COMP 1210 (COMP-1210-001-Fall-2022): M08 Project 08 Skeleton Code (ungraded), by Ssai Sanjanna Ganji

< Back to Summary

Assignment	COMP 1210 (COMP-1210-001-Fall-2022): M08 Project 08 Skeleton Code (ungraded) try #5
Name	Ssai Sanjanna Ganji (szg0161)
Partners	Ssai Sanjanna Ganji (szg0161)
Submitted	10/28/22 10:37PM, 1 hr, 21 mins early
Total Score	**2.0**/2.0

Score Summary	
Style/Coding:	1.0/1.0
Correctness/Testing:	1.0/1.0
Final score:	**2.0/2.0**
Position in class:	

File	Remarks	Deductions	Methods and Conditions Executed
RingTorus.java	0	0.0	100.0%
RingTorusList.java	0	0.0	94.2%
RingTorusListTest.java	0	0.0	
RingTorusTest.java	0	0.0	

RingTorus.java

```
1  import java.text.DecimalFormat;
2   /**
3  * Program that contains two classes: First class represents a RingTorus object,
4  * implements comparable interface for object of type RingTorus and the
5  * second class is build with set of test methods which includes method,
6  * statement and conditional coverage for each emthod in RingTorus class.
7  *
8  * Project_7B.
9  * @author Ssai Sanjanna Ganji - COMP 1210-006
10 * @version 10/21/22
11 */
12 public class RingTorus implements Comparable<RingTorus> {
13 /**
14      * Initialising instance variables and class variables.
15      * @param args Command line arguments (not used).
16      */
17
18    // Instance variables
19    private String label = "";
```

```
20      private double largeRadius = 0;
21      private double smallRadius = 0;
22
23      // Class variable
24      // number of RingTorus objects
25      private static int count = 0;
26
27      /**
28         * constructor takes a string and two double values as parameters.
29         * @param labelIn for label
30         * @param largeRadiusIn for largeRadius
31         * @param smallRadiusIn for smallRadius
32         */
33      // constructor
34      public RingTorus(String labelIn, double largeRadiusIn, double smallRadiusIn)
35      {
36         setLabel(labelIn);
37         setLargeRadius(largeRadiusIn);
38         setSmallRadius(smallRadiusIn);
39         // count is increased by 1, each time a RingTorus object is constructed
40         count++;
41      }
42
43      // methods
44     /**
45         * creating a method to get label of type string.
46         * @return label
47         */
48      public String getLabel() {
49         return label;
50      }
51
52      /**
53         * creating a method to set the string label.
54         * @param labelIn for label
55         * @return true if labelIn is not null, otherwise false
56         */
57      public boolean setLabel(String labelIn) {
58         if (labelIn != null) {
59            // sets trimmed string labelIn to label
60            label = labelIn.trim();
61            return true;
62         }
63         // if string labelIn is null then label is not set
64         return false;
65      }
66
67      /**
68         * creating a method to get large radius.
69         * @return large radius
70         */
71      public double getLargeRadius() {
72         return largeRadius;
73      }
74
75      /**
76         * creating a method to set largeRadius of type double.
```

```
77          * @param largeRadiusIn for largeRadius
78          * @return true if largeRadiusIn is positive and greater than
79          * current small radius, otherwise false
80          */
81       public boolean setLargeRadius(double largeRadiusIn) {
82          if (largeRadiusIn > 0 && largeRadiusIn > smallRadius) {
83             // sets largeRadiusIn to largeRadius
84             largeRadius = largeRadiusIn;
85             return true;
86          }
87          // if largeRadiusIn is negative or is less than small radius
88          // largeRadius is not set
89          return false;
90       }
91
92       /**
93          * creating a method to get small radius.
94          * @return small radius
95          */
96       public double getSmallRadius() {
97          return smallRadius;
98       }
99
100      /**
101         * creating a method to set the smallradius of type double.
102         * @param smallRadiusIn for smallRadius
103         * @return true if smallRadiusIn is positive and less than
104         * current large radius, otherwise false
105         */
106      public boolean setSmallRadius(double smallRadiusIn) {
107         if (smallRadiusIn > 0 && smallRadiusIn < largeRadius) {
108            // sets smallRadiusIn to smallRadius
109            smallRadius = smallRadiusIn;
110            return true;
111         }
112         // if smallRadiusIn is negative or is less than large radius
113         // smallRadius is not set
114         return false;
115      }
116
117      /**
118         * creating a method to calculate diamter of the object.
119         * @return diameter
120         */
121      public double diameter() {
122         // diameter = 2(R + r)
123         double diameter = 2 * (largeRadius + smallRadius);
124         return diameter;
125      }
126
127      /**
128         * creating a method to calculate surface area of the object.
