```
/**
   Source Code: BRectTester.java
   Author: Alp Karavil
   Student ID: 5827197
   Assignment: Program #4 BetterRectangle

   Course: 3337-Programming II
   Section: U09
   Instructor: William Feild
   Due Date: 10/18/2018 - Beginning of class

   I hereby certify that this collective work is my own
   and none of it is the work of any other person or entity

   Signature:


   Language: Java
   Compile/Run:
   javac BRectTester.java
   javac BetterRectangle.java
   java BRectTester

   ----------------------------------------------------------------

   Description:
   This is a tester class for the BetterRectangle class which extends the
   java.awt.Rectangle class. This tester creates four (4) better rectangles,
   using each of the required constructors. Next, utilizing one of the
   rectangles, this class executes all of the added accessor methods. Then,
   using two of the remaining rectangles, the Tester will execute the
   utility methods (and the equals() method), comparing the two rectangles.
   Finally, using the remaining rectangle, this tester will execute the mutator
   method.

   This tester will print the expected values while testing the accessor,
   utility, and mutator methods to make sure output is correct.

   Input:
   No input is needed, the created BetterRectangles objects are hard-coded
   into this tester class.

   Output:

   This program output will print information regarding to the 4 created
   BetterRectangle objects, then print information about BetterRectangle A
   using accessor methods, print expected values, then print utility checks
   between BetterRectangle B and C along with expected values, and then finally
   print mutator method information, along with expected values, that will
   be performed on BetterRectangle D.

   Process:
   1. Create 4 BetterRectangles with different constructors
   2. Print information about all 4
   3. Use and print accessor methods on first BetterRectangle
   4. Print expected values
   5. Use and print utility methods between second and third
   6. Print expected outcome
   7. Use and print mutator, scaleBy(multiplier) method, on last BetterRectangle
   8. Print expected outcome
```

```
61
62      No particular algorithms are used.
63
64      Known Bugs: None
65  **/
66
67  //Import Point class which is used for the midpoint of a BetterRectangle
68  import java.awt.Point;
69
70  public class BRectTester {
71      public static void main(String[] args) {
72          BetterRectangle bRectA = new BetterRectangle(3, 4);
73          BetterRectangle bRectB = new BetterRectangle(bRectA);
74          BetterRectangle bRectC = new BetterRectangle(1, 1, 4, 3);
75          BetterRectangle bRectD = new BetterRectangle();
76
77          //Print information about the BetterRectangles (toString() is called)
78          System.out.println("Rectangle A: " + bRectA.toString());
79          System.out.println("Rectangle B: " + bRectB.toString());
80          System.out.println("Rectangle C: " + bRectC.toString());
81          System.out.println("Rectangle D: " + bRectD.toString());
82          System.out.println();
83
84          System.out.println("Accessor methods being executed for Rectangle A...");
85          accessorCheck(bRectA, "A");
86
87          //Print expected values for the accessor check
88          System.out.println("Expected values:");
89          System.out.println("area: 12");
90          System.out.println("perimeter: 14");
91          System.out.println("slope: 1.33");
92          System.out.println("midpoint: (1,1)");
93          System.out.println();
94
95          System.out.println("Utility methods being executed for Rectangle B and "
96                          + "C...");
97          utilityCheck(bRectB, "B", bRectC, "C");
98
99          //Print the expected values for the utility check
100         System.out.println("Expected values:");
101         System.out.println("B is equal to C: false");
102         System.out.println("B is congruent to C: true");
103         System.out.println("B is equivalent to C: true");
104         System.out.println("B is similar to C: true");
105         System.out.println("B is concentric to C: false");
106
107         final int FIRST_MULTIPLIER = 4;
108         final int SECOND_MULTIPLIER = -4;
109         System.out.println("Mutator methods being executed for Rectangle D...");
110         mutatorCheck(bRectD, "D", FIRST_MULTIPLIER,
111                 SECOND_MULTIPLIER);
112
113         //Print expected values for the mutator check.
114         System.out.println("Expected values:");
115         System.out.println("[x=0,y=0,width=1,height=1][area=1,perimeter=4,"
116                 + "slope=1.0,mid-point=java.awt.Point[x=1,y=1]]");
117         System.out.println("Scale by 4 true");
118         System.out.println("Scale by -4 false");
119         System.out.println("[x=0,y=0,width=4,height=4][area=16,perimeter=16,"
120                 + "slope=1.0,mid-point=java.awt.Point[x=1,y=1]");
```

```
121         System.out.println();
122     }
123 
124     /**
125      * Calls and prints the accessor methods provided by the BetterRectangle
126      * class.
