodo SubstractMaterials parametrizado como <T> que recibe un objeto T. Dicho metodo se utiliza para restar de la lista de materiales (en poder de la clase Factory) los materiales que consumen crear una instancia de un objeto.
   Al ser generico se le pueden pasar objetos tanto de tipo MechanicalKeyboards como Thinkpad sin distincion.
   Funciona de la siguiente manera:

- Si se realiza la llamda al metodo significa que StockAvailable devolvio true y existen suficientes materiales para crear una instancia del objeto.
- Se crea una copia local del diccionario en poder de la Factory y se le asigna el diccionario de Materials
- Se recorren las claves del diccionario de materiales del objeto
- Se recorren las claves del diccionario de materiales de Materials
- Se comparan las claves
- Si hay coincidencia se resta al diccionario Original (no la copia que esta siendo recorrida) la clave del diccionario del objeto.

4. Por ultimo dentro de la clase Factory.cs se implementa el metodo LoadMoreStock parametrizado como <T> que recibe un objeto T. Dicho metodo se utiliza para cargar mas materiales al diccionario de materiales.
   Funciona de la siguiente manera:

- Se crea una copia local del diccionario en poder de la Factory y se le asigna el diccionario de Materials
- Se recorren las claves del diccionario de materiales del objeto
- Se recorren las claves del diccionario de materiales de Materials
- Se comparan las claves
- Si hay coincidencia y el objeto es de tipo Keyboard se agrega stock basado en un random
- Si hay coincidencia y el objeto es de tipo Notebook se agrega stock basado en otro random
- El metodo se implemento de manera Generica para poder agregar STOCKS por separado, dependiendo del tipo de objeto que llegue por parametro.

Implementacion del Tema 18

Se crea la interfa Ifile la cual se parametriza como Generic.
Dicha interfaz crea la firma para un metodo Save, el cual recibe un string (path) y un T (objeto generico)
La misma se implementa en la clase Serializator.cs (pto. 19)

Implementacion del Tema 19

Se crea la clase Serializator.cs a la cual se parametriza como Generic e implementa una interfaz y servira para guardar objetos en un archivo XML.
Esta clase contiene un metodo Save que recibe por parametro un string correspondiente al path del archivo y un objeto T (Generic), permitiendo poder recibir tanto List<Product>, Keyboard, Notebook etc. Al ser generica, el tipo de objeto se resuelve en tiempo de ejecucion.

Dicho metodo hace uso del XmlTextWriter que obtiene un puntero al archivo, si el mismo no existe lo crea. Y del XmlSerializer encargado de serializar el objeto en el archivo.