28/ 08/ 2019

Lección 7

Continuación patrones de diseño

* Singleton: Patrón de diseño que permite restringir la instanciación de una clase.

Únicamente se puede instanciar una sola clase.

Public class Singleton{

Private static Singleton instance = null;

Private Singleton(){

------------------------------

---------------------------------

}

Public void doSomething(){

----------------------------------------

----------------------------------------

}

Public static Singleton getInstance(){ // Se puede poner sinchronized para hacer // una cola, conrolando la entrada a el método.

If(instance == null){

Instance = new Singleton();

}

Return instance;

}

}

}

Main:

Singleton.getInstance().doSomething();

SIngleton var = SIngleton.geInstance();

Builder: Delegar la construcción de una clase compleja a una clase complementaria.

Public class Phone{

Private camara;

Private radio;

Private touchID;

Private waterproof;

Public Phone(pCamara, pRadiom pTouch, pWaterproof){

//

}

}

Main:

Pone p1 = new Phone(camara, null, null, null);

Public class Phone{

Private camara;

Private radio;

Private touchID;

Private waterproof;

Public Phone(Builder builder){

// asignar parámetros

}

}

Public static class Builder{

Cámara:

Radio;

touchID;

waterproof;

public builder withCamara(pCamara){

this.camara = pCamara;

return this;

}

Public Phone build(){

----

----

Return new Phone(this);

}

}

Phone phone2 = new Phone(camara, null, null, null);

Phone p2 = new PhoneBuilder()

P2.withCamara();

P2.withRadio();

.build();