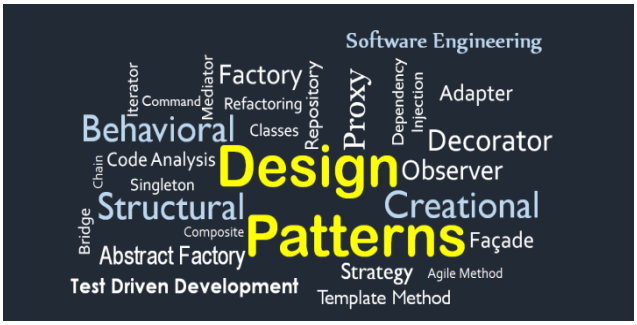

PATRONES DE DISEÑO ESTRUCTURALES

ETSII - URJC

Sebastián Ramiro Entierros García,
Adrián Martín Martín y
Álvaro Martínez Quiroga

Fundamentos del diseño del software

Índice

Contenido

Ilustraciones

Introducción y ficheros

En esta practica nos disponemos a desarrollar dos patrones de diseño, estos son el patrón Singleton y el patrón Factory Method. El paquete de la practica incluye:

- Código:
  - Adapter, hace uso del patrón de diseño Adapter.
  - Decorator, hace uso del patrón de diseño Decorator.
  - Facade, hace uso del patrón de diseño Facade.
- Diagrama UML del patrón Facade.
- Memoria.
- Archivos compilados .jar de cada programa.

Patrón Decorator

CriptoWriter

```
import java.io.*;

public class CriptoWriter extends BufferedWriter {

	// Constructor:
	public CriptoWriter(Writer writer) {
		super(writer);
	}
	// End Constructor.

	// Public Methods:
	public void write(char c) throws IOException {
		this.append((char) (c + 1));
	}

	@Override
	public void write(String s) throws IOException {
		for (int i = 0; i < s.length(); i++)
			this.write(s.charAt(i));
	}
	// End Public Methods.
}
```

CriptoReader

```
import java.io.*;

public class CriptoReader extends BufferedReader {
	// Constructor:
	public CriptoReader(Reader reader) {
		super(reader);
	}
	// End Constructor.
```

```
	// Public Method:
	@Override
	public int read() throws IOException {
		return super.read() - 1;
	}
	// End Public Method.
}
```

Test

```
import java.io.*;

public class Test {
	public static void main(String[] args) {

		String fileDirection = "prueba.txt";
		String texto1 = "grupo de contrasenyas: 12334-123442-111233";
		String texto2 = "grupo de ID_local: 123-321-213";

		// Creamos o abrimos el archivo y escribimos:
		try {
			FileWriter fw = new FileWriter(fileDirection);
			BufferedWriter bw = new BufferedWriter(fw);
			CriptoWriter outCripted = new CriptoWriter(bw);

			System.out.println("1.Se escribe de forma encripatada la
siguiente frase:");
			System.out.println(texto1);
			outCripted.write(texto1);
			outCripted.write("\n");
			System.out.println();

			outCripted.close();

			// Actualizamos el buffer para que concatene:
			bw = new BufferedWriter(new FileWriter(fileDirection,
true));
			outCripted = new CriptoWriter(bw);

			System.out.println("2. Se escribe de forma encriptada la
siguiente frase:");
			System.out.println(texto2);
			outCripted.write(texto2);
			System.out.println();

			outCripted.close();

		} catch (IOException iOException) {
			System.out.println("Excepcion de E/S: " + iOException);
		}

		// lectura de fichero
		String texto = new String();
		String textoEncript = new String();

		try {
```

```
                    // Leemos sin descifrar:
                    FileReader fr = new FileReader(fileDirection);
                    BufferedReader entrada = new BufferedReader(fr);
                    String s;
                    while ((s = entrada.readLine()) != null) {
                          texto += s;
                    }
                    entrada.close();

                    // Leemos el texto descifrado:
                    fr = new FileReader(fileDirection);
                    CriptoReader inCripted = new CriptoReader(fr);
                    char c;

                    while ((c = (char) inCripted.read()) != '\ufffe') {
                          textoEncript += c;
                    }

                    inCripted.close();

              } catch (FileNotFoundException fileNotFoundException) {
                    System.out.println("Archivo no encontrado: " +
fileNotFoundException);
              } catch (IOException iOException) {
                    System.out.println("Excepcion de E/S: " + iOException);
              }

              // Por ultimo vemos los textos:

              System.out.println("-----------------------------------------------
------------");
              System.out.println();
              System.out.println("Textos originales: ");
              System.out.println(texto1);
              System.out.println(texto2);
              System.out.println();
              System.out.println("Textos Sin descifrar:");
              System.out.println(texto);
              System.out.println("Texto descifrado:");
              System.out.println(textoEncript);
        }
}
```

