MEMORIA PRACTICA 4

---------------------------------------------------------EJERCICIO 1: ------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

CODIGO BaseDatos.java:

package pattern;

import java.util.ArrayList;

public class BaseDatos {

private static BaseDatos instance;//Objeto instancia para comprobar si sólo hay un objeto //creado

private ArrayList<String> Personas;

private BaseDatos(){ //Constructor privado para que solo se pueda acceder a el a //traves de la funcion getInstance();

Personas = new ArrayList<String>();

}

public void add(String name) {

Personas.add(name);

}

public static BaseDatos getInstance() { //funcion que comprueba si el programa ya ha // creado una instancia previa de este tipo de //objeto, si ya esta creado devuelve el anterior

if(instance==null) {

instance=new BaseDatos();

}

return instance;

}

}

CODIGO ExampleCode.java:

package pattern;

public class ExampleCode {

public static void main(String[] args) {

Client ClienteA,ClienteB;

BaseDatos BD1 = null , BD2 = null ;

// Sección modificable, sólo estas 2 líneas

BD1 = BaseDatos.getInstance();//se crea el objeto a traves de la funcion para solo //crear una instancia

BD2 = BaseDatos.getInstance();

ClienteA = new Client(BD1);

ClienteB = new Client(BD2);

//Comprobar que al implementar el Singleton, el resultado es true

System.out.println(ClienteA.getBD()==ClienteB.getBD());

}

}

---------------------------------------------------------EJERCICIO 2: ------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

CODIGO Adapter.java:

package pattern;

import java.util.ArrayList;

public class Adapter {

public Adapter(Service servicioA) {

// TODO Auto-generated constructor stub

}

//Funcion que se encarga de llamar a la funcion dataGeneration de service y saca el resultado de un //array estatico y lo guarda en un array list

public ArrayList<Integer> dataGeneration(int i, int j, int k) {

ArrayList<Integer> AL= new ArrayList<Integer>();

Service service = new Service();

Integer[] arr=service.dataGeneration(i, j, k);

for(int l=0;l<arr.length;l++) {

AL.add(arr[l]);

}

return AL;

}

//Funcion que se encarga de llamar a la funcion dataSelection de service

//primero guardando los datos en un array estatico para que la funcion pueda operar

//y luego vuelve a guardar todos los datos en un array list para que el codigo del cliente //pueda operar con ellos

public ArrayList<Integer> dataSelection(int i, int j, ArrayList<Integer> numbers) {

ArrayList<Integer> AL= new ArrayList<Integer>();

Integer arr[]= new Integer[numbers.size()];

Service service = new Service();

for(int l=0;l<numbers.size();l++) {

arr[l]=numbers.get(l);

}

arr=service.dataSelection(i, j, arr);

for(int l=0;l<arr.length;l++) {

AL.add(arr[l]);

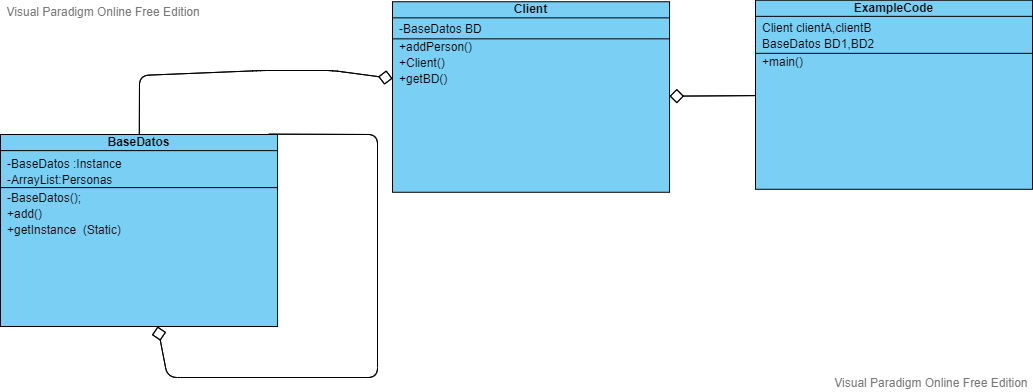
}

return AL;

}

}

---------------------------------------------------------UML1: ------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------



---------------------------------------------------------UML2: ------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