129         * @return surface area
130         */
131      public double surfaceArea() {
132         // surface area = (2.Pi.R)(2.Pi.r)
```

```
133         double surfaceArea = (2 * Math.PI * largeRadius)
134            * (2 * Math.PI * smallRadius);
135         return surfaceArea;
136      }
137
138      /**
139         * creating a method to calculate volume of the object.
140         * @return volume
141         */
142      public double volume() {
143         // volume = (2.Pi.R)(Pi.r^2)
144         double volume = (2 * Math.PI * largeRadius)
145            * (Math.PI * Math.pow(smallRadius, 2));
146         return volume;
147      }
148
149     /**
150         * creating a method to represent information about the RingTorus object
151         * such as label, large radius, small radius, diameter, surface area
152         * and volume.
153         * @return output string
154         */
155      public String toString() {
156         DecimalFormat df = new DecimalFormat("#,##0.0##");
157         String output = "RingTorus \"" + label + "\"\n\t";
158         output += "large radius = " + df.format(largeRadius) + " units\n\t";
159         output += "small radius = " + df.format(smallRadius) + " units\n\t";
160         output += "diameter = " + df.format(diameter()) + " units\n\t";
161         output += "surface area = " + df.format(surfaceArea()) + " square units";
162         output += "\n\tvolume = " + df.format(volume()) + " cubic units\n";
163         return output;
164      }
165
166      /**
167         * creating a method to get number of RingTorus objects.
168         * @return count
169         */
170      public static int getCount() {
171         return count;
172      }
173
174     /**
175         * creating a method to reset number of RingTorus objects.
176         */
177      public static void resetCount() {
178         // number of objects sets to zero
179         count = 0;
180      }
181
182      /**
183        * creating an instance method that takes an Object.
184        * @param obj for Object
185        * @return false if parameter of object is not a RingTorus object
186        * otherwise true
187       */
188      public boolean equals(Object obj) {
```

```
189         // if object taken at instance is not a RingTorus object
190         if (!(obj instanceof RingTorus)) {
191            return false;
192         }
193         else {
194            // creating an instance object equal to a RingTorus object
195            RingTorus rt = (RingTorus) obj;
196            return (label.equalsIgnoreCase(rt.getLabel())
197                   && (Math.abs(largeRadius - rt.getLargeRadius()) < .000001)
198                   && (Math.abs(smallRadius - rt.getSmallRadius()) < .000001));
199         }
200      }
201
202      /**
203       * creating a method which is used only when equals method is implemented.
204       * @return zero
205      */
206      public int hashCode() {
207         return 0;
208      }
209
210      /**
211         * creating a compareTo method to compare objects between
212         * a RingTorus object and any other object which is of type RingTorus.
213         * @param obj for any other object
214         * @return -1 or 1 or 0 based on conditions mentioned
215         */
216      public int compareTo(RingTorus obj) {
217
218      // if RingTorus object's volume is less than
219      // volume of the object taken as paramter
220         if (this.volume() < obj.volume()) {
221            // returns the negative integer '-1'
222            return -1;
223         }
224
225         // if RingTorus object's volume is greater than
226         // volume of the object taken as paramter
227         else if (this.volume() > obj.volume()) {
228            // returns the positive integer '1'
229            return 1;
230         }
231
232         // if RingTorus object's volume is equal to
233         // volume of object taken as paramter
234         else {
235            // returns the integer '0'
236            return 0;
237         }
238      }
239   }
```

RingTorusList.java

```
1   import java.text.DecimalFormat;
```

```java
 /**
* Program that contains four classes: The first represents a RingTorus object,
* the second one is a JUnit test class for RingTorus class,
* the third one represents a list of RingTorus objects,
* and the fourth is a JUnit test class for RingTorusList class.
*
* Project_8.
* @author Ssai Sanjanna Ganji - COMP 1210-006
* @version 10/28/22
*/
public class RingTorusList {
/**
     * Initialising instance variables and class variables.
     * @param args Command line arguments (not used).
     */

   // Instance variables
   private String listName;
   private RingTorus[] rtList;
   private int numObjects;

   /**
      * constructor takes a string, an array and a int.
      * @param listNameIn for listNsme
      * @param rtListIn for rtList
      * @param numObjectsIn for numObjects
      */
   // constructor
   public RingTorusList(String listNameIn, RingTorus[] rtListIn,
      int numObjectsIn) {
      listName = listNameIn;
      rtList = rtListIn;
      numObjects = numObjectsIn;
   }

   // methods
  /**
      * creating a method to get name of list of type String.
      * @return listName
      */
   public String getName() {
      return listName;
   }

    /**
      * creating a method to represent number of RingTorus objects in list.
      * @return number of RingTorus objects
      */
   public int numberOfRingToruses() {
      return numObjects;
   }

   /**
      * creating a method to represent the total diameters for all objects
      * in list.
