1.1

1.2

1.3

1.4

1.5

1.6

1.7

```
2 * Devante Wilson
    * November 2, 2015
    * Class defines static methods
 5
 6 * to be utilized by other classes
 9 public class Geometry
10 {
     public static double cubeVolume(double h)
119
12
13
          double cuVolume = Math.pow(h, 3);
14
          return cuVolume;
15
      }
16
     public static double cubeSurface(double h)
17⊖
18
      {
19
           double cuSurface = 6 * Math.pow(h, 2);
20
          return cuSurface;
      }
21
22
23⊝
      public static double sphereVolume(double r)
24
25
           double sVolume = (4/3.0) * Math.PI * Math.pow(r, 3);
26
          return sVolume;
27
     }
28
29@ public static double sphereSurface(double r)
30
      -{
31
          double sSurface = 4 * Math.PI * Math.pow(r, 2);
32
          return sSurface;
33
      }
34
35⊖
     public static double cylinderVolume(double r, double h)
36
37
          double cyVolume = Math.PI * Math.pow(r, 2) * h;
38
          return cyVolume;
39
40
41⊖
     public static double cylinderSurface(double r, double h)
42
43
          double cySurface = (2 * Math.PI * r) * (h + r);
44
          return cySurface;
45
      }
46
47⊖
       public static double coneVolume(double r, double h)
48
49
           double coVolume = Math.PI * Math.pow(r, 2) * (h/3.0);
50
           return coVolume;
51
52
53⊖
       public static double coneSurface(double r, double h)
54
          double coSurface = (Math.PI * r) * (r + Math.sqrt(Math.pow(h,2) + Math.pow(r, 2)));
55
56
          return coSurface;
57
       }
```

```
8 // import packages
 9 import java.util.Scanner;
 10
 11 public class GeometryTester
 12 {
 13⊝
         public static void main(String[] args)
 14
 15
             // define variables/objects
 16
             Geometry geo = new Geometry();
17
             Scanner scan = new Scanner (System.in);
 18
             double r = 0, h = 0;
 19
            // prompt user for values
 20
 21
             System.out.print("Please enter a radius: ");
 22
             r = scan.nextDouble();
 23
             System.out.print("Please enter a height: ");
 24
            h = scan.nextDouble();
 25
 26
             // call methods and print results
 27
             System.out.printf("Cube volume: %.2f" +
 28
                     "\nCube surface area: %.2f" +
 29
                     "\nSphere volume: %.2f" +
 30
                     "\nSphere surface area: %.2f" +
 31
                     "\nCylinder volume: %.2f" +
 32
                     "\nCylinder surface area: %.2f" +
                     "\nCone volume: %.2f" +
 33
 34
                     "\nCone surface: %.2f",
%35
                     geo.cubeVolume(h),
₹36
                     geo.cubeSurface(h),
37
                     geo.sphereVolume(r),
38
                     geo.sphereSurface(r),
%39
                     geo.cylinderVolume(r,h),
%40
                     geo.cylinderSurface(r,h),
41
                     geo.coneVolume(r, h),
W42
                     geo.coneSurface(r,h));
 43
 44 }
 45
 <
🖳 Console 🛭
<terminated> GeometryTester [Java Application] C:\Software\IBM Eclipse\ecl
Please enter a radius: 4
Please enter a height: 5
Cube volume: 125.00
Cube surface area: 150.00
Sphere volume: 268.08
Sphere surface area: 201.06
Cylinder volume: 251.33
Cylinder surface area: 226.19
Cone volume: 83.78
Cone surface: 130.73
```

Devante Wilson Group 7

Assignment 3 - OOP

2.

