



INSTITUTO TECNOLÓGICO DE OAXACA

Ingeniería en Sistemas Computacionales

Diseño E Implementación De Software Con Patrones

Alumnos:

Méndez Mendoza Luisa Michel

Pérez Carrasco Samuel

López García Lourdes Gloria

Girón Pacheco Fernando

Barbosa Santiago Mario Alberto

Unidad 4

Espinosa Pérez Jacob

Horario: 07:00 am- 08:00 am

8SC

PATRON INTERPRETER

El patrón de diseño Intérprete es un patrón de diseño conductual que define una forma de interpretar y evaluar la gramática o las expresiones lingüísticas. Proporciona un mecanismo para evaluar oraciones en un idioma mediante la representación de su gramática como un conjunto de clases. Cada clase representa una regla o expresión en la gramática, y el patrón permite que estas clases se compongan jerárquicamente para interpretar expresiones complejas.

Ventajas

Flexibilidad: Permite interpretar nuevos tipos de expresiones.

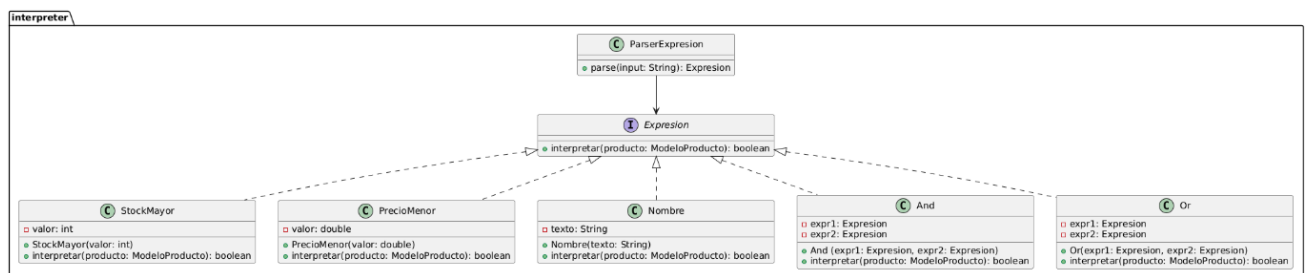
Extensibilidad: Fácil de ampliar y modificar el lenguaje o gramática.

Desventajas

Complejidad: Puede volverse complejo si la gramática del lenguaje es complicada.

Rendimiento: Interpretar expresiones puede ser menos eficiente que otros métodos de procesamiento.

Diagrama UML





Interprete

- Analizador.java
- And.java
- Expresion.java
- Nombre.java
- Or.java
- PrecioMenor.java
- StockMayor.java

Primero creamos nuestra interfaz base, esta interfaz define el contrato es decir que cualquier clase que implemente esta interfaz debe interpretar o evaluar si un producto cumple una condición.

```
1  /*
2  * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/licen...
3  * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class...
4  */
5  package Interprete;
6
7  import modelo.ModeloProducto;
8
9  /**
10   *
11   * @author Garaudy
12   */
13  public class StockMayor implements Expresion {
14      private int valor;
15
16      public StockMayor(int valor) {
17          this.valor = valor;
18      }
19
20      public boolean interpretar(ModeloProducto producto) {
21          return producto.getStockProducto() > valor;
22      }
23  }
```

Evalúa si el stock de un producto es mayor al valor indicado

Evalúa si el precio de un producto es menor

```
...va PrecioMenor.java x Nombre.java x And.java x Or.java x An
Source History
1  /*
2   * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses
3   * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/
4   */
5   package Interprete;
6
7   import modelo.ModeloProducto;
8
9   /**
10    *
11    * @author Garaudy
12    */
13   public class PrecioMenor implements Expresion {
14       private double valor;
15
16       public PrecioMenor(double valor) {
17           this.valor = valor;
18       }
19
20       public boolean interpretar(ModeloProducto producto) {
21           return producto.getPrecioProducto() < valor;
22       }
23   }
```

Este evalúa si el nombre de un producto contiene cierta palabra que se coloque en el buscador



```

1  /*
2  * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to
3  * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this
4  */
5  package Interprete;
6
7  import modelo.ModeloProducto;
8
9  /**
10   *
11   * @author Garaudy
12   */
13  public class Nombre implements Expression {
14      private String texto;
15
16      public Nombre(String texto) {
17          this.texto = texto.toLowerCase();
18      }
19
20      public boolean interpretar(ModeloProducto producto) {
21          return producto.getNombreProducto().toLowerCase().contains(":" + texto);
22      }
23  }
24