      * @return sum of diameters
```

```
58          */
59       public double totalDiameter() {
60          double objDiameter = 0;
61          double sumDiameters = 0;
62          int index = 0;
63          if (numObjects > 0) {
64             while (index < numObjects) {
65                RingTorus torusObj = rtList[index];
66                objDiameter = torusObj.diameter();
67                sumDiameters += objDiameter;
68                index++;
69             }
70          }
71          return sumDiameters;
72
73       }
74
75        /**
76          * creating a method to represent the total surface areas for all objects
77          * in list.
78          * @return sum of surface areas
79          */
80       public double totalSurfaceArea() {
81          double objArea = 0;
82          double sumAreas = 0;
83          int index = 0;
84          if (numObjects > 0) {
85             while (index < numObjects) {
86                RingTorus torusObj = rtList[index];
87                objArea = torusObj.surfaceArea();
88                sumAreas += objArea;
89                index++;
90             }
91          }
92          return sumAreas;
93       }
94
95        /**
96          * creating a method to represent the total volume for all objects
97          * in list.
98          * @return sum of volumes
99          */
100      public double totalVolume() {
101         double objVolume = 0;
102         double sumVolumes = 0;
103         int index = 0;
104         if (numObjects > 0) {
105            while (index < numObjects) {
106               RingTorus torusObj = rtList[index];
107               objVolume = torusObj.volume();
108               sumVolumes += objVolume;
109               index++;
110            }
111         }
112         return sumVolumes;
113      }
```

```
114
115      /**
116         * creating a method to represent the average diameter for all objects
117         * in list.
118         * @return average of diameters
119         */
120      public double averageDiameter() {
121         double avgDiameter = 0;
122         if (numObjects != 0) {
123            avgDiameter = totalDiameter() / numberOfRingToruses();
124            return avgDiameter;
125         }
126         return 0;
127      }
128
129      /**
130         * creating a method to represent the average surface area for all objects
131         * in list.
132         * @return average of surface areas
133         */
134      public double averageSurfaceArea() {
135         double avgArea = 0;
136         if (numObjects != 0) {
137            avgArea = totalSurfaceArea() / numberOfRingToruses();
138            return avgArea;
139         }
140         return 0;
141      }
142
143      /**
144         * creating a method to represent the average volume for all objects
145         * in list.
146         * @return average of volumes
147         */
148      public double averageVolume() {
149         double avgVolume = 0;
150         if (numObjects != 0) {
151            avgVolume = totalVolume() / numberOfRingToruses();
152            return avgVolume;
153         }
154         return 0;
155      }
156
157      /**
158         * creating a method to represent information about the RingTorus list
159         * such as name of list, number of objects in it, total and average
160         * diameter, total and average surface area, total and average of volume.
161         * @return output string
162         */
163      public String toString() {
164         DecimalFormat df = new DecimalFormat("#,##0.0##");
165         String output = "";
166         output = "----- Summary for " + getName() + " -----";
167         output += "\nNumber of RingToruses: " + numberOfRingToruses();
168         output += "\nTotal Diameter: " + df.format(totalDiameter()) + " units";
169         output += "\nTotal Surface Area: " + df.format(totalSurfaceArea())
```

```
170            + " square units";
171         output += "\nTotal Volume: " + df.format(totalVolume())
172            + " cubic units";
173         output += "\nAverage Diameter: " + df.format(averageDiameter())
174            + " units";
175         output += "\nAverage Surface Area: " + df.format(averageSurfaceArea())
176            + " square units";
177         output += "\nAverage Volume: " + df.format(averageVolume())
178            + " cubic units";
179         return output;
180      }
181
182       /**
183         * creating a method to get the array list of RingTorus objects.
184         * @return array list
185         */
186      public RingTorus[] getList() {
187         return rtList;
188      }
189
190       /**
191         * creating a method to add a new RingTorus object to the array list.
192         * @param label1 for label
193         * @param largeRadius1 for largeRadius
194         * @param smallRadius1 for smallRadius
195         */
196      public void addRingTorus(String label1, double largeRadius1,
197            double smallRadius1) {
198         RingTorus torusObj = new RingTorus(label1, largeRadius1, smallRadius1);
199         rtList[numObjects] = torusObj;
200         numObjects++;
201      }
202
203       /**
204         * creating a method to find the corresponding RingTorus object from list.
205         * @param label1 for label
206         * @return RingTorus object
207         */
208      public RingTorus findRingTorus(String label1)
209      {
210         int index = 0;
211         while (index < numObjects)
212         {
213            RingTorus torus = rtList[index];
214            String label2 = torus.getLabel();
215            if (label1.equalsIgnoreCase(label2))
216            {
217               return rtList[index];
218            }
219            index++;
220         }
221         return null;
222      }
223
224       /**
225         * creating a method to delete the corresponding RingTorus object
```

```
226         * from list.
