For: Nick Schneider

Assignment: Exercise 13.5 Enable Geometric Objects Comparable

GitHub URL: <https://github.com/NickSchneider54/CIS171SchneiderN/tree/master/Project_2/Exercise%2013_5>

Student: Please answer the questions, then use the Insert, Screenshot option in Word to snip an appropriate sample of your executing program’s output.

Copy the code from your .java file(s) into the code section below. Your code should match the code submitted in GitHub.

Be sure to review your graded assignment for instructor comments!

|  |
| --- |
| **Analysis** |
| *Implemented the Comparable Interface into the GeometricObject class in order to compare two rectangles, circles, hexagons, …etc.* |
|  |

|  |
| --- |
| **Design** |
| *I took the code we worked on in class and implemented it into the GeometricObject class as a whole. I then added a Scanner for user input. I allowed the user to choose between a rectangle, circle, or hexagon to set for the first GeometricObject and did the same for the second. Therefore the program could compare any combination of two GeometricObjects.* |
|  |

|  |
| --- |
| **Testing** |
| *I began by running the programming without user input, hard coding the GeometricObjects. Once that worked I added a user input section for the user to decide what they compare.* |
|  |

|  |
| --- |
| **Screenshot(s)** |
|  |
|  |

|  |
| --- |
| **Code** |
| import java.util.\*;  /\*\*  \*  \* @author Nick Schneider  \* CIS171 Project\_2  \* EX-13\_5  \* Takes user input in sets of GeometricObjects to compare Area sizes  \* and then displays which object is LARGER, SMALLER, or if they are  \* the SAME SIZE. Any of the given Child-Classes of the GeometricObject  \* Super-Class can be compared (Rectangle, Circle, Hexagon).  \*/  public class TestGeometricObject  {  /\*\* Scanner to take user input \*/  protected static Scanner input = new Scanner(System.in);    /\*\* Array of GeometricObjects \*/  protected static GeometricObject shapes[] = new GeometricObject[2];    /\*\* Initializes the control for statements within the while loop\*/  protected static int counter = 0;    /\*\* Main method \*/  public static void main(String[] args)  {  // Initializes the control for while loop  String answer = "0";  /\*\* Takes user input for GeometricObjects to compare areas \*/  while(!answer.equals("N"))  {  /\*\* Initial statement when receiving GeometricObject 1 from user \*/  if(counter == 0)  {  System.out.println("Hello please enter a set of GeometricObjects to be Compared");  }  else if(counter >= 1)  {  System.out.println();  System.out.println("Please enter another GeometricObject");  }  System.out.println();  System.out.println("What kind of GeometricObject would you like to enter?");  System.out.println("(R)ectangle, (C)ircle, (H)exagon");  answer = input.nextLine();    if(answer.equals("R"))  {  initiateRectangle();  }  else if(answer.equals("C"))  {  initiateCircle();  }  else if(answer.equals("H"))  {  initiateHexagon();  }  /\*\* Prints GeometricObject information and compares Areas once two GeometricObjects have been entered \*/  if(counter == 1)  {  for(int i = 0; i < shapes.length; i++)  {  System.out.println();  System.out.println("GeometricObject " + (i + 1));  displayGeometricObject(shapes[i]);  }  int compare = shapes[0].compareTo(shapes[1]);  System.out.println();    if(compare == 1)  {  System.out.println("Geometric Object 1 is LARGER than Geometric Object 2");  }  else if(compare == -1)  {  System.out.println("Geometric Object 1 is SMALLER than Geometric Object 2");  }  else  {  System.out.println("Geometric Object 1 is the SAME SIZE as Geometric Object 2");  }  }  if(counter >= 1)  {  System.out.println();  System.out.println("Would you like to enter another set of GeometricObject to Compare?");  System.out.println("(Y)es or (N)o");  answer = input.nextLine();  }  counter += 1;  /\*\* Resets the counter to refill the shapes Array \*/  if(answer.equals("Y"))  {  counter = 0;  System.out.println();  }  }    }  /\*\* A method for comparing the areas of two geometric objects \*/  public static boolean equalArea(GeometricObject object1,GeometricObject object2)  {  return object1.getArea() == object2.getArea();  }  /\*\* A method for displaying a geometric object \*/  public static void displayGeometricObject(GeometricObject object)  {  System.out.println("The area is " + object.getArea());  System.out.println("The perimeter is " + object.getPerimeter());  }  /\*\* A method for initializing a Rectangle GeometricObject \*/  public static void initiateRectangle()  {  System.out.print("Width: ");  int width = input.nextInt();  System.out.print("Height: ");  int height = input.nextInt();  input.nextLine();  shapes[counter] = new Rectangle(width, height);  }  /\*\* A method for initializing a Circle GeometricObject \*/  public static void initiateCircle()  {  System.out.print("Radius: ");  int radius = input.nextInt();  input.nextLine();  shapes[counter] = new Circle(radius);  }  /\*\* A method for initializing a Hexagon GeometricObject \*/  public static void initiateHexagon()  {  System.out.print("Side Size: ");  int sideSize = input.nextInt();  input.nextLine();  shapes[counter] = new Hexagon(sideSize);  }    }  public abstract class GeometricObject implements Comparable<GeometricObject>  {  private String color = "white";  private boolean filled;  private java.util.Date dateCreated;  /\*\* Construct a default geometric object \*/  protected GeometricObject()  {  dateCreated = new java.util.Date();  }  /\*\* Construct a geometric object with color and filled value \*/  protected GeometricObject(String color, boolean filled)  {  dateCreated = new java.util.Date();  this.color = color;  this.filled = filled;  }  /\*\* Return color \*/  public String getColor()  {  return color;  }  /\*\* Set a new color \*/  public void setColor(String color)  {  this.color = color;  }  /\*\* Return filled. Since filled is boolean,  \* the get method is named isFilled \*/  public boolean isFilled()  {  return filled;  }  /\*\* Set a new filled \*/  public void setFilled(boolean filled)  {  this.filled = filled;  }  /\*\* Get dateCreated \*/  public java.util.Date getDateCreated()  {  return dateCreated;  }  @Override  public String toString()  {  return "created on " + dateCreated + "\ncolor: " + color +  " and filled: " + filled;  }    /\*\* Abstract method getArea \*/  public abstract double getArea();  /\*\* Abstract method getPerimeter \*/  public abstract double getPerimeter();    @Override /\*\* Compares the areas of two GeometricObjects \*/  public int compareTo(GeometricObject o)  {  if (this.getArea() > o.getArea())  return 1; //return positive value    else if(this.getArea() <o.getArea())  return -1;    else  return 0;  }    }  public class Rectangle extends GeometricObject  {  private double width;  private double height;  public Rectangle()  {  }  public Rectangle(double width, double height)  {  this.width = width;  this.height = height;  }  /\*\* Return width \*/  public double getWidth()  {  return width;  }  /\*\* Set a new width \*/  public void setWidth(double width)  {  this.width = width;  }  /\*\* Return height \*/  public double getHeight()  {  return height;  }  /\*\* Set a new height \*/  public void setHeight(double height)  {  this.height = height;  }  @Override /\*\* Return area \*/  public double getArea()  {  return width \* height;  }  @Override /\*\* Return perimeter \*/  public double getPerimeter()  {  return 2 \* (width + height);  }  }  public class Hexagon extends GeometricObject  {  private int sideSize;    /\*\*  \* default no-arg constructor  \*/  public Hexagon()  { }  /\*\*  \*  \* @param sideSize  \*/  public Hexagon(int sideSize)  {  this.sideSize = sideSize;  }    /\*\* Sets the sideSize \*/  public void setSideSize(int sideSize)  {  this.sideSize = sideSize;  }    /\*\* Returns sideSize \*/  public int getSideSize()  {  return sideSize;  }    @Override /\*\* Return area \*/  public double getArea()  {  return sideSize\*60;  }    @Override /\*\* Return perimeter \*/  public double getPerimeter()  {  return sideSize\*6.0;  }  }  public class Circle extends GeometricObject  {  private double radius;  public Circle()  {  }  public Circle(double radius)  {  this.radius = radius;  }  /\*\* Return radius \*/  public double getRadius()  {  return radius;  }  /\*\* Set a new radius \*/  public void setRadius(double radius)  {  this.radius = radius;  }  @Override /\*\* Return area \*/  public double getArea()  {  return radius \* radius \* Math.PI;  }  /\*\* Return diameter \*/  public double getDiameter()  {  return 2 \* radius;  }  @Override /\*\* Return perimeter \*/  public double getPerimeter()  {  return 2 \* radius \* Math.PI;  }  /\* Print the circle info \*/  public void printCircle()  {  System.out.println("The circle is created " + getDateCreated() +  " and the radius is " + radius);  }    } |