Programación Orientada a Objetos II

Universidad Nacional de Quilmes

Trabajo Práctico – Template Method y Adapter

Objetivo: Comprender el patrón de diseño Template Method y el patrón Adapter. Discutir alternativas de diseño e implementación.

Ejercicio 1 - Lea el pattern Template Method y...

1. Utilizando las ideas propuestas por el pattern y responda:
    a. ¿Dónde se define el esqueleto del algoritmo?
    b. ¿Se puede redefinir el esqueleto?
    c. ¿Qué es lo que se puede redefinir?
    d. ¿Qué es un hook method?

Ejercicio 2 - Sueldos recargado

Una empresa realiza periódicamente los pagos de sueldos y cargas de cada uno de sus empleados. En la empresa los empleados están organizados en tres tipos: Temporarios, Pasantes y de Planta. El sueldo de un empleado Temporario está determinado por el pago de $5 por hora que trabajó más el sueldo básico que es de $1000, además se le paga $100 si posee hijos y/o está casado. A los Pasantes se les paga $40 las horas trabajadas en el mes. Por último a los empleados de Planta se les paga un sueldo básico de $ 3000 y un plus por cada hijo que posea de $ 150 cada hijo. Por otro lado, de cada sueldo se deben descontar en conceptos de aportes y obra social un 13 % del sueldo.

Modele la jerarquía de Empleados de forma tal que cualquier empleado puede responder al mensaje #sueldo. Utilice template method para su solución e indique los roles participantes.

Ejercicio 3 - Elementos similares

Es necesario programar un filtro para que a partir de una colección de páginas de Wikipedia retorne aquellas que sean similares a una particular. Para ello es necesario implementar una jerarquía de clases filtros en las que se pueda definir la similitud en función de diferentes heurísticas. Las heurísticas están definidas en función a atributos que poseen las páginas de Wikipedia. Para esto, una página de Wikipedia cuenta con la interfaz WikipediaPage que posee el siguiente protocolo:

```
String getTitle(); /*retorna el título de la página.*/

List<WikipediaPage> getLinks(); /*retorna una Lista de las páginas de
Wikipedia con las que se conecta.*/
```

```
Map<String, WikipediaPage> getInfobox(); /*retorna un Map con un valor en texto y la pagina que describe ese valor que aparecen en los infobox de la página de Wikipedia.*/
```


En esta oportunidad ud debe implementar tres tipos de filtros: MismaLetraInicial, LinkEnComun, PropiedadEnComun. La forma en que funcionan las clases filtros es mediante el método

```
public List<WikipediaPage> getSimilarPages(WikipediaPage page,
List<WikipediaPage) wikipedia)
```

La lógica de cada filtro es la siguiente:

MismaLetraInicial retorna como páginas similares aquellas que poseen la misma primera letra en el comienzo del título, por ejemplo "La Plata" es similar con "Lucas Art" y "Lobo".

LinkEnComun retorna como páginas similares aquellas que posean al menos un link a una página en común, por ejemplo si la página de "Gimnasia y Esgrima La Plata" tiene un link a la página "La Plata" y la página "Buenos Aires" tiene un link a "La Plata" esas páginas serian similares.

PropiedadEnComun, retorna aquellas páginas que poseen alguna propiedad del infobox en común, por ejemplo si la página de una persona tiene la propiedad "birth_place" y otra página posee también la propiedad "birth_place" serian similares mutuamente. En este caso, no importa que el valor de la propiedad sea diferente.

1. Realice un diagrama de clases que modele el problema planteado. 2. Realice un diagrama de secuencia para el caso de usar un filtro de tipo **MismaLetraInicial** para obtener las paginas similares a "Bernal" asumiendo que Wikipedia posee como paginas "Bernal", "Quilmes", "Buenos Aires" 3. Programe el test de unidad utilizando JUnit para las tres clases involucradas. 4. Programa la funcionalidad completa en Java.