227         * @param label1 for label
228         * @return RingTorus object
229         */
230      public RingTorus deleteRingTorus(String label1) {
231         RingTorus torusDelete = null;
232         for (int i = 0; i < numObjects; i++) {
233            if (rtList[i].getLabel().equalsIgnoreCase(label1)) {
234               torusDelete = rtList[i];
235               for (int j = i; j < numObjects - 1; j++) {
236                  rtList[j] = rtList[j + 1];
237               }
238               rtList[numObjects - 1] = null;
239               numObjects--;
240            }
241         }
242         return torusDelete;
243      }
244
245       /**
246         * creates a method to edit the object in the list.
247         * @param label1 for label
248         * @param largeRadius1 for largeRadius
249         * @param smallRadius1 for smallRadius
250         * @return true if input label is matched, otherwise return false
251         */
252      public boolean editRingTorus(String label1, double largeRadius1,
253         double smallRadius1) {
254         boolean result = false;
255         int index = 0;
256         while (index < numObjects)
257         {
258            RingTorus torusEdit = rtList[index];
259            String label2 = torusEdit.getLabel();
260            if (label1.equalsIgnoreCase(label2))
261            {
262               torusEdit.setLargeRadius(largeRadius1);
263               torusEdit.setSmallRadius(smallRadius1);
264               result = true;
265            }
266            index++;
267         }
268         return result;
269      }
270
271       /**
272         * creating a method to find the object with largest volume.
273         * @return RingTorus object
274         */
275      public RingTorus findRingTorusWithLargestVolume() {
276
277         if (numObjects > 0) {
278            RingTorus largeVolumeObj = rtList[0];
279            for (int i = 0; i < numObjects; i++) {
280               if (largeVolumeObj.volume() <= rtList[i].volume()) {
281                  largeVolumeObj = rtList[i];
```

```
282             }
283          }
284          return largeVolumeObj;

285       }
286       return null;
287    }
288 }
```

RingTorusListTest.java

```
 1  import org.junit.Assert;
 2  // import static org.junit.Assert.*;
 3  import org.junit.Before;
 4  import org.junit.Test;
 5  /**
 6  * Test file that tests whether the methods from
 7  * RingTorusList class are correct or not.
 8  *
 9  * Project_8.
10  * @author Ssai Sanjanna Ganji - COMP 1210-006
11  * @version 10/28/22
12  */
13  public class RingTorusListTest {
14    /** Fixture initialization (common initialization
15       *  for all tests).
16       **/
17     @Before public void setUp() {
18     }
19
20  /**
21     * A test to check getName method.
22     */
23     @Test public void getNameTest() {
24        RingTorus ex1 = new RingTorus("Small Example", 9.5, 1.25);
25        RingTorus ex2 = new RingTorus("Medium Example", 35.1, 10.4);
26        RingTorus ex3 = new RingTorus("Large Example", 134.28, 32.46);
27        RingTorus[] rtList = {ex1, ex2, ex3};
28        RingTorusList torusList = new RingTorusList("Array List", rtList, 3);
29
30        Assert.assertEquals("Get Name test: ", "Array List", torusList.getName());
31     }
32  /**
33     * A test to check numberOfRingToruses method.
34     */
35     @Test public void numberOfRingTorusesTest() {
36        RingTorus ex1 = new RingTorus("Small Example", 9.5, 1.25);
37        RingTorus ex2 = new RingTorus("Medium Example", 35.1, 10.4);
38        RingTorus ex3 = new RingTorus("Large Example", 134.28, 32.46);
39        RingTorus[] rtList = {ex1, ex2, ex3};
40        RingTorusList torusList = new RingTorusList("Array List", rtList, 3);
41
42        Assert.assertEquals("Number of objects: ", 3,
43           torusList.numberOfRingToruses());
44     }
45  /**
```

```
46      * A test to check totalDiameter method.
47      */
48      @Test public void totalDiameter() {
49         RingTorus ex1 = new RingTorus("Small Example", 9.5, 1.25);
50         RingTorus ex2 = new RingTorus("Medium Example", 35.1, 10.4);
51         RingTorus ex3 = new RingTorus("Large Example", 134.28, 32.46);
52         RingTorus[] rtList1 = {ex1, ex2, ex3};
53         RingTorusList torusList1 = new RingTorusList("Array List", rtList1, 3);
54
55         Assert.assertEquals("Total diameter test: ", 445.98,
56            torusList1.totalDiameter(), 0.001);
57         RingTorus[] rtList2 = {};
58         RingTorusList torusList2 = new RingTorusList("Array List 2", rtList2, 0);
59         Assert.assertEquals("Total diameter test: ", 0,
60            torusList2.totalDiameter(), 0.001);
61      }
62      /**
63      * A test to check totalSurfaceArea method.
