**Entrega ejercicios 25, 26, 27**

**Continúa con el proyecto de los anteriores ejercicios ahora se va a trabajar sobre las letras I y D de los principios SOLID, deberás:**

**Si no has creado interfaces ya, créalas siguiendo el principio de segregación de interfaz (I) el cual indica que las interfaces grandes se deben segregar en varias interfaces pequeñas. Si ya las has creado, modifícalas para que cumplan el principio.**

En una empresa tenemos una clase Manager(class Manager) que representa a la persona que gestiona a los trabajadores. Y tenemos 2 tipos de trabajadores, unos en el promedio (class Worker) y otros muy eficientes (class SuperWorker). Ambos tipos trabajan (work()) y necesitan un descanso diario para comer (eat()).

Ahora han llegado algunos robots a la empresa que trabajan, pero no comen con lo que las clases tendrían funciones que no estarían utilizando y con esta implementación estaría violando por tanto el principio de segregación de interfaces.

**// interface segregation principle – Mal implementado**

interface IWorker{

public void work();

public void eat();

}

class Worker implements IWorker{

public void work() {

// ....working

}

public void eat() {

// ...... eating in launch break

}

}

class SuperWorker implements IWorker{

public void work() {

//.... working much more

}

public void eat() {

//.... eating in launch break

}

}

class Manager {

IWorker worker;

public void setWorker(IWorker w) {

worker=w;

}

public void manage() {

worker.work();

}

}

Una solucíón seria separar la interface en dos e implementarlas en las clases según la funcionalidades que necesitemos.

La nueva clase Robot ya no está obligada a implementar el método eat.

**// interface segregation principle – bien implementado**

interface IWorker extends Feedable, Workable {}

interface IWorkable {

public void work();

}

interface IFeedable{

public void eat();

}

class Worker implements IWorkable, IFeedable{

public void work() {

// ....working

}

public void eat() {

//.... eating in launch break

}

}

class Robot implements IWorkable{

public void work() {

// ....working

}

}

class SuperWorker implements IWorkable, IFeedable{

public void work() {

//.... working much more

}

public void eat() {

//.... eating in launch break

}

}

class Manager {

Workable worker;

public void setWorker(Workable w) {

worker=w;

}

public void manage() {

worker.work();

}

}

**Deberás crear una conexión a base de datos aplicando el último principio de la inversión de dependencias (D).**

Se crea clase Usuario

package Codigo.Demo26Solid;  
  
public class Usuario {  
 String nombre;  
  
 public String getNombre() {  
 return nombre;  
 }  
  
 public void setNombre(String nombre) {  
 this.nombre = nombre;  
 }  
}

Se crea una interfaz DatabaseStore

package Codigo.Demo26Solid;  
  
public interface DatabaseStore {  
  
 void Guardar(Usuario usuario);  
}

Una clase DatabaseMySQL donde se implementa DatabaseStore

package Codigo.Demo26Solid;  
  
public class DatabaseMySQL implements DatabaseStore{  
 public void Guardar(Usuario usuario){  
  
 System.*out*.println("Soy Guardar() en DatabaseMySQL");  
 }  
}

Una clase DatabaseSQLite donde se implementa DatabaseStore

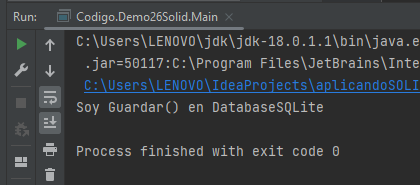
package Codigo.Demo26Solid;  
  
public class DatabaseSQLite implements DatabaseStore {  
 public void Guardar(Usuario usuario) {  
 System.*out*.println("Soy Guardar() en DatabaseSQLite");  
 }  
}

Main

package Codigo.Demo26Solid;  
  
public class Main {  
 public static void main(String []args){  
 Usuario usuario = new Usuario();  
 usuario.setNombre("Pepe");  
  
 DatabaseSQLite db = new DatabaseSQLite();  
 *GuardarEnBaseDeDatos*(db, usuario);  
 }  
  
 public static void GuardarEnBaseDeDatos(DatabaseStore db, Usuario usuario){  
 db.Guardar(usuario);  
  
 }  
}

En la clase Main según a que clase corresponde (DatabaseSQLite o DatabaseMySQL) el objeto que creamos guarda el usuario en una u otra base de datos.

En caso DatabaseSQLite



En caso DatabaseMySQL

package Codigo.Demo26Solid;  
  
public class Main {  
 public static void main(String []args){  
 Usuario usuario = new Usuario();  
 usuario.setNombre("Pepe");  
  
 DatabaseMySQL db = new DatabaseMySQL();  
 *GuardarEnBaseDeDatos*(db, usuario);  
 }  
  
 public static void GuardarEnBaseDeDatos(DatabaseStore db, Usuario usuario){  
 db.Guardar(usuario);  
  
 }  
}