Ejercicio 4 - Llamadas telefónicas

A partir de las siguientes líneas de código identifique estos conceptos, si es posible:

- Template Method.
- Operaciones primitivas.
- Operaciones concretas.
- Hook Method.

```
public abstract class LlamadaTelefonica {
	private int tiempo;
	private int horaDelDia;

	public LlamadaTelefonica(int tiempo, int horaDelDia){
		this.tiempo=tiempo;
		this.horaDelDia=horaDelDia;
	}
```

```
        public int getTiempo(){
                return this.tiempo;
        }
        public int getHoraDelDia(){
                return this.horaDelDia;
        }
        public abstract boolean esHoraPico();
```



```
        public float costoFinal(){
                if(this.esHoraPico()){
                        return this.costoNeto()*1.2f*this.getTiempo();
                }else{
                        return this.costoNeto()*this.getTiempo();
                }
        }
        public float costoNeto(){
                return this.getTiempo()*1;
        }
}

public classLlamadaDescuento extends LlamadaTelefonica{
        public LlamadaDescuento(int tiempo, int horaDelDia) {
                super(tiempo, horaDelDia);
        }
        @Override
        public boolean esHoraPico() {
                return false;
        }
        @Override
        public float costoNeto(){
                return this.getTiempo()*0.95f;
        }
}
```

Ejercicio 5 - Ayudando al soberano

Un banco tiene dos tipos de cuentas bancarias. Cajas de ahorro y Cuentas corrientes. Ambas cuentas se manejan en forma similar excepto para la extracciones de dinero. En las caja de ahorro es posible realizar la extracción solamente si el saldo de la cuenta es superior o igual a la cantidad de dinero que quiero extraer y no supera el límite por extracción, si esto sucede, entonces realiza efectivamente la extracción (actualizando el saldo) y luego registra que se hizo una extracción simplemente agregando un texto a una colección. Por otra parte, las cuentas corrientes pueden realizar una extracción solamente cuando la cantidad de dinero a extraer no supera el saldo más el límite en rojo que posee la cuenta. Si esto ocurre entonces realiza la extracción y luego registra que se hizo una extracción simplemente agregando un texto a una colección.

Un compañero del curso, realizó el ejercicio de la siguiente manera:

```
import java.util.ArrayList;
import java.util.List;

public abstract class CuentaBancaria {
    private String titular;
    private int saldo;
    private List<String> movimientos;


    public CuentaBancaria(String titular){
```



```
        this.titular=titular;
        this.saldo=0;
        this.movimientos=new ArrayList<String>();
    }
        public String getTitular(){
        return this.titular;
    }
    public int getSaldo(){
        return this.saldo;
    }
    protected void setSaldo(int monto){
        this.saldo=monto;
    }
    public void agregarMovimientos(String movimiento){
        this.movimientos.add(movimiento);
    }
    public abstract void extraer(int monto);
}

public class CuentaCorriente extends CuentaBancaria {
    private int descubierto;

    public CuentaCorriente(String titular, int descubierto){
        super(titular);
        this.descubierto=descubierto;
    }
    public int getDescubierto(){
        return this.descubierto;
    }
    @Override
    public void extraer(int monto) {
        if(this.getSaldo()+this.getDescubierto()>=monto){
              this.setSaldo(this.getSaldo()-monto);
            this.agregarMovimientos("Extraccion");
        }
    }
}

public class CajaDeAhorro extends CuentaBancaria {
```

```
    private int limite;

    public CajaDeAhorro(String titular, int limite){
        super(titular);
        this.limite=limite;
    }
    public int getLimite(){
        return this.limite;
    }
    @Override
    public void extraer(int monto) {
        if(this.getSaldo()>=monto && this.getLimite()>=monto){
            this.setSaldo(this.getSaldo()-monto);
            this.agregarMovimientos("Extraccion");
        }
    }
}
```


El cual compila y retorna los resultados esperados. Su compañero se da cuenta que hay código repetido y además no utilizar template method, el cual es indispensable para que apruebe la entrega.

Su objetivo es el siguiente:

1. Debe realizar los cambios necesarios en el código para que el método extraer sea un template method. (Escriba solamente los cambios, el resto se asume que queda intacto)
2. Analice en el código resultante los elementos que indica Gamma et. al que pueden aparecer en su solución, por ejemplo: Clase abstracta, clase concreta, cual es el template method, cuales son las operaciones primitivas, cuales son las operaciones concretas, cuales los hook methods.

Importante: No necesariamente aparecen todos los elementos que lista Gamma, ud. debe explicar solamente aquellos que están en su solución.

Ejercicio 6 – Contestar y justificar las respuestas

1. Existe más de un tipo de adaptadores, mencione y explique cada uno de ellos.
2. ¿En qué se diferencia un tipo de adaptador del otro?
3. ¿Se pueden utilizar ambas alternativas de implementación del patrón en java? ¿Justifique la respuesta?
4. Ver la interface Enumeration de java y la clase Vector, preste atención a que dicha clase contiene un método "elements()". Explique cómo funciona el patrón adapter y cuáles son los roles de los participantes entre la interface y clase mencionada. Mencione qué tipo de implementación del patrón se utiliza.
5. Realice el mismo análisis del punto 3, pero con la interface Iterator, la clase ArrayList (preste atención al método "iterator()"). Muestre un ejemplo de funcionamiento y especifique los roles de cada participante.