```

CLASES CON EXPRESIONES COMBINADAS

Estas clases permites usar lógica compuesta (AND, OR)

Devuelve true si ambas condiciones son verdaderas

```

1  /*
2  * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to
3  * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this
4  */
5  package Interprete;
6
7  import modelo.ModeloProducto;
8
9  /**
10   *
11   * @author Garaudy
12   */
13  public class And implements Expression {
14      private Expression expr1;
15      private Expression expr2;
16
17      public And(Expression expr1, Expression expr2) {
18          this.expr1 = expr1;
19          this.expr2 = expr2;
20      }
21
22      public boolean interpretar(ModeloProducto producto) {
23          return expr1.interpretar(producto) && expr2.interpretar(producto);
24      }
25  }

```

Devuelve true si al menos una condición es verdadera



```

...va PrecioMenor.java x Nombre.java x And.java x Or.java x Analizador.java x ControlA
Source History
1  /*
2   * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt
3   * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit th
4   */
5   package Interprete;
6
7   import modelo.ModeloProducto;
8
9   /**
10    *
11    * @author Garaudy
12    */
13   public class Or implements Expression {
14       private Expression expr1;
15       private Expression expr2;
16
17       public Or(Expression expr1, Expression expr2) {
18           this.expr1 = expr1;
19           this.expr2 = expr2;
20       }
21
22       public boolean interpretar(ModeloProducto producto) {
23           return expr1.interpretar(producto) || expr2.interpretar(producto);
24       }
25   }
26

```

Esta clase convierte el texto ingresado por el usuario que implementan la interfaz Expresión.

```

...va PrecioMenor.java x Nombre.java x And.java x Or.java x Analizador.java x ControlAlquiler.java x login.java
Source History
5   package Interprete;
6
7   /**
8    *
9    * @author Garaudy
10   */
11   public class Analizador{
12       public static Expression parse(String input) {
13           input = input.toLowerCase().trim();
14
15           if (input.contains(" and ")) {
16               String[] partes = input.split(" and ");
17               return new And( parse(partes[0]), parse(partes[1]));
18           } else if (input.contains(" or ")) {
19               String[] partes = input.split(" or ");
20               return new Or( parse(partes[0]), parse(partes[1]));
21           } else if (input.contains("stock >")) {
22               int valor = Integer.parseInt(input.replaceAll("^[^0-9]", ""));
23               return new StockMayor(valor);
24           } else if (input.contains("precio <")) {
25               double valor = Double.parseDouble(input.replaceAll("^[^0-9]", ""));
26               return new PrecioMenor(valor);
27           } else if (input.startsWith("nombre ")) {
28               String texto = input.substring(7).trim();
29               return new Nombre(texto);
30           } else {
31               return new Nombre(texto: input.trim());
32           }
33       }
34   }

```

```

367 public void BuscarProductoAvanzado(String consulta, JTable tablaProducto) {
368     configuracion.Conexion conexion = new configuracion.Conexion();
369     DefaultTableModel modelo = new DefaultTableModel();
370
371     modelo.addColumn( columnName: "ID");
372     modelo.addColumn( columnName: "Nombre");
373     modelo.addColumn( columnName: "Precio");
374     modelo.addColumn( columnName: "Stock");
375
376     tablaProducto.setModel( dataModel: modelo);
377
378     try {
379         String sql = "SELECT * FROM producto";
380         PreparedStatement ps = conexion.estableceConexion().prepareStatement(sql);
381         ResultSet rs = ps.executeQuery();
382
383         Interprete.Expression expression = Interprete.Analizador.parse( input: consulta);
384
385         while (rs.next()) {
386             modelo.ModeloProducto producto = new modelo.ModeloProducto();
387             producto.setIdProducto( idProducto: rs.getInt( columnLabel: "idproducto"));
388             producto.setNombreProducto( nombreProducto: rs.getString( columnLabel: "nombre"));
389             producto.setPrecioProducto( precioProducto: rs.getDouble( columnLabel: "precioProducto"));
390             producto.setStockProducto( stockProducto: rs.getInt( columnLabel: "stock"));
391
392             if (expression.interpretar(producto)) {
393                 modelo.addRow(new Object[] {
394                     producto.getIdProducto(),
395                     producto.getNombreProducto(),
396                     producto.getPrecioProducto(),
397                     producto.getStockProducto()
398                 });
399             }
400         }
401     } catch (SQLException e) {
402         e.printStackTrace();
403     }
404 }

```

Evalúa cada producto con la Expresión, solo los que devuelven true se muestran en la tabla de productos.