2.1

2.2

```
10 / **
  2 * Devante Wilson
  3 * November 3, 2015
     * cube subclass - question 2.
  8 public class Cube extends Geo
  9 {
△10⊝
        public double volume (double h)
11
 12
            double cuVolume = Math.pow(h, 3);
 13
            return cuVolume;
 14
 15
△16⊖
        public double surfaceArea(double h)
 17
            double cuSurface = 6 * Math.pow(h, 2);
18
 19
            return cuSurface;
20
 21
22
        // must implement - but not used in this class
△23⊝
        public double volume (double r, double h)
 24
 25
            return 0:
26
 27
28
        // must implement - but not used in this class
△29⊝
        public double surfaceArea(double r, double h)
3.0
31
            return 0;
 32
33 }
```

```
10/**
  2 * Devante Wilson
  3 * November 3, 2015
     * sphere subclass - question 2.
  8 public class Sphere extends Geo
  9 {
△10⊝
        public double volume (double r)
11
            double sVolume = (4/3.0) * Math.PI * Math.pow(r, 3);
 13
            return sVolume;
 14
 15
△16⊝
        public double surfaceArea(double r)
 17
            double sSurface = 4 * Math.PI * Math.pow(r, 2);
 18
 19
            return sSurface;
 20
 21
        // must implement - but not used in this class
22
23⊝
        public double volume (double r, double h)
 24
            return 0;
26
 27
 28
        // must implement - but not used in this class
△29⊝
        public double surfaceArea(double r, double h)
 30
 31
            return 0;
 32
33 }
```

2.3

```
19/**
                                                                    10 / **
                                                                      * Devante Wilson
  2 * Devante Wilson
                                                                       * November 3, 2015
  3 * November 3, 2015
                                                                       * cone subclass - question 2.
     * cylinder subclass - question 2.
                                                                    8 public class Cone extends Geo
  8 public class Cylinder extends Geo
                                                                         public double volume(double r, double h)
  9 {
△10⊝
         public double volume (double r, double h)
                                                                             double coVolume = Math.PI * Math.pow(r. 2) * (h/3.0):
                                                                             return coVolume;
 12
             double cyVolume = Math.PI * Math.pow(r, 2) * h;
 13
             return cyVolume;
                                                                         public double surfaceArea(double r, double h)
 14
 15
                                                                             double coSurface = (Math.PI * r) * (r + Math.sqrt(Math.pow(h,2) + Math.pow(r, 2)));
△16⊖
        public double surfaceArea(double r, double h)
                                                                             return coSurface;
 17
             double cySurface = (2 * Math.PI * r) * (h + r);
                                                                          // must implement - but not used in this class
             return cySurface;
 19
                                                                         public double volume(double x)
 20
 21
                                                                             return 0;
 22
         // must implement - but not used in this class
△23⊝
         public double volume(double x)
                                                                          // must implement - but not used in this class
 24
                                                                         public double surfaceArea(double x)
             return 0;
 26
                                                                              return 0;
 27
28
         // must implement - but not used in this class
△29⊝
         public double surfaceArea(double x)
30
             return 0;
32
33 }
```

```
public static void main(String[] args)
 14
        {
 15
             // define variables/objects
 16
             Geometry geo = new Geometry();
 17
            Cube cu = new Cube();
 18
            Sphere s = new Sphere();
 19
            Cylinder cy = new Cylinder();
 20
            Cone co = new Cone();
 21
            Scanner scan = new Scanner(System.in);
 22
            double r = 0, h = 0;
 23
 24
            // prompt user for values
 25
            System.out.print("Please enter a radius: ");
 26
            r = scan.nextDouble();
 27
            System.out.print("Please enter a height: ");
 28
            h = scan.nextDouble();
 29
 30
            // close Scanner object
 31
             scan.close();
 32
 33
            // call methods and print results - from Geometry class
 34
             System.out.printf("Cube volume: %.2f" +
 35
                    "\nCube surface area: %.2f" +
 36
                     "\nSphere volume: %.2f" +
 37
                     "\nSphere surface area: %.2f" +
 38
                     "\nCylinder volume: %.2f" +
                                                                              OUTPUT
 39
                     "\nCylinder surface area: %.2f" +
 40
                     "\nCone volume: %.2f" +
                                                                  Please enter a radius: 4
 41
                    "\nCone surface: %.2f",
                                                                  Please enter a height: 5
42
                    geo.cubeVolume(h),
                                                                  Cube volume: 125.00
43
                    geo.cubeSurface(h),
                                                                  Cube surface area: 150.00
M44
                    geo.sphereVolume(r),
                                                                  Sphere volume: 268.08
M45
                    geo.sphereSurface(r),
№46
                                                                  Sphere surface area: 201.06
                     geo.cylinderVolume(r,h),
47
                                                                  Cylinder volume: 251.33
                     geo.cylinderSurface(r,h),
48
                                                                  Cylinder surface area: 226.19
                    geo.coneVolume(r, h),
                                                                  Cone volume: 83.78
49
                    geo.coneSurface(r,h));
                                                                  Cone surface: 130.73
 50
 51
             // call methods and print results - from Geo class
             System.out.printf("\n\nCube volume: %.2f" +
 52
                                                                  Cube volume: 125.00
                                                                 Cube surface area: 150.00
 53
                    "\nCube surface area: %.2f" +
                                                                  Sphere volume: 268.08
 54
                     "\nSphere volume: %.2f" +
 55
                     "\nSphere surface area: %.2f" +
                                                                 Sphere surface area: 201.06
 56
                     "\nCylinder volume: %.2f" +
                                                                  Cylinder volume: 251.33
 57
                                                                  Cylinder surface area: 226.19
                     "\nCylinder surface area: %.2f" +
                                                                  Cone volume: 83.78
 58
                     "\nCone volume: %.2f" +
                                                                  Cone surface: 130.73
 59
                    "\nCone surface: %.2f",
 60
                    cu.volume(h),
 61
                    cu.surfaceArea(h),
 62
                    s.volume(r),
 63
                    s.surfaceArea(r),
 64
                    cy.volume(r,h),
 65
                    cy.surfaceArea(r,h),
 66
                    co.volume(r, h),
 67
                    co.surfaceArea(r,h));
```