64      */
65      @Test public void totalSurfaceAreaTest() {
66         RingTorus ex1 = new RingTorus("Small Example", 9.5, 1.25);
67         RingTorus ex2 = new RingTorus("Medium Example", 35.1, 10.4);
68         RingTorus ex3 = new RingTorus("Large Example", 134.28, 32.46);
69         RingTorus[] rtList1 = {ex1, ex2, ex3};
70         RingTorusList torusList1 = new RingTorusList("Array List 1", rtList1, 3);
71
72         Assert.assertEquals("Total surface area test: ", 186955.724,
73            torusList1.totalSurfaceArea(), 0.001);
74         RingTorus[] rtList2 = {};
75         RingTorusList torusList2 = new RingTorusList("Array List 2", rtList2, 0);
76         Assert.assertEquals("Total volume test: ", 0,
77            torusList2.totalVolume(), 0.001);
78      }
79   /**
80      * A test to check totalVolume method.
81      */
82      @Test public void totalVolumeTest() {
83         RingTorus ex1 = new RingTorus("Small Example", 9.5, 1.25);
84         RingTorus ex2 = new RingTorus("Medium Example", 35.1, 10.4);
85         RingTorus ex3 = new RingTorus("Large Example", 134.28, 32.46);
86         RingTorus[] rtList1 = {ex1, ex2, ex3};
87         RingTorusList torusList1 = new RingTorusList("Array List 1", rtList1, 3);
88
89         Assert.assertEquals("Total volume test: ", 2868020.119,
90            torusList1.totalVolume(), 0.001);
91         RingTorus[] rtList2 = {};
92         RingTorusList torusList2 = new RingTorusList("Array List 2", rtList2, 0);
93         Assert.assertEquals("Total volume test: ", 0,
94            torusList2.totalVolume(), 0.001);
95      }
96      /**
97      * A test to check averageDiameter method.
98      */
99      @Test public void averageDiameterTest() {
100        RingTorus ex1 = new RingTorus("Small Example", 9.5, 1.25);
101        RingTorus ex2 = new RingTorus("Medium Example", 35.1, 10.4);
```

```
102         RingTorus ex3 = new RingTorus("Large Example", 134.28, 32.46);
103         RingTorus[] rtList1 = {ex1, ex2, ex3};
104         RingTorusList torusList1 = new RingTorusList("Array List 1", rtList1, 3);
105
106         Assert.assertEquals("Avg diameter test: ", 148.66,
107            torusList1.averageDiameter(), 0.001);
108         RingTorus[] rtList2 = {};
109         RingTorusList torusList2 = new RingTorusList("Array List 2", rtList2, 0);
110         Assert.assertEquals("Avg diameter test: ", 0,
111            torusList2.averageDiameter(), 0.001);
112      }
113   /**
114      * A test to check averageSurfaceArea method.
115      */
116      @Test public void averageSurfaceAreaTest() {
117         RingTorus ex1 = new RingTorus("Small Example", 9.5, 1.25);
118         RingTorus ex2 = new RingTorus("Medium Example", 35.1, 10.4);
119         RingTorus ex3 = new RingTorus("Large Example", 134.28, 32.46);
120         RingTorus[] rtList1 = {ex1, ex2, ex3};
121         RingTorusList torusList1 = new RingTorusList("Array List 1", rtList1, 3);
122
123         Assert.assertEquals("Avg surface area test: ", 62318.575,
124            torusList1.averageSurfaceArea(), 0.001);
125         RingTorus[] rtList2 = {};
126         RingTorusList torusList2 = new RingTorusList("Array List 2", rtList2, 0);
127         Assert.assertEquals("Avg surface area test: ", 0,
128            torusList2.averageSurfaceArea(), 0.001);
129      }
130   /**
131      * A test to check averageVolume method.
132      */
133      @Test public void averageVolumeTest() {
134         RingTorus ex1 = new RingTorus("Small Example", 9.5, 1.25);
135         RingTorus ex2 = new RingTorus("Medium Example", 35.1, 10.4);
136         RingTorus ex3 = new RingTorus("Large Example", 134.28, 32.46);
137         RingTorus[] rtList1 = {ex1, ex2, ex3};
138         RingTorusList torusList1 = new RingTorusList("Array List 1", rtList1, 3);
139
140         Assert.assertEquals("Avg volume test: ", 956006.706,
141            torusList1.averageVolume(), 0.001);
142         RingTorus[] rtList2 = {};
143         RingTorusList torusList2 = new RingTorusList("Array List 2", rtList2, 0);
144         Assert.assertEquals("Avg volume test: ", 0,
145            torusList2.averageVolume(), 0.001);
146      }
147   /**
148      * A test to check toString method.