PRUEBAS

stock > 200 and precio < 20

Productos Disponibles

Buscador:

Click para Seleccionar

ID	Nombre	Precio	Stock
1	SILLAS ME...	5.0	1000
2	SILLAS PL...	5.0	1000
3	SILLAS MA...	15.0	1000
5	SILLAS AC...	15.0	1000
6	SILLAS INF...	5.0	1000

Consulta Rapida de Productos

Busqueda avanzada

silla or stock < 100

Productos Disponibles
Buscador:
Click para Seleccionar

ID	Nombre	Precio	Stock
1	SILLAS M...	5.0	1000
2	SILLAS PL...	5.0	1000
3	SILLAS MA...	15.0	1000
4	SILLAS TI...	20.0	1000
5	SILLAS AC...	15.0	1000
6	SILLAS IN...	5.0	1000

Consulta Rapida de Productos

mesa

Productos Disponibles
Buscador:
Click para Seleccionar

ID	Nombre	Precio	Stock
7	MESAS RE...	70.0	100
8	MESAS CL...	70.0	100
9	MESAS CU...	70.0	100
10	MESAS INF...	70.0	100

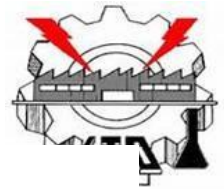
Consulta Rapida de Productos

stock > 100 Productos con más de 100 unidades disponibles

Productos Disponibles
Buscador:
Click para Seleccionar

ID	Nombre	Precio	Stock
11	MANTELE...	40.0	150
13	PLATOS E...	5.0	200
14	PLATOS / ...	5.0	200
15	PLATOS P...	5.0	200
16	TAZONES ...	5.0	200
17	TAZAS.CH	5.0	200

Consulta Rapida de Productos



precio < 30 Productos que cuesten menos de 30

Productos Disponibles
Buscador:
Click para Seleccionar

ID	Nombre	Precio	Stock
1	SILLAS M...	5.0	1000
2	SILLAS PL...	5.0	1000
3	SILLAS MA...	15.0	1000
4	SILLAS TL...	20.0	1000
5	SILLAS AC...	15.0	1000
6	SILLAS IN...	5.0	1000

Consulta Rapida de Productos

silla Productos cuyo nombre contiene la palabra "silla"

Productos Disponibles
Buscador:
Click para Seleccionar

ID	Nombre	Precio	Stock
1	SILLAS M...	5.0	1000
2	SILLAS PL...	5.0	1000
3	SILLAS MA...	15.0	1000
4	SILLAS TL...	20.0	1000
5	SILLAS AC...	15.0	1000
6	SILLAS IN...	5.0	1000

Consulta Rapida de Productos

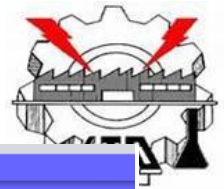
stock > 100 and precio < 20 Productos con stock alto y precio bajo

Productos Disponibles
Buscador:
Click para Seleccionar

ID	Nombre	Precio	Stock
2	SILLAS PL...	5.0	1000
3	SILLAS MA...	15.0	1000
5	SILLAS AC...	15.0	1000
6	SILLAS IN...	5.0	1000
13	PLATOS E...	5.0	200
14	PLATOS / ...	5.0	200

Consulta Rapida de Productos

mesa or stock < 50 Mesas o productos con poco stock



Productos Disponibles

Buscador:

Click para Seleccionar

ID	Nombre	Precio	Stock
7	MESAS RE...	70.0	100
8	MESAS CI...	70.0	100
9	MESAS CU...	70.0	100
10	MESAS INF...	70.0	100

Consulta Rapida de Productos

mesa or stock < 50

Busqueda avanzada

plato and precio < 10

Productos Disponibles

Buscador:

Click para Seleccionar

ID	Nombre	Precio	Stock
13	PLATOS E...	5.0	200
14	PLATOS / T...	5.0	200
15	PLATOS P/ ...	5.0	200

Consulta Rapida de Productos

plato and precio < 10

Busqueda avanzada