8/28/19

Continuando los patrones de diseño.

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

Únicamente se puede instanciar una sola clase.

public class Singleton {

private static Singleton instance = new Singleton();

private Singleton(){

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

}

public void doSomething(){

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

}

//Usar public static Synchronized para evitar problemas con los threads (pondrá las llamadas múltiples en cola, y las ejectura 1 por 1 ).

Public static Singleton getInstance(){

If(instance == null) {

Instance = new Singleton();

}

Return instance;

}

}

Main:

Singleton.getInstance().doSomething();

Singleton var = Singleton.getInstance();

-Builder: delegar la construcción de una clase compleja a una clase complementación.

public class Phone{

private cámara;

private radio;

private touch id;

private waterproof;

public Phone(pCamara, pRadio, pTouchID, pWaterproof){

this…..

}

Public Phone(pCamara){

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

}

Public Phone(Builder builder){

This.camara = builder.camara;

}

Public static class Builder{

Camara;

Radio;

TouchId;

Waterproof;  
 public Builder withCamara(pCamara){

This.camara = pCamara;

Return this;

}

Public Phone build(){

-----

Return new Phone(this);

}

}

Main:

Phone p1 = new Phone(null, null, touched, null);

Phone p2 = new Phone.Builder().withCamara(------).withRadio(------).build();