6. Implemente un Adaptador, que adapte un Iterator a un Enumeration. Escribir código que utilice dicha implementación adaptando un ArrayList.

Ejercicio 7 – Palabras Ordenadas

Se desea visualizar una lista de palabras ordenadas alfabéticamente, para realizar esta tarea ya se dispone de dos clases. La clase Ventana es la encargada de visualizar la lista de palabras, sin orden alfabético, por otro lado se dispone de la clase ListaDePalabrasOrdenadas, esta mantiene una lista de palabras ordenadas alfabéticamente y tiene el comportamiento para agregar nuevas palabras.
El objetivo es que Ventana pueda recibir una "instancia adaptada" de ListaDePalabrasOrdenadas.


```
public class Ventana extends JFrame{

	public Ventana(DefaultListModel listModel){
		this.setLayout(new BoxLayout(this.getContentPane(),BoxLayout.Y_AXIS))
		this.setBounds(new Rectangle(180,180));
		this.setVisible(true);
		JLabel label=new JLabel("Lista de palabras");
		label.setBounds(new Rectangle(40,40));
		this.add(label);
		listModel.addElement("casa");
		listModel.addElement("arbol");
		listModel.addElement("perro");
		listModel.addElement("telefono");
		listModel.addElement("brazo");
		JList lista=new JList(listModel);
		lista.setBounds(new Rectangle(110,160));
		this.add(lista);
	}

	public static void main(String[] args) {
		new Ventana(new DefaultListModel);
	/*esta línea puede modificarse luego de resolver el ejercicio.*/
	}
}

public class ListaDePalabrasOrdenadas {
	private List<String>palabras;
	private final Comparator comparador;

	public ListaDePalabrasOrdenadas(){
		this.palabras=new LinkedList<String>();
```

```
                this.comparador=newComparator(){

                        @Override
                        public int compare(Object palabraA, Object palabraB) {
                                String a=((String)palabraA);
                                String b=(((String)palabraB));
                                return a.compareToIgnoreCase(b);
                        }
                };
        }
        public void agregarPalabra(String palabra){
                this.palabras.add(palabra);
                Collections.sort(this.palabras, this.comparador);
        }
        public Integer cantidadDePalabras(){
                return this.palabras.size();
        }
        public String getPalabraDePosicion(int posicion){
                return this.palabras.get(posicion);
        }
}
```

Actividad:

1. Analizar el código de ambas clases.
2. Proponer una solución en java.

Ejercicio 8 – Fechas

Los alumnos de la universidad programaron el sistema que permite detectar los vencimientos de las materias. El sistema basa sus fechas en una clase llamada FechaAlumnos que implementa la siguiente interfaz:

```
package fechas;
public interface IFecha {
        public IFecha restarDias(int dias);
        public boolean antesDeAhora();
        public boolean antesDe(IFecha fecha);
        public boolean despuesDeAhora();
        public boolean despuesDeFecha(IFecha fecha);
        public int dia();
        public int mes();
        public int anio();
}
```

Lamentablemente, luego de mucho tiempo se dieron cuenta que existe la clase DateTime del proyecto joda-time que implementa un comportamiento similar y muy bien realizado, a diferencia de la implementación que hicieron los alumnos anteriormente y tenía algún error.

Realice un adaptador para que el sistema de vencimientos de materia pueda utilizar la clase DateTime del proyecto joda-time en cualquier lugar donde se utiliza un IFecha.

Puede descargar las librerías necesarias para utilizar las clases del proyecto joda-time desde: http://www.joda.org/joda-time/

Ejercicio 9 – Colecciones

- Ver la interface Enumeration de java y la clase Vector, preste atención a que dicha clase contiene un método "elements()". Explique cómo funciona el patrón adapter y cuáles son los roles de los participantes entre la interface y clase mencionada. Mencione qué tipo de implementación del patrón se utiliza.

- Realice el mismo análisis del punto 1.3, pero con la interface Iterator, la clase ArrayList (preste atención al método "iterator()"). Muestre un ejemplo de funcionamiento y especifique los roles de cada participante.

- Implemente un Adaptador, que adapte un Iterator a un Enumeration. Escribir código que utilice dicha implementación adaptando un ArrayList.