|  |  |
| --- | --- |
| **Decorator** | |
| **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** | // Shows two decorators and the output of various  // combinations of the decorators on the basic component  **using** System;  **namespace** Decorator{  **interface** IComponent{  **string** Operation();  }  **class** Component:IComponent{  **public string** Operation(){  **return** "I am walking ";  }  }  **abstract class** Decorator : IComponent {  **protected** IComponent component;  **abstract public string** Operation();  **public** Decorator(IComponent comp) {  component = comp;  }  }  **class** DecoratorA : Decorator {  **public** DecoratorA(IComponent c):**base**(c){}  **override public string** Operation(){  **string** s = component.Operation();  s += "and listening to Classic FM ";  **return** s;  }  }  **class** DecoratorB : Decorator {  **public string** addedState = "past the Coffee Shop ";  **public** DecoratorB(IComponent c):**base**(c){}  **override public string** Operation() {  **string** s = component.Operation();  s += "to school ";  **return** s;  }  **public string** AddedBehavior(){  **return** "and I bought a cappucino ";  }  }  **class** Program{  **static void** Display(**string** s, IComponent c){  Console.WriteLine(s + c.Operation());  }  **static void** Main(**string**[] args){  Console.WriteLine("Decorator Pattern\n");  IComponent component = **new** Component();  Display("1. Basic component: ", component);  Display("2. A-decorated : ", **new** DecoratorA(component));  Display("3. B-decorated : ", **new** DecoratorB(component));  Display("4. B-A-decorated : ", **new** DecoratorB(  **new** DecoratorA(component)));  // Explicit DecoratorB  DecoratorB b = **new** DecoratorB(**new** Component());  Display("5. A-B-decorated : ", **new** DecoratorA(b));  //Invoking its added state and added behaviour  Console.WriteLine("\t\t\t" + b.addedState + b.AddedBehavior());  Console.ReadKey();  }  }  } |