Centro Universitário UniBH
Ciência da Computação
Práticas de Programação
Professor: Lucas Schmidt

Aula Prática - 15 Padrões de Comportamento - Interpreter

Intenção

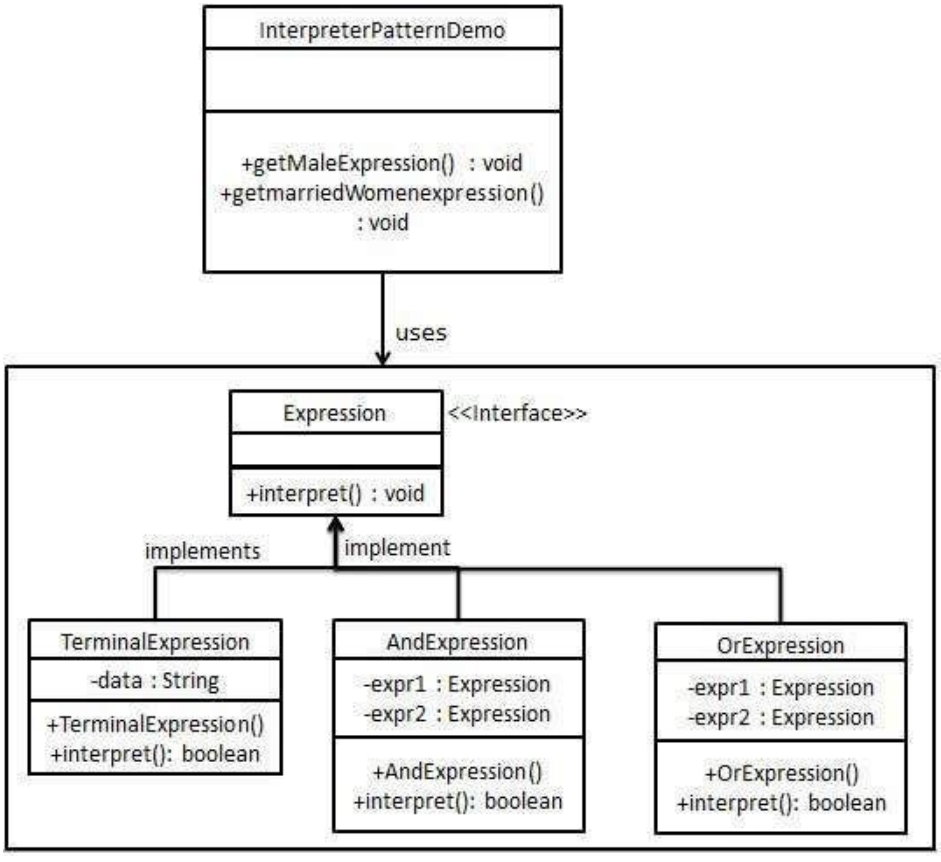
Definir a gramática de uma linguagem e criar um interpretador que leia instruções nesta linguagem e interprete-as para realizar tarefas.

Usar este padrão quando…

- Existe uma linguagem a ser interpretada que pode ser descrita como uma árvore sintática;
- Funciona melhor quando:
  - A linguagem é simples;
  - Desempenho não é uma questão crítica

Vantagens e desvantagens

- É fácil mudar e estender a gramática:
  - Pode alterar expressões existentes, criar novas expressões, etc.;
  - Implementação é simples, pois as estruturas são parecidas.
- Gramáticas complicadas dificultam:
  - Se a gramática tiver muitas regras complica a manutenção.



Passo 1
Crie uma interface de expressão.

Expression.java

```
public interface Expression {
   public boolean interpret(String context);
}
```

Passo 2
Crie classes concretas implementando a interface acima.

TerminalExpression.java

```
public class TerminalExpression implements Expression {

   private String data;

   public TerminalExpression(String data){
      this.data = data;
   }

   @Override
   public boolean interpret(String context) {

      if(context.contains(data)){
         return true;
      }
      return false;
   }
}
```

OrExpression.java

```
public class OrExpression implements Expression {

   private Expression expr1 = null;
   private Expression expr2 = null;

   public OrExpression(Expression expr1, Expression expr2) {
      this.expr1 = expr1;
      this.expr2 = expr2;
   }

   @Override
   public boolean interpret(String context) {
      return expr1.interpret(context) || expr2.interpret(context);
   }
}
```

AndExpression.java

```
public class AndExpression implements Expression {

   private Expression expr1 = null;
   private Expression expr2 = null;

   public AndExpression(Expression expr1, Expression expr2) {
      this.expr1 = expr1;
      this.expr2 = expr2;
   }

   @Override
   public boolean interpret(String context) {
      return expr1.interpret(context) && expr2.interpret(context);
   }
}
```

Passo 3
O InterpreterPatternDemo usa a classe Expression para criar regras e, em seguida, analisá-las.

InterpreterPatternDemo.java

```
public class InterpreterPatternDemo {

   //Rule: Robert and John are male
   public static Expression getMaleExpression(){
      Expression robert = new TerminalExpression("Robert");
      Expression john = new TerminalExpression("John");
      return new OrExpression(robert, john);
   }

   //Rule: Julie is a married women
   public static Expression getMarriedWomanExpression(){
      Expression julie = new TerminalExpression("Julie");
      Expression married = new TerminalExpression("Married");
      return new AndExpression(julie, married);
   }

   public static void main(String[] args) {
      Expression isMale = getMaleExpression();
      Expression isMarriedWoman = getMarriedWomanExpression();

      System.out.println("John is male? " + isMale.interpret("John"));
      System.out.println("Julie is a married women? " + isMarriedWoman.interpret("Married Julie"));
   }
}
```

Passo 4

Teste sua implementação!