127      * @param rectangleInput BetterRectangle which will be accessed
128      * @param rectangleName Name of this BetterRectangle
129      */
130     private static void accessorCheck(BetterRectangle rectangleInput,
131                                       String rectangleName)
132     {
133         //Store all accessor methods
134         int area = rectangleInput.getArea();
135         int perimeter = rectangleInput.getPerimeter();
136         float slope = rectangleInput.getSlope();
137         Point midPoint = rectangleInput.getMidPoint();
138 
139         //Print stored accessor values
140         System.out.println(rectangleName + ": " + rectangleInput.toString());
141         System.out.println("area: " + area);
142         System.out.println("perimeter: " + perimeter);
143         System.out.println("slope: " + slope);
144         System.out.println("midpoint: " + midPoint.toString());
145         System.out.println();
146     }
147 
148     /**
149      * Calls and prints utility methods between BetterRectangle inputs.
150      * @param rectangle1 BetterRectangle that will be compared (1)
151      * @param rectangle1Name Name of BetterRectangle rectangle1 parameter
152      * @param rectangle2 BetterRectangle that will be compared (2)
153      * @param rectangle2Name Name of BetterRectangle rectangle2 parameter
154      */
155     private static void utilityCheck(BetterRectangle rectangle1,
156                                      String rectangle1Name,
157                                      BetterRectangle rectangle2,
158                                      String rectangle2Name)
159     {
160         //Store utility method results
161         boolean equalResult = rectangle1.equals(rectangle2);
162         boolean congruentResult = rectangle1.congruent(rectangle2);
163         boolean equivalentResult = rectangle1.equivalent(rectangle2);
164         boolean similarResult = rectangle1.similar(rectangle2);
165         boolean concentricResult = rectangle1.concentric(rectangle2);
166 
167         //Print utility method results
168         System.out.println(rectangle1Name + ": " + rectangle1.toString());
169         System.out.println(rectangle2Name + ": " + rectangle2.toString());
170 
171         System.out.println(rectangle1Name + " is equal to " + rectangle2Name
172                 + ": " + equalResult);
173         System.out.println(rectangle1Name + " is congruent to " + rectangle2Name
174                 + ": " + congruentResult);
175         System.out.println(rectangle1Name + " is equivalent to " + rectangle2Name
176                 + ": " + equivalentResult);
177         System.out.println(rectangle1Name + " is similar to " + rectangle2Name
178                 + ": " + similarResult);
179         System.out.println(rectangle1Name + " is concentric to " + rectangle2Name
180                 + ": " + concentricResult);
```

```
181         System.out.println();
182     }
183 
184     /**
185      * Calls and prints mutator method, scaleBy(multiplier), on inputted
186      * BetterRectangle.
187      * @param inputRectangle BetterRectangle that will be mutated
188      * @param rectangleName Name of the BetterRectangle input
189      * @param firstScale  Integer that will be used by the scaleBy(multiplier)
190      *                    method first.
191      * @param secondScale Integer that will be used by the scaleBy(multiplier)
192      *                    method second.
193      */
194     private static void mutatorCheck(BetterRectangle inputRectangle,
195                                      String rectangleName, int firstScale,
196                                      int secondScale)
197     {
198         //Print information about this BetterRectangle before mutation.
199         System.out.println(rectangleName + ": " + inputRectangle.toString());
200 
201         //Store multiplier values.
202         final int SCALE_MULTIPLIER_1 = firstScale;
203         final int SCALE_MULTIPLIER_2 = secondScale;
204 
205         //Call the mutator methods, and store boolean values regarding success.
206         boolean scaleResult1 = inputRectangle.scaleBy(SCALE_MULTIPLIER_1);
207         boolean scaleResult2 = inputRectangle.scaleBy(SCALE_MULTIPLIER_2);
208 
209         //Print information about mutator success
210         System.out.println("Scale by " + SCALE_MULTIPLIER_1 + " "
211                 + scaleResult1);
212         System.out.println("Scale by " + SCALE_MULTIPLIER_2 + " "
213                 + scaleResult2);
214         //Print information about this BetterRectangle after the mutation
215         System.out.println(rectangleName + ": " + inputRectangle.toString());
216         System.out.println();
217     }
218 }
```