Diagrama UML

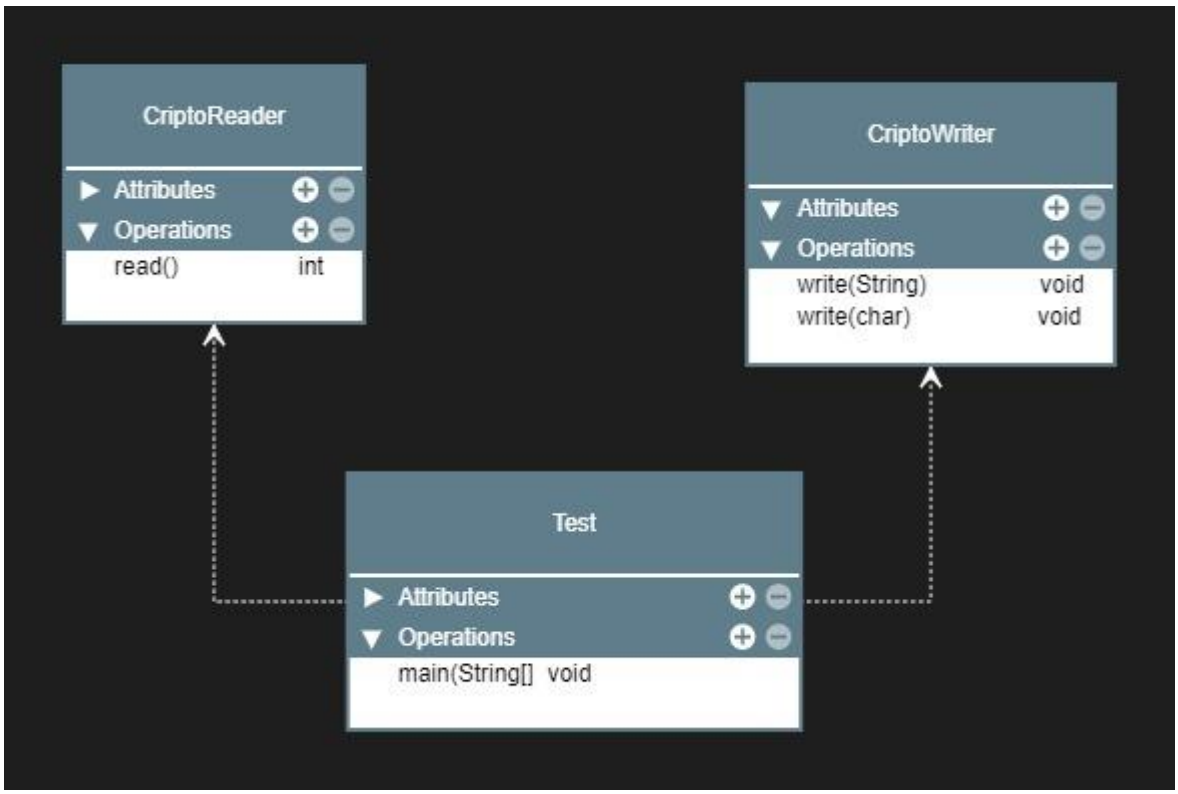

Ilustración 1: diagrama patrón Decorator

Patrón Facade

Fachada

```
public class Fachada {

	private Configura IConfigura;
	private Presenta IPresenta;
	private Imprime IImprime;

	public Fachada() {
		this.IConfigura = new ConfiguraClass();
		this.IPresenta = new PresentaClass();
		this.IImprime = new ImprimeClass();
	}

	public void imprimeLujo(String texto) {
		IConfigura.setTipoHoja("A4");
		IConfigura.setColor(true);
		IPresenta.setOrdena(true);
		IPresenta.setGrapa(true);
		IImprime.setTexto(texto);
		IImprime.imprimirDocumento();
	}

	public void imprimeBorrador(String texto) {
		IConfigura.setTipoHoja("B5");
		IConfigura.setColor(false);
		IConfigura.setBorrador(true);
		IImprime.setTexto(texto);
		IImprime.imprimirDocumento();
	}

	public Configura getIConfigura() {
		return IConfigura;
	}

	public void setIConfigura(Configura IConfigura) {
		this.IConfigura = IConfigura;
	}

	public Presenta getIPresenta() {
		return IPresenta;
	}

	public void setIPresenta(Presenta IPresenta) {
		this.IPresenta = IPresenta;
	}

	public Imprime getIImprime() {
		return IImprime;
	}

	public void setIImprime(Imprime IImprime) {
		this.IImprime = IImprime;
	}
```

```
}
```

