SINGLETON Y ABSTRAC FACTORY



Autores:

ROBINSON FERNANDO RODRIGUEZ APONTE

MARÍA CAMILA PRECIADO ROJAS



UNIVERSIDAD PEDAGÓGICA Y TECNOLÓGICA DE COLOMBIA

ESCUELA DE SISTEMAS Y COMPUTACIÓN

FACULTAD DE INGENIERÍA

TUNJA, BOYACÁ

2017

**SINGLETON**

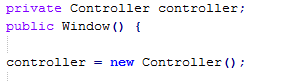
Es quizás el más sencillo de los patrones, es también uno de los patrones más conocidos y utilizados. Su propósito es asegurar que sólo exista una instancia de una clase.

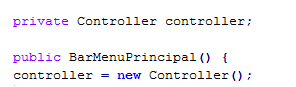
**Ejemplo**:

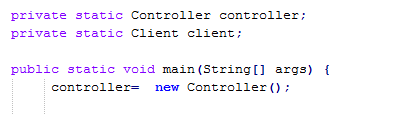
Existe un proyecto en el cual se requiere mejorar las buenas prácticas de programación, por ende, se va a aplicar el patrón singleton, el cual permite que solo se instancie en una vez la clase controlador (la cual contiene las acciones del programa) y en las otras solo obtenga esa instancia.

Problema

La clase Controller se instancia en todas las clases que necesitan acceso a ella, por ende, genera que esta instancia se haga muchas veces y sea deficiente. A continuación se explicara mediante imágenes.

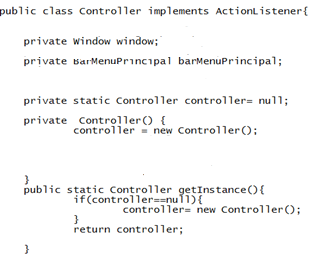






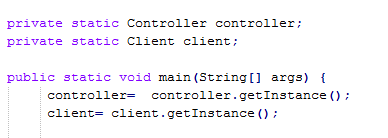
Solución

Para solucionar esto mal práctica de programación, se debe crear una sola instancia en la clase Controller la cual va a generar un acceso global para que el resto de clases puedan acceder a esta. A continuación, se ve en las imágenes más claro.

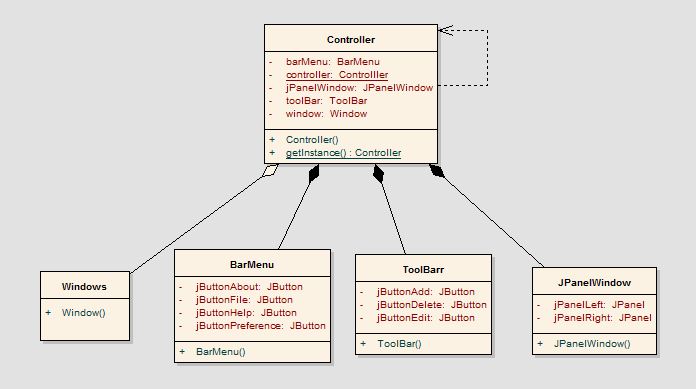








**DIAGRAMA DE CLASES DEL EJEMPLO SINGLETON**

****

**ABSTRAC FACTORY**

Este es un patrón de diseño creacional y que sirve para construir una jerarquía de clases.

Su función fundamental es encapsular una jerarquía de objetos y reducir el conjunto de conceptos con los que trabajamos.

**EJEMPLO:**

En un almacén de abarrotes se tiene una serie de bodegas en las cuales se ingresan productos del mismo tipo, pero que internamente se realizan procesos distintos; la clase abstracta es en este caso la BODEGA, creando métodos como ingresarProducto (Producto) por el cual recibe por parámetro producto, en este ejemplo se crean las clases concretas:

Class BODEGA\_refinados extends BODEGA(){}

Class BODEGA\_comunes extends BODEGA(){}

En las cuales por cada una se ingresan productos del mismo tipo pero internamente hacen procesos distintos; se crea la clase abstracta PRODUCTO

Abstrac class PRODUCTO(){}

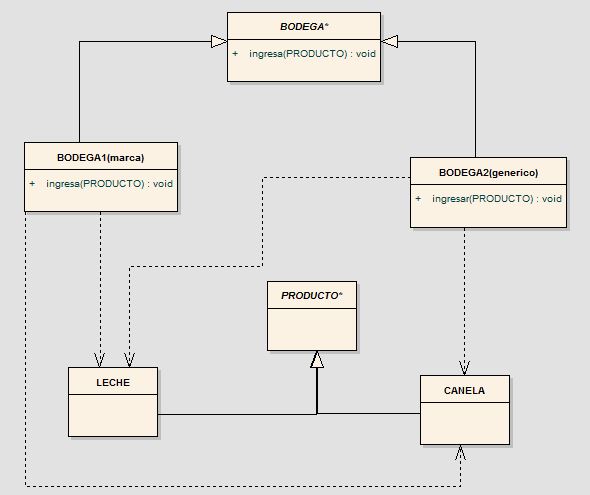
Y las clases concretas:

Class LECHE extends PRODUCTO(){}

Class CANELA extends PRODUCTO(){}

Por último se asocian los dos productos creados con las clases concretas BODEGA\_refinados, BODEGA\_comunes.

**DIAGRAMA DE CLASES DEL EJEMPLO DE ABSTRACT FACTORY**

****