

|  |
| --- |
| **Interpreter** |
| **Line** | **Code** |
| **1****2****3****4****5****6****7****8****9****10****11****12****13****14****15****16****17****18****19****20****21****22****23****24****25****26****27****28****29****30****31****32****33****34****35****36****37****38****39****40****41****42****43****44****45****46****47****48****49****50****51****52****53****54****55****56****57****58****59****60****61****62****63****64****65****66****67****68****69****70****71****72****73****74****75****76****77****78****79****80****81****82****83****84****85****86** | /\*-----------------------------------------------------------------Given a language, defines a representation for its grammar along with an interpreter that uses the representation to interpret sentences in the language.This structural code demonstrates the Interpreter patterns, which using a defined grammer, provides the interpreter that processes parsed statements.-------------------------------------------------------------------\*/**using** System;**using** System.Collections;**namespace** Interpreter { **class** Context { } **public interface** Expression { **bool** interpret(**string** context); } **class** TerminalExpression : Expression { **private string** data; **public** TerminalExpression(**string** data) { **this**.data = data; } **public bool** interpret(**string** context) { **return** context.Contains(data); } } **public class** OrExpression : Expression { **private** Expression expr1 = **null**; **private** Expression expr2 = **null**; **public** OrExpression(Expression expr1, Expression expr2) { **this**.expr1 = expr1; **this**.expr2 = expr2; } **public bool** interpret(**string** context) { **return** expr1.interpret(context) || expr2.interpret(context); } } **public class** AndExpression : Expression { **private** Expression expr1 = **null**; **private** Expression expr2 = **null**; **public** AndExpression(Expression expr1, Expression expr2) { **this**.expr1 = expr1; **this**.expr2 = expr2; } **public bool** interpret(**string** context) { **return** expr1.interpret(context) && expr2.interpret(context); } } **public class** InterpreterDemo { //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); } **static void** Main() { Expression isMale = getMaleExpression(); Expression isMarriedWoman = getMarriedWomanExpression(); Console.WriteLine("John is male? " + isMale.interpret("John")); Console.WriteLine("Julie is a married women? " +  isMarriedWoman.interpret("Married Julie")); Console.ReadKey(); } }} |