|  |
| --- |
| **IteratorExComposite.java** |
| **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****87****88****89****90****91****92****93****94****95****96****97****98****99****100****101****102****103****104****105****106****107****108****109****110** | **using** System;**using** System.Collections.Generic;**namespace** IteratorExComposite { **abstract public class** Node { **public readonly string** Name; **public abstract void** addChild(Node child); **public abstract void** removeChild(Node child); **public abstract void** Display(**int** depth); **public** Node(**string** name) { Name = name; } } **internal class** Leaf : Node { **public** Leaf(**string** name):**base**(name){ } **public override void** addChild(Node child) { } **public override void** removeChild(Node child) { } **public override void** Display(**int** depth) { Console.WriteLine(**new** String('-',depth) + Name); } } **internal class** Composite : Node { **internal readonly** List<Node> children = **new** List<Node>(); **public** Composite(**string** name):**base**(name) { } **public** Composite(**string** name, **params** Node[] nodes):**base**(name) { **foreach** (Node node **in** nodes) children.Add(node); } **public override void** addChild(Node child) { children.Add(child); } **public override void** removeChild(Node child) { **if** (children.Contains(child)) children.Remove(child); } **public override void** Display(**int** depth) { Console.WriteLine(**new** String('-',depth) + Name); **foreach** (Node child **in** children) child.Display(depth + 2); } } **abstract public class** Aggregate { **public abstract** Iterator CreateIterator(); } **public class** Tree:Aggregate { **public override** Iterator CreateIterator() { **return new** ConcreteIterator(**this**); } **internal** Node root = **null**; **public** Tree() { root = **new** Composite("Root", **new** Leaf("A"), **new** Composite("B", **new** Composite("E", **new** Leaf("X"), **new** Leaf("Y")), **new** Leaf("F")), **new** Leaf("C"), **new** Composite("D", **new** Leaf("G"), **new** Leaf("H") ) ); } **public void** Show() { root.Display(1); } } **abstract public class** Iterator { **public abstract** Node Next(); } **public class** ConcreteIterator : Iterator { **private readonly** Tree tree; **private int** current = 0; List<Node> nodes = **new** List<Node>(); **public** ConcreteIterator(Aggregate aggregate) { **this**.tree = (Tree)aggregate; FindNode(tree.root, nodes); } **private void** FindNode(Node node, List<Node> nodes) { nodes.Add(node); **if** (node **is** Composite){ **foreach** (Node child **in** ((Composite)node).children)  FindNode(child, nodes); } } **public override** Node Next() { **if** (current < nodes.Count) **return** nodes[current++]; **return null**; } } **class** Program { **static void** Main(**string**[] args) { Tree tree = **new** Tree(); tree.Show(); Console.WriteLine("------ The Nodes in the Tree -------"); Iterator i = tree.CreateIterator(); Node n = i.Next(); **while** (n != **null**) { Console.WriteLine(n.Name); n = i.Next(); } Console.ReadKey(); } }} |