149      */
150      @Test public void toStringTest() {
151         RingTorus ex1 = new RingTorus("Small Example", 9.5, 1.25);
152         RingTorus ex2 = new RingTorus("Medium Example", 35.1, 10.4);
153         RingTorus ex3 = new RingTorus("Large Example", 134.28, 32.46);
154         RingTorus[] rtList = {ex1, ex2, ex3};
155         RingTorusList torusList = new RingTorusList("Array List", rtList, 3);
156
157         Assert.assertTrue("String test: ",
```

```
158             torusList.toString().contains("Summary"));
159
160       }
161    /**
162       * A test to check getList method.
163       */
164       @Test public void getListTest() {
165          RingTorus ex1 = new RingTorus("Small Example", 9.5, 1.25);
166          RingTorus ex2 = new RingTorus("Medium Example", 35.1, 10.4);
167          RingTorus ex3 = new RingTorus("Large Example", 134.28, 32.46);
168          RingTorus[] rtList = {ex1, ex2, ex3};
169          RingTorusList torusList = new RingTorusList("Array List", rtList, 3);
170
171          Assert.assertArrayEquals("Get Name test: ", rtList, torusList.getList());
172       }
173       /**
174       * A test to check addRingTorus method.
175       */
176       @Test public void addRingTorusTest()
177       {
178          RingTorus[] rtList1 = new RingTorus[4];
179          RingTorus ex1 = new RingTorus("Small Example", 9.5, 1.25);
180          RingTorus ex2 = new RingTorus("Medium Example", 35.1, 10.4);
181          RingTorus ex3 = new RingTorus("Large Example", 134.28, 32.46);
182          RingTorusList torusList = new RingTorusList("Array List", rtList1, 3);
183
184          //Creating new list
185          RingTorus ex4 = new RingTorus("Fourth Example", 12.8, 8.9);
186          torusList.addRingTorus("Fourth Example", 12.8, 8.9);
187          RingTorus[] rtList2 = torusList.getList();
188          Assert.assertEquals(ex4, rtList1[3]);
189       }
190
191       /**
192       * A test to check findRingTorus method.
193       */
194       @Test public void findRingTorusTest() {
195          RingTorus ex1 = new RingTorus("Small Example", 9.5, 1.25);
196          RingTorus ex2 = new RingTorus("Medium Example", 35.1, 10.4);
197          RingTorus ex3 = new RingTorus("Large Example", 134.28, 32.46);
198          RingTorus[] rtList = {ex1, ex2, ex3};
199          RingTorusList torusList = new RingTorusList("Array List", rtList, 3);
200
201          Assert.assertEquals("Find object test: ", ex1,
202             torusList.findRingTorus("Small Example"));
203          Assert.assertEquals("Find object test: ", null,
204             null);
205          RingTorus[] rtList2 = {};
206          RingTorusList torusList2 = new RingTorusList("Array List 2", rtList2, 0);
207          Assert.assertEquals("Find object test ", null,
208             torusList2.findRingTorus("Small Example"));
209       }
210       /**
211       * A test to check deleteRingTorus method.
212       */
213       @Test public void deleteRingTorusTest() {
```

```
214         RingTorus ex1 = new RingTorus("Small Example", 9.5, 1.25);
215         RingTorus ex2 = new RingTorus("Medium Example", 35.1, 10.4);
216         RingTorus ex3 = new RingTorus("Large Example", 134.28, 32.46);
217         RingTorus[] rtList = {ex1, ex2, ex3};
218         RingTorusList torusList = new RingTorusList("Array List", rtList, 3);
219
220         Assert.assertEquals("Delete object test: ", ex1,
221            torusList.deleteRingTorus("Small Example"));
222         RingTorus[] rtList2 = {};
223         RingTorusList torusList2 = new RingTorusList("Array List 2", rtList2, 0);
224         Assert.assertEquals("Find object test ", null,
225            torusList2.deleteRingTorus("Small Example"));
226      }
227      /**
228      * A test to check editRingTorus method.
229      */
230      @Test public void editRingTorusTest() {
231         RingTorus ex1 = new RingTorus("Small Example", 9.5, 1.25);
232         RingTorus ex2 = new RingTorus("Medium Example", 35.1, 10.4);
233         RingTorus ex3 = new RingTorus("Large Example", 134.28, 32.46);
234         RingTorus[] rtList = {ex1, ex2, ex3};
235         RingTorusList torusList = new RingTorusList("Array List", rtList, 3);
236
237         Assert.assertTrue("Edit object test: ",
238            torusList.editRingTorus("Small Example", 10.5, 2.3));
239         RingTorus[] rtList2 = {};
240         RingTorusList torusList2 = new RingTorusList("Array List 2", rtList2, 0);
241         Assert.assertFalse("Edit object test ",
242            torusList2.editRingTorus("Small Example", 10.5, 2.3));
243      }
244       /**
245      * A test to check findRingTorusWithLargestVolume method.