Cliente

```
public class Cliente {
	public static void main(String[] args) {

		Fachada fachada = new Fachada();

		fachada.imprimeLujo("ESTO ES DE LUJO");
		System.out.println("-----------------------------");
		fachada.imprimeBorrador("ESTO ES UN BORRADOR");
		System.out.println("-----------------------------");
		fachada.getIImprime().setTexto("ESTO ES UN TEXTO");
		fachada.getIPresenta().setGrapa(true);
		fachada.getIConfigura().setTipoHoja("A3");
		System.out.println("-----------------------------");
	}
}
```

Patrón Adapter

Adaptador

```
import java.util.*;

public class AdapterSeleccionDirecta implements Objetivo {

	private SeleccionDirecta selDir;

	public AdapterSeleccionDirecta(SeleccionDirecta selDir) {
		this.selDir = selDir;
	}

	@Override
	public void ordena(ArrayList<Integer> a) {

		int[] v = new int[a.size()];

		// Copiamos todos los elementos a una array:
		Iterator it = a.iterator();
		int i = 0;
		while (it.hasNext()) {
			v[i] = (int) it.next();
			i++;
		}

		// Ordenamos el array:
		selDir.ordena(v);

		// Se vacia y se rellena el arrayList ya ordenado:
```

```
            a.clear();
            for (i = 0; i < v.length; i++) {
                  a.add(v[i]);
            }
      }
}
```

Cliente

```
import java.util.ArrayList;
import java.util.Iterator;

public class Cliente {
      public static void main(String[] args) {

            SeleccionDirecta sD = new SeleccionDirecta();
            AdapterSeleccionDirecta adSD = new
AdapterSeleccionDirecta(sD);
            MergeSort mS = new MergeSort();

            int[] v = inicializarVector();

            ArrayList<Integer> al = inicializarArrayList(v);

            System.out.println("Se crea un vector con estos datos");
            imprimirArray(v);
            System.out.println();
            System.out.println();

            System.out.println("Se crea un array list");
            imprimirArrayList(al);
            System.out.println();
            System.out.println();

            System.out.println("Se ordena el vector con seleccion
directa");
            sD.ordena(v);
            imprimirArray(v);
            System.out.println();
            System.out.println();

            System.out.println("Se ordena el vector con mergesort");
            mS.ordena(al);
            imprimirArrayList(al);
            System.out.println();
            System.out.println();
```

```
            System.out.println("Se recrean los vectores para volver a
desordenarlos");
            System.out.println();
            v = inicializarVector();
            al = inicializarArrayList(v);

            System.out.println("Se muestra el nuevo vector");
            imprimirArray(v);
            System.out.println();
            System.out.println();

            System.out.println("Se muestra el array list para ver los
nuevos valores tambien");
            imprimirArrayList(al);
            System.out.println();
            System.out.println();

            System.out.println("Se ordena el ArrayList mediante seleccion
directa usando el adaptador:");
            adSD.ordena(al);
            imprimirArrayList(al);
            System.out.println();
      }

      private static int[] inicializarVector() {

            int[] v = { 2, 3, 144, 321, 8, 2300, 6, 5, 10, 456, 988, 444
};

            return v;
      }

      private static ArrayList inicializarArrayList(int[] v) {

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

            for (int i = 0; i < v.length; i++) {
                  al.add(v[i]);
            }

            return al;
      }

      private static void imprimirArray(int[] v) {

            for (int i = 0; i < v.length; i++) {
                  System.out.print(v[i] + " ");
            }
      }

      private static void imprimirArrayList(ArrayList l) {
```

```
            Iterator it = l.iterator();

            while (it.hasNext()) {
                  System.out.print((int) it.next() + " ");
            }
      }
}
```