3. & 4.

```
10/**
  2 * Devante Wilson
 3 * November 3, 2015
    * Class implements the java.awt.Rectangle class
  5
  6 * and supplies methods to compute
 7
    * the area and perimeter of a rectangle
 8
 9
 10 // import rectangle class
 11 import java.awt.Rectangle;
 12
13 public class BetterRectangle extends Rectangle
 14 {
 15
        // compute rectangle perimeter
 16⊖
        public double getPerimeter()
 17
           double perimeter = 2 * (width + height);
 18
 19
           return perimeter;
 20
       }
 21
 22
       // compute rectangle area
 23⊝
        public double getArea()
 24
 25
           double area = width * height;
 26
          return area;
 27
       }
 28
 29
        // constructor
 30⊝
        public BetterRectangle(int x, int y, int width, int height)
 31
 32
           // create Rectangle object (invoke constructor of superclass)
№33
           Rectangle rect = new Rectangle();
 34
 35
           // set the location of the rectangle
 36
          setLocation(x, y);
 37
           // set the size of the rectangle
           setSize(width, height);
 38
 39
       }
 40
 41
        // default no parameter constructor
 42⊖
        public BetterRectangle()
 43
        {
 44
 45
       }
46 }
```

```
Group 7
```

```
* Devante Wilson
  3 * November 3, 2015
  5 * Class tests the BetterRectangle class
  6
  7
  8 // import package(s)
🈘 9⊕ import java.util.Scanner;[.]
 11
 12 public class BetterRectangleTester
 14⊖
         public static void main(String[] args)
 15
 16
             // define variables/objects
17
             Scanner scan = new Scanner(System.in);
 18
             int width = 0, height = 0, x = 0, y = 0;
 19
            BetterRectangle rect;
 20
 21
             // prompt user for rectangle information
 22
             System.out.print("Enter rectangle width: ");
 23
            width = scan.nextInt();
 24
             System.out.print("Enter rectangle height: ");
 25
             height = scan.nextInt();
 26
            System.out.print("Enter rectangle position x: ");
 27
            x = scan.nextInt();
 28
             System.out.print("Enter rectangle position y ");
 29
            y = scan.nextInt();
 30
            rect = new BetterRectangle(x,y,width,height);
 31
 32
             // call methods and print results
 33
             System.out.printf("Rectangle perimeter: %.2f", rect.getPerimeter());
 34
             System.out.printf("\nRectangle area: %.2f", rect.getArea());
 35
             rect.setSize(width, height);
 36
             rect.setLocation(x,y);
 37
 38
             System.out.println("\n" + rect.getSize());
 39
             System.out.println(rect.getLocation());
 40
         }
 41 }
🖳 Console 🛭
<terminated> BetterRectangleTester [Java Application] C:\Software\IBM Eclipse\eclipseDevelopmentPackage\
Enter rectangle width: 2
Enter rectangle height: 4
Enter rectangle position x: 1
Enter rectangle position y 2
Rectangle perimeter: 12.00
Rectangle area: 8.00
java.awt.Dimension[width=2,height=4]
java.awt.Point[x=1,y=2]
```

```
10/**
 2 * Devante Wilson
 3 * November 3, 2015
 4
 5 * Program models a circuit.
   * This class is a superclass.
7
   */
9 public class Circuit
10 {
110 public double getResistance()
12
13
          return 0;
14
      }
15 }
```

```
10 / **
 2 * Devante Wilson
 3 * November 3, 2015
    * Program models a circuit.
 6 * This subclass represents a single resistor.
 7