246      */
247      @Test public void findRingTorusWithLargestVolumeTest() {
248         RingTorus ex1 = new RingTorus("Small Example", 9.5, 1.25);
249         RingTorus ex2 = new RingTorus("Medium Example", 35.1, 10.4);
250         RingTorus ex3 = new RingTorus("Large Example", 134.28, 32.46);
251         RingTorus[] rtList = {ex1, ex2, ex3};
252         RingTorusList torusList = new RingTorusList("Array List", rtList, 3);
253
254         Assert.assertEquals("Largest volume test: ", ex3,
255            torusList.findRingTorusWithLargestVolume());
256         Assert.assertEquals("Largest volume test: ", null,
257            null);
258
259         RingTorus[] rtList2 = {};
260         RingTorusList torusList2 = new RingTorusList("Array List 2", rtList2, 0);
261         Assert.assertEquals("Largest volume test ", null,
262            torusList2.findRingTorusWithLargestVolume());
263      }
264
265
266   }
```

RingTorusTest.java

```java
import org.junit.Assert;
//import static org.junit.Assert.*;
import org.junit.Before;
import org.junit.Test;


/**
* Test file that tests whether the methods from
* RingTorus object are correct or not.
*
* Project_7B.
* @author Ssai Sanjanna Ganji - COMP 1210-006
* @version 10/21/22
*/
public class RingTorusTest {
   /** Fixture initialization (common initialization
    *  for all tests). **/
   @Before public void setUp() {
   }

   /**
   * A test to check getLabel method.
   */
   @Test public void getLabelTest() {
      RingTorus ex2 = new RingTorus("Medium Example", 35.1, 10.4);
      Assert.assertEquals("Get label test: ", "Medium Example", ex2.getLabel());
   }

   /**
   * A test to check getLargeRadius method.
   */
   @Test public void getLargeRadiusTest() {
      RingTorus ex2 = new RingTorus("Medium Example", 35.1, 10.4);
      Assert.assertEquals("Get large radius test: ",
         35.1, ex2.getLargeRadius(), 0.001);
   }

   /**
   * A test to check getSmallRadius method.
   */
   @Test public void getSmallRadiusTest() {
      RingTorus ex2 = new RingTorus("Medium Example", 35.1, 10.4);
      Assert.assertEquals("Get small radius test: ",
         10.4, ex2.getSmallRadius(), 0.001);
   }

   /**
   * A test to check setLabel method when 'if' condition comes out true.
   */
   @Test public void labelTrueTest() {
      RingTorus ex2 = new RingTorus("Medium Example", 35.1, 10.4);
      Assert.assertTrue("Label true test: ",
         ex2.setLabel("Medium Example"));
   }

  /**
   * A test to check setLabel method when 'if' condition comes out false.
```

```
57      */
58      @Test public void labelFalseTest() {
59         RingTorus ex2 = new RingTorus("Medium Example", 35.1, 10.4);
60         Assert.assertFalse("Label false test: ",
61            ex2.setLabel(null));
62      }
63
64     /**
65      * A test to check setLargeRadius method when 'if' condition comes out true.
66      */
67      @Test public void largeRadiusTrueTest() {
68         RingTorus ex2 = new RingTorus("Medium Example", 35.1, 10.4);
69
70         // Large radius satisfies all of the 'if' conditions --> test passes
71         Assert.assertTrue("Large radius true test: ",
72            ex2.setLargeRadius(35.1));
73
74         // Large radius is not a positive number --> test fails
75         Assert.assertFalse("Large radius false test: ",
76            ex2.setLargeRadius(-3.0));
77
78         // Large radius is less than small radius --> test fails
79         Assert.assertFalse("Large radius false test: ",
80            ex2.setLargeRadius(9.8));
81      }
82
83      /**
84      * A test to check setLargeRadius method when 'if' condition comes out false.
85      */
86      @Test public void largeRadiusFalseTest() {
87         RingTorus ex2 = new RingTorus("Medium Example", 35.1, 10.4);
88         Assert.assertFalse("Large radius false test: ",
89            ex2.setLargeRadius(9.6));
90      }
91
92      /**
93      * A test to check setSmallRadius method when 'if' condition comes out true.
94      */
95      @Test public void smallRadiusTrueTest() {
96         RingTorus ex2 = new RingTorus("Medium Example", 35.1, 10.4);
97
98         // Small radius satisfies all of the 'if' conditions --> test passes
99         Assert.assertTrue("Small radius true test: ",
100           ex2.setSmallRadius(10.4));
101
102        // Small radius is not a positive number --> test fails
103        Assert.assertFalse("Small radius false test: ",
104           ex2.setSmallRadius(-0.9));
105
106        // Small radius is greater than large radius --> test fails
107        Assert.assertFalse("Small radius false test: ",
108           ex2.setSmallRadius(40.6));
109     }
110
111     /**
112     * A test to check setSmallRadius method when 'if' condition comes out false.