Diagrama UML

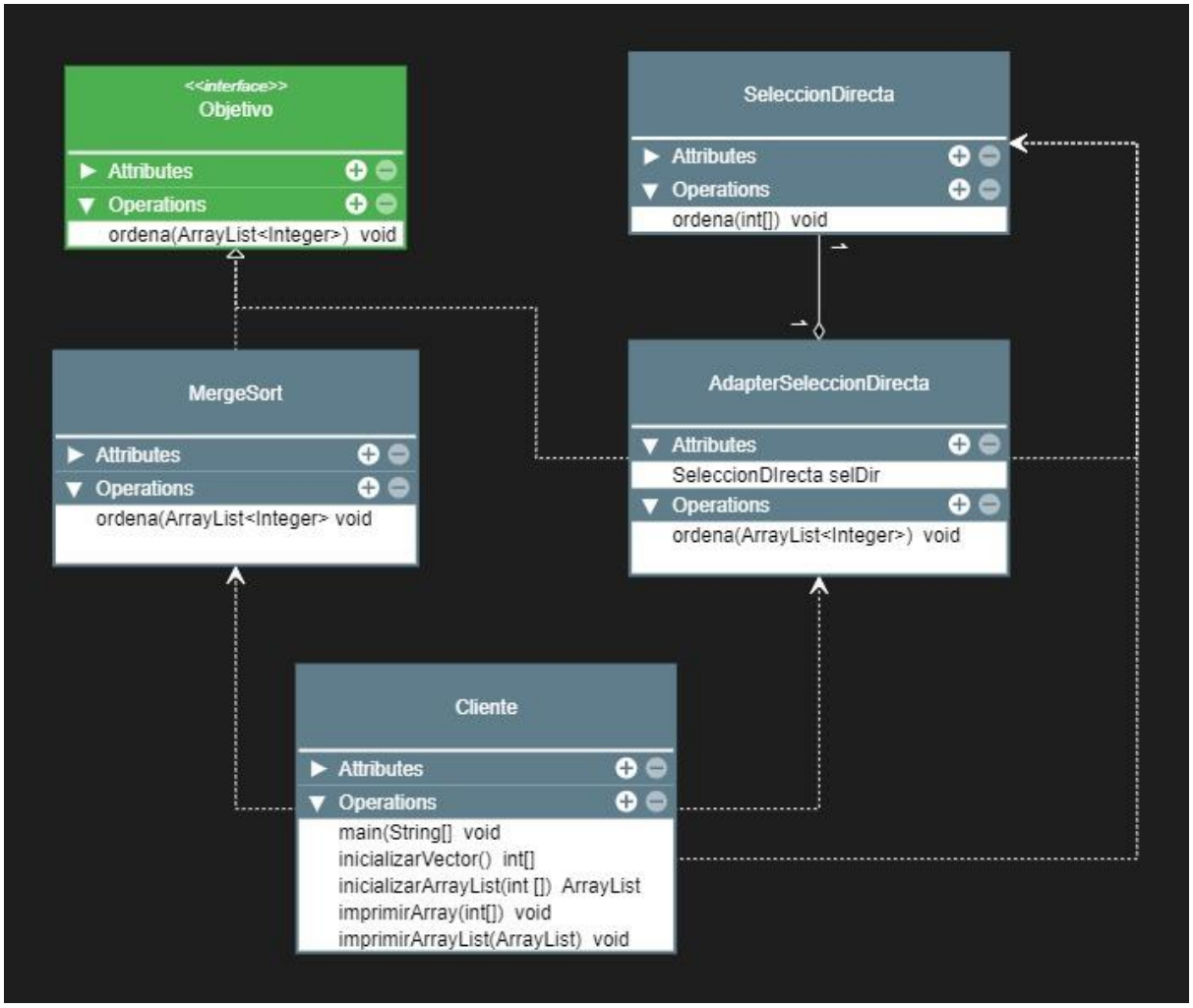

Ilustración 2: diagrama patrón Adapter

Conclusiones de la práctica

Consideramos esta practica muy interesante para tener y comprender la idea de dividir código. Gracias a esta práctica, al igual que la anterior, hemos sido capaces de recapacitar en nuestra anterior forma

de programar sin seguir ningún patrón de diseño, algo que resultaba en código no mantenibles en para el futuro de estos.

Referencias

- Temario del aula virtual.