 9 public class Resistor extends Circuit
10 {
11 // class variable
      private double resistance;
12
13
     // constructor/mutator method
14
      public Resistor(double r)
15⊝
16
17
         resistance = r;
18
      }
19
    // accessor method
20
      public double getResistance()
≙21⊖
22
23
      return resistance;
24
      }
25 }
```

```
2 * Devante Wilson
 3 * November 3, 2015
 4
 5 * Program models a circuit.
 6
    * This subclass models a set of circuits in parallel.
 7 */
 8
 9 // import package(s)
10 import java.util.ArrayList;
11
12 public class Parallel extends Circuit
13 {
14
       // declare new ArrayList
15
      private ArrayList<Circuit> parallel;
16
      // constructor
17
189 public Parallel()
19
      {
20
         parallel = new ArrayList<Circuit>();
21
      }
22
23
      // method to add resistors
      public void add(Circuit c)
24⊖
25
      {
26
          parallel.add(c);
27
      }
28
29
      // compute combined resistance
△30⊝
     public double getResistance()
      {
31
32
          // define variables/objects
33
         double rTotal = 0, rVal;
         Circuit resist;
34
35
36
          for (int a = 0; a < parallel.size(); a++)
37
38
              resist = parallel.get(a); // retrieve arrayList object
39
              rVal = resist.getResistance(); // call mutator method from Resistor class
40
              rTotal += 1.0/rVal;
41
          }
42
          if (rTotal != 0) // prevent divide by 0
43
44
              rTotal = 1.0/rTotal;
45
46
          return rTotal;
47
       }
48 }
```

```
19/**
 2 * Devante Wilson
 3 * November 3, 2015
 5 * Program models a circuit.
    * This subclass models a series of circuits.
 7 */
 9 // import package(s)
10 import java.util.ArrayList;
11
12 public class Serial extends Circuit
13 {
14
       // define new ArrayList
15
      ArrayList<Circuit> serial;
16
17
      // constructor
     public Serial()
18⊝
19
      {
20
           serial = new ArrayList<Circuit>();
21
      }
22
23
      // method to add resistors
24⊖
     public void add(Circuit c)
25
      {
26
           serial.add(c);
27
      }
28
29
      // compute combined resistance
△30⊝
       public double getResistance()
31
32
           double rTotal = 0, rVal;
33
           Circuit resist;
34
35
          for (int b = 0; b < serial.size(); b++)</pre>
36
37
               resist = serial.get(b); // retrieve arrayList object
38
               rVal = resist.getResistance(); // call mutator method from Resistor class
39
               rTotal += rVal;
40
41
42
          return rTotal;
43
       }
44 }
```

```
10/**
 2 * Devante Wilson
 3 * November 6, 2015
 5 * Class is a tester for Circuit program
 6 */
 8 public class CircuitTester
 9 {
10⊝
        public static void main(String[] args)
 11
 12
             // create objects
13
            Parallel p = new Parallel();
14
            Serial s = new Serial();
15
16
           p.add(new Resistor(100));
17
            s.add(new Resistor(100));
 18
            s.add(new Resistor(200));
 19
           p.add(s);
 20
21
           // print result
22
            System.out.println("Combined resistance: " + p.getResistance());
23
            System.out.println("Expected: 75.0");
24
        }
 25 }
 26
<terminated> CircuitTester [Java Application] C:\Software\IBM Eclipse\eclipseDevelopmentPackage\ibm_sdk80\bin\javaw.exe (Nov 14, 2015, 7:08
Combined resistance: 75.0
Expected: 75.0
```

