|  |  |
| --- | --- |
| **CompositeDemo** | |
| **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** | /\*---------------------------------------------------------------  Composes objects into tree structures to represent part-whole  hierarchies. Composite lets clients treat individual objects and  compositions of objects uniformly.  This structural code demonstrates the Composite pattern which allows  the creation of a tree structure in which individual nodes are accessed  uniformly whether they are leaf nodes or branch (composite) nodes  -----------------------------------------------------------------\*/  **using** System;  **using** System.Collections.Generic;  **namespace** Composite{  **abstract class** Component{  **protected string** name;  **public** Component(**string** name){  **this**.name = name;  }  **public abstract void** Add(Component child);  **public abstract void** Remove(Component child);  **public abstract void** Display(**int** depth);  }  **class** Composite:Component{  **private** List<Component> children = **new** List<Component>();  **public** Composite(**string** name):**base**(name){}  **public override void** Add(Component child){  children.Add(child);  }  **public override void** Remove(Component child) {  children.Remove(child);  }  **public override void** Display(**int** depth){  Console.WriteLine(**new** String('-', depth) + name);  //Recursively display child nodes  **foreach** (Component child **in** children){  child.Display(depth + 2);  }  }  }  **class** Leaf:Component{  **public** Leaf(**string** name):**base**(name){}  **public override void** Add(Component child) {  //Console.WriteLine("Cannot add to a leaf");  }  **public override void** Remove(Component child) {  //Console.WriteLine("Cannot remove from a leaf");  }  **public override void** Display(**int** depth){  Console.WriteLine(**new** String('-', depth) + name);  }  }    **class** Program{  **static void** Main(){  // Create a tree structure  Component root = **new** Composite("root");  root.Add(**new** Leaf("Leaf A"));  root.Add(**new** Leaf("Leaf B"));  Composite comp = **new** Composite("Composite X");  comp.Add(**new** Leaf("Leaf XA"));  comp.Add(**new** Leaf("Leaf XB"));  root.Add(comp);  root.Add(**new** Leaf("Leaf C"));  //Add and remove a leaf  Leaf leaf = **new** Leaf("Leaf D");  root.Add(leaf);  root.Remove(leaf);  //Recursively display tree  root.Display(1);  Console.ReadKey();  }  }  }  /\*=================== OUTPUT ===================  -root  ---Leaf A  ---Leaf B  ---Composite X  -----Leaf XA  -----Leaf XB  ---Leaf C  =================================================\*/ |