|  |  |
| --- | --- |
|  | |
| Create Rectangle Classes | |
| **Updated** | **10.28.2018 2:22 PM** |

# **Problem 1:**

Create a class Rectangle. The class has attributes length and width, each of which defaults to 1. It has methods that calculate the perimeter and the area of the rectangle. It has set and get methods for both length and width. The set methods should verify that length and width are each floating-point numbers larger than 0.0 and less than 20.0. Write a program to test class Rectangle.

# **Approach**

I built upon the assignment for week one except to focus on a rectangle exclusively.

# **Design**

**I did not have enough time to create a flow chart**

TestRectangle.java creates the object myRectangleX and sets the length and width

setLength verifies that height is larger than 0.0 and less than 20.0

IF not throw illegal exception argument

ELSE sets this.length

setLength verifies that width is larger than 0.0 and less than 20.0

IF not throw exception argument

ELSE sets this.width

getLength will return this.length

getWidth will return this.width

getArea will return perimeter

Formula for area is width multiplied by height: A = W \* H

getPerimeter will return perimeter

Formula for area is 2\*width plus 2\*Height: P = 2W + 2H

String to string returns: Length, width, area, perimeter

System.out.println returns results to user

EXIT

# **Testing**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Test** | **Expected Result** | **Actual Result** |
| 1 | Enter positive float numbers larger than 20.0 for Length | IllegalArgumentException | IllegalArgumentException |
| 2 | Enter negative float numbers less than 0.0 for Length | IllegalArgumentException | IllegalArgumentException |
| 3 | Enter positive float numbers larger than 20.0 for Width | IllegalArgumentException | IllegalArgumentException |
| 4 | Enter negative float numbers less than 0.0 for Width | IllegalArgumentException | IllegalArgumentException |
| 5 | Enter non float numbers for Length | Compilation Error | Compilation Error |
| 6 | Enter no variables when creating the object | Uses the default variables | Uses the default variables |
| 7 | Enter no variable for width when creating the object | Uses the default width variable | Uses the default width variable |
| 8 | Enter no variable for length when creating the object | Uses the default length variable | Uses the default length variable |
| 7 | Enter a letters into width/height | Compilation Error | Compilation Error |
| 10 | Enter symbols as width/height | Compilation Error | Compilation Error |
| 11 | Enter float numbers between 0.00 and 20.0 for width/height | Accept numbers and show result output | Accept numbers and show result output |
| 15 | Check Perimeter is correct | Perimeter of Rectangle is correct | Perimeter of Rectangle is correct |
| 16 | Check Rectangle Area is correct | Area of Rectangle is correct | Area of Rectangle is correct |
| 17 | Programme closes at end | Programme closes at end | Programme closes at end |

# **Source Code – Rectangle.java**

|  |
| --- |
| /\* McMahonMichael\_Wk2\_Q1\_MCT619  \* Q1. Create a class Rectangle. The class has attributes length and width, each of which defaults to 1.  \* It has methods that calculate the perimeter and the area of the rectangle. It has set and get methods  \* for both length and width. The set methods should verify that length and width are each floating-point  \* numbers larger than 0.0 and less than 20.0. Write a program to test class Rectangle.  \*  \* This is the Simple Rectangle Class file  \*/  **public** **class** Rectangle {    **private** **float** length; // private floats to hold the length of rectangle  **private** **float** width; // private floats to hold the width of rectangle  **public** **float** perimeter; // private floats to hold the perimeter of rectangle  **public** **float** area; // private floats to hold the area of rectangle    //Rectangle default constructor  **public** Rectangle()  {  **this**.length = 1.0F;  **this**.width = 1.0F;  }  //Set method for the height - need to check float  **public** **void** setLength(**float** leng)  {  **if**(leng < 0.0 || leng >= 20.0) //verify height is larger than 0.0 and less than 20.0  {  **throw** **new** IllegalArgumentException("The width must be a float between 0.0 and 20.0\nYou sent: " +leng);  }  **else**  **this**.length = leng; //set the height  }  //Set method for the width - Need to check float  **public** **void** setWidth(**float** wide)  {  **if**(wide < 0.0 || wide >= 20.0) //verify width is larger than 0.0 and less than 20.0  {  **throw** **new** IllegalArgumentException("The