6. 10 /** 2 * Devante Wilson 3 * November 5, 2015 4 * * Program computes the alternating sum of an array's elements. 6 * (Operator switches from - to + between each element) 7 */ 8 9 // import package 10 import java.util.Arrays; 11 12 public class Q6 13 [14⊖ static int alternatingSum(int[] arr) 15 { 16 int sum = 0;17 18 // calculute alternating sum 19 for (int a = 0; a < arr.length; a+=2)</pre> 20 sum += arr[a]; 21 22 for (int b = 1; b < arr.length; b+=2)</pre> 23 sum -= arr[b]; 24 25 return sum; 26 } 27 28⊖ public static void main(String[] args) 29 30 // define array 31 int[] sumArr = {1,4,9,16,9,7,4,9,11}; 32 33 // call method and print result 34 System.out.print("Array: " + Arrays.toString(sumArr) + 35 "\nAlternating sum: " + alternatingSum(sumArr) + 36 "\nExpected: -2"); 37 } 38 } 39 <terminated> Q6 [Java Application] C:\Software\IBM Eclipse\eclipseDevelopmentPackage\ibm_sdk80\bin\ Array: [1, 4, 9, 16, 9, 7, 4, 9, 11] Alternating sum: -2 Expected: -2

```
10 /**
  2 * Devante Wilosn
  3
     * November 5, 2015
  5
     * Program generates a sequence of 20 random values from 0-99
  6
     * prints the sequence, sorts it, and prints the sorted sequence.
  7
 8
 9 // import package(s)
 10@ import java.util.Random;
 11 import java.util.Arrays;
 12
 13 public class Q7
 14 {
 15⊖
        public static void main(String[] args)
 16
 17
            // create variables/objects
 18
           Random randomInt = new Random();
 19
            int[] randomArr = new int[20];
 20
 21
           System.out.print("Unsorted: ");
 22
 23
            // generate sequence of random integers
 24
            for (int i = 0; i < randomArr.length; i++)
 25
 26
                // store values in array
 27
                randomArr[i] = randomInt.nextInt(100);
 28
 29
                // print random sequence
 30
                System.out.print(randomArr[i] + " ");
 31
            }
 32
 33
            // sort array
 34
            Arrays.sort(randomArr);
 35
 36
            System.out.print("\nSorted: ");
 37
           for (int j = 0; j < randomArr.length; j++)
 38
 39
            {
 40
                // print sorted array
 41
                System.out.print(randomArr[j] + " ");
 42
            }
 43
        }
 44 }
🖳 Console 🛭
```

