







|  |  |
| --- | --- |
| **CompositeEx.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** | **package** Structural;  **import** java.util.ArrayList;  **abstract class** Valuable {  **public final double** weight;  **public** Valuable(**double** weight) {  **this**.weight = weight;  }  **public abstract void** Add(Valuable child);  **public abstract void** Remove(Valuable child);  **public abstract double** getValue();  }  **class** VN **extends** Valuable {  **double** value;  **public** VN(**double** weight, **double** value){  **super**(weight);  **this**.value = value;  }  **@Override**  **public void** Add(Valuable child) {  //Console.WriteLine("Cannot add to a leaf");  }  **@Override**  **public void** Remove(Valuable child) {  //Console.WriteLine("Cannot remove from a leaf");  }  **public double** getValue() {  **return** value;  }  }  **class** MN **extends** Valuable {  **private final** ArrayList<Valuable> children = **new** ArrayList<>();  **public** MN(**double** weight){  **super**(weight);  }  **public** MN(**double** weight, Valuable... inputs){  **super**(weight);  **for**(Valuable input:inputs) Add(input);  }  **@Override**  **public void** Add(Valuable child) {  children.add(child);  }  **@Override**  **public void** Remove(Valuable child) {  children.remove(child);  }  **final double** getSum() {  **double** totalSumProduct = 0;  **double** totalWeight = 0;  **for**(Valuable child:children) {  totalSumProduct += child.getValue() \* child.weight;  totalWeight += child.weight;  }  **return** totalSumProduct / totalWeight;  }  **protected double** getExNormalize() {  **return** getSum();  }  **@Override**  **public double** getValue() {  **return** getExNormalize();  }  }  **public class** CompositeEx {  **static public void** main(String[] args){  MN TMX2014 = **new** MN( 0.4,  **new** VN(0.1, 60),  **new** VN(0.3, 70),  **new** VN(0.2, 80),  **new** VN(0.3, 90)  );  MN TMT2034 = **new** MN(0.1,  **new** VN(0.2, 80),  **new** VN(0.3, 40),  **new** VN(0.3, 90),  **new** VN(0.4, 90)  );  MN Year2Result = **new** MN(0.3,  **new** VN(0.3, 80),  TMX2014,  **new** VN(0.5, 80),  TMT2034  );  VN CoCurriculum = **new** VN(0.2, 80);  VN Internship = **new** VN(0.1, 90);  MN CGPA = **new** MN(1,  CoCurriculum,  Internship,  Year2Result,  **new** VN(0.4, 75)  );  System.out.printf("CGPA is %.2f%n",4\*CGPA.getValue()/100);  }  }  /\*  Output: CGPA is 3.15  \*/ |

|  |  |
| --- | --- |
| **CompositeAndPrototypeEx.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**  **111**  **112**  **113**  **114**  **115**  **116**  **117**  **118**  **119**  **120**  **121**  **122**  **123**  **124**  **125**  **126**  **127**  **128**  **129**  **130**  **131**  **132**  **133**  **134**  **135**  **136**  **137**  **138**  **139**  **140**  **141**  **142**  **143**  **144**  **145**  **146**  **147**  **148**  **149**  **150**  **151**  **152**  **153**  **154**  **155**  **156**  **157**  **158**  **159**  **160**  **161** | **package** Structural;  **import** java.util.ArrayList;  **public class** CompositeAndPrototypeEx {  **static abstract class** Valuable{  **public final double** weight;  **public** Valuable(**double** weight) {  **this**.weight = weight;  }  **public abstract void** Add(Valuable child);  **public abstract void** Remove(Valuable child);  **public abstract double** getValue();  **public abstract** Valuable ShallowCopy();  **public abstract** Valuable DeepCopy();  }  **static class** VN **extends** Valuable **implements** Cloneable{  **double** value;  **public** VN(**double** weight, **double** value){  **super**(weight);  **this**.value = value;  }  **@Override**  **public void** Add(Valuable child) {  //Console.WriteLine("Cannot add to a leaf");  }  **@Override**  **public void** Remove(Valuable child) {  //Console.WriteLine("Cannot remove from a leaf");  }  **@Override**  **public double** getValue() {  **return** value;  }    **@Override**  **public** Valuable ShallowCopy(){  **try**{  **return** (Valuable)**this**.clone();  }  **catch**(Exception e){  System.out.println(e.getMessage());  }  **return** **null**;  }    **@Override**  **public** Valuable DeepCopy(){  **return** ShallowCopy();  }  }  **static class** MN **extends** Valuable **implements** Cloneable{  **private** ArrayList<Valuable> children = **new** ArrayList<>();//Non final  No more final  **public** MN(**double** weight){  **super**(weight);  }  **public** MN(**double** weight, Valuable... inputs){  **super**(weight);  **for**(Valuable input:inputs) Add(input);  }  **@Override**  **public void** Add(Valuable child) {  children.add(child);  }  **@Override**  **public void** Remove(Valuable child) {  children.remove(child);  }  **final double** getSum() {  **double** totalSumProduct = 0;  **double** totalWeight = 0;  **for**(Valuable child:children) {  totalSumProduct += child.getValue() \* child.weight;  totalWeight += child.weight;  }  **return** totalSumProduct / totalWeight;  }  **protected double** getExNormalize() {  **return** getSum();  }  **@Override**  **public double** getValue() {  **return** getExNormalize();  }    **@Override**  **public** Valuable ShallowCopy(){  **try**{  **return**(MN)**this**.clone();  }  **catch**(Exception e){  System.out.println(e.getMessage());  }  **return** **null**;  }  **@Override**  **public** Valuable DeepCopy(){  MN clone = (MN)ShallowCopy();  **if**(clone!=**null**){  clone.children = **new** ArrayList<Valuable>();  **for**(Valuable v:children){  clone.children.add(v.DeepCopy());  }  }  **return** clone;  }  }    **static public void** main(String[] args){  MN TMX2014 = **new** MN( 0.4,  **new** VN(0.1, 60),  **new** VN(0.3, 70),  **new** VN(0.2, 80),  **new** VN(0.3, 90)  );  MN TMT2034 = **new** MN(0.1,  **new** VN(0.2, 80),  **new** VN(0.3, 40),  **new** VN(0.3, 90),  **new** VN(0.4, 90)  );  MN Year2Result = **new** MN(0.3,  **new** VN(0.3, 80),  TMX2014,  **new** VN(0.5, 80),  TMT2034  );  VN CoCurriculum = **new** VN(0.2, 80);  VN Internship = **new** VN(0.1, 90);  MN CGPA = **new** MN(1,  CoCurriculum,  Internship,  Year2Result,  **new** VN(0.4, 75)  );  System.out.printf("CGPA is %.2f%n",4\*CGPA.getValue()/100);  /\* Output: CGPA is 3.15 \*/    System.out.println("---------Deep Copy-----------");  Valuable Clone = CGPA.DeepCopy();  Clone.Add(**new** VN(0.2, 60));  System.out.printf("Clone's CGPA is %.2f%n",4\*Clone.getValue()/100);  System.out.printf("CGPA is %.2f%n",4\*CGPA.getValue()/100);    System.out.println("---------Shallow Copy-----------");  Clone = CGPA.ShallowCopy();  Clone.Add(**new** VN(0.2, 90));  System.out.printf("Clone's CGPA is %.2f%n",4\*Clone.getValue()/100);  System.out.printf("CGPA is %.2f%n",4\*CGPA.getValue()/100);  }  } |