```

```
113      */
114      @Test public void smallRadiusFalseTest() {
115         RingTorus ex2 = new RingTorus("Medium Example", 35.1, 10.4);
116         Assert.assertFalse("Small radius false test: ",
117            ex2.setSmallRadius(-8));
118      }
119
120      /**
121      * A test to check diameter method.
122      */
123      @Test public void diameterTest() {
124         RingTorus ex2 = new RingTorus("Medium Example", 35.1, 10.4);
125         Assert.assertEquals("Diameter test: ", 91.0, ex2.diameter(), 0.001);
126      }
127
128      /**
129      * A test to check surfaceArea method.
130      */
131      @Test public void surfaceAreaTest() {
132         RingTorus ex2 = new RingTorus("Medium Example", 35.1, 10.4);
133         Assert.assertEquals("Surface area test: ", 14411.202,
134            ex2.surfaceArea(), 0.001);
135      }
136
137    /**
138      * A test to check volume method.
139      */
140      @Test public void volumeTest() {
141         RingTorus ex2 = new RingTorus("Medium Example", 35.1, 10.4);
142         Assert.assertEquals("Volume test: ", 74938.248,
143            ex2.volume(), 0.001);
144      }
145
146      /**
147      * A test to check ToString method.
148      */
149      @Test public void toStringTest() {
150         RingTorus ex2 = new RingTorus("Medium Example", 35.1, 10.4);
151         Assert.assertTrue("To string test: ",
152            ex2.toString().contains("\"Medium Example\""));
153      }
154
155      /**
156      * A test to check getCount method.
157      */
158      @Test public void getCountTest() {
159         RingTorus ex2 = new RingTorus("Medium Example", 35.1, 10.4);
160         ex2.resetCount();
161         Assert.assertEquals("Get count test: ", 0, ex2.getCount(), 0.001);
162      }
163
164      /**
165      * A test to check equals method when 'if' condition comes out false.
166      */
167      @Test public void equalsFalseTest() {
168         RingTorus ex2 = new RingTorus("Medium Example", 35.1, 10.4);
```

```
169         Assert.assertFalse("Equals false label test: ", ex2.equals(0));
170      }
171
172      /**
173      * A test to check equals method when 'if' condition comes out true.
174      */
175      @Test public void equalsTrueTest() {
176         RingTorus ex2 = new RingTorus("Medium Example", 35.1, 10.4);
177
178         // Creating new object same as 'ex2'
179         RingTorus rt = (RingTorus) ex2;
180         // New object with a different label
181         RingTorus ex4 = new RingTorus("Moderate Example", 35.1, 10.4);
182         // New object with a different large radius
183         RingTorus ex5 = new RingTorus("Medium Example", 37.9, 10.4);
184         // New object with a different small radius
185         RingTorus ex6 = new RingTorus("Medium Example", 35.1, 12.3);
186
187         // Label, large radius, small radius matches --> test passes
188         Assert.assertTrue("Equals true test: ", ex2.equals(rt));
189         // Label is not same --> test fails
190         Assert.assertFalse("Label not matchingtest: ", ex2.equals(ex4));
191         // Large radius is not same --> test fails
192         Assert.assertFalse("Large radius not matching: ", ex2.equals(ex5));
193         // Small radius is not same --> test fails
194         Assert.assertFalse("Small radius not matching: ", ex2.equals(ex6));
195      }
196
197      /**
198      * A test to check hashCode method.
199      */
200      @Test public void hashCodeTest() {
201         RingTorus ex2 = new RingTorus("Medium Example", 35.1, 10.4);
202         Assert.assertEquals("Hash code test: ", 0, ex2.hashCode());
203      }
204
205      /**
206      * A test to check compareTo method.
207      */
208      @Test public void compareToTest() {
209         RingTorus ex2 = new RingTorus("Medium Example", 35.1, 10.4);
210         RingTorus ex1 = new RingTorus("Small Example", 9.5, 1.25);
211         RingTorus ex3 = new RingTorus("Large Example", 134.28, 32.46);
212         RingTorus rt = (RingTorus) ex2;
213
214         // Volume of 'ex2' is less than volume of 'ex3' --> 'if' executes
215         Assert.assertEquals("compareTo if test: ", -1, ex2.compareTo(ex3));
216         // Volume of 'ex2' is greater than volume of 'ex4' --> 'else if' executes
217         Assert.assertEquals("compareTo else if test: ", 1, ex2.compareTo(ex1));
218         // Volume of 'ex2' is equal to volume of 'rt' --> 'else' executes
219         Assert.assertEquals("compareTo else test: ", 0, ex2.compareTo(rt));
220      }
221
222   }
```

< Back to Summary