<terminated> Q7 [Java Application] C:\Software\IBM Eclipse\eclipseDevelopmentPackage\ibm_sdk80\bin\ja
Unsorted: 45 85 2 45 19 4 29 30 61 67 67 22 33 79 51 90 52 0 26 60
Sorted: 0 2 4 19 22 26 29 30 33 45 45 51 52 60 61 67 67 79 85 90

```
10/**
 2 * Devante Wilson
3 * November 6, 2015
 5 * Class provides the nominal and actual values
6 * for the voltage divider's gain.
7
8
9 public class VoltageDivider
10 {
11
      // instance variables
12
     private Resistor2 r1, r2;
13
14⊖
     /* constructor:
      * accepts two Resistor objects,
15
16
       * nominal resistance and resistor tolerance.
17
       */
     public VoltageDivider(Resistor2 r1, Resistor2 r2)
18⊖
19
20
         this.r1 = r1;
21
          this.r2 = r2;
22
     }
23
24
     // retrieve nominal value of voltage divider's gain
25⊖
      public double getNominalGain()
26
     {
27
         return r1.getNominal()/(r1.getNominal() + r2.getNominal());
28
29
30
      // retrieve actual value of voltage divider's gain
31⊖
     public double getActualGain()
32
      {
33
         return r1.getActual()/(r1.getActual() + r2.getActual());
34
      }
35
36
     // default no parameter constructor
37⊖
      public VoltageDivider()
38
       {
39
40
41 }
```

```
9 // import package(s)
 10 import java.util.concurrent.ThreadLocalRandom;
 11
 12 public class Resistor2
 13 {
 14
       // define instance variables
      private double nominalResistance, tolerance, actualResistance;
 15
 16
      /* constructor:
 17⊖
 18
        * determines actual resistance value randomly
 19
        * given nominal resistance and tolerance (+/-5\%)
        */
 20
 21 public Resistor2 (double nominal, double tolerance)
 22
 23
           // determine actual value randomly
 24
          actualResistance = ThreadLocalRandom.current().nextDouble(
 25
                  nominal - (nominal * (tolerance/100.0)),
 26
                  nominal * ((tolerance/100.0) + 1));
 27
 28
          nominalResistance = nominal;
 29
           this.tolerance = tolerance;
 30
      }
 31
 32
       // Default no parameter constructor
 33⊖
     public Resistor2()
 34
       -{
 35
 36
       }
 37
 38
       // get nominal resistance
       public double getNominal()
 39⊝
 40
      {
 41
           return nominalResistance;
 42
      }
 43
 44
       // get resistance tolerance
 450 public double getTolerance()
 46
 47
        return tolerance;
 48
      }
 49
 50
       // get actual resistance
 51@ public double getActual()
 52
       {
 53
           return actualResistance;
 54
 55 }
```

9: 0.2526 10: 0.2577

```
10/**
2 * Devante Wilson
3 * November 7, 2015
 5 * Class tests the VoltageDivider and Resistor classes.
 6 * Question 8
 8
 9 public class GainTester
10 {
119 public static void main(String[] args)
12
13
          final int circuitAmount = 10; // amount of voltage dividers
14
15
        // output actual and nominal gains
for (int i = 1; i <= circuitAmount; i++)</pre>
16
 17
 18
               Resistor2 r1 = new Resistor2(250, 5);
 19
               Resistor2 r2 = new Resistor2 (750, 5);
 20
               VoltageDivider divider = new VoltageDivider(r1, r2);
 21
22
              if (i == 1)
23
              System.out.print("Nominal Gain (for all Resistors): " +
                      divider.getNominalGain() + " Ohms\n\nActual Gains (Ohms):\n\n");
24
25
26
              System.out.printf(i + ": %.4f", divider.getActualGain());
27
               System.out.print("\n");
28
          }
29
      }
30 }
Nominal Gain (for all Resistors): 0.25 Ohms
Actual Gains (Ohms):
1: 0.2585
2: 0.2449
3: 0.2492
4: 0.2546
5: 0.2562
6: 0.2425
7: 0.2408
8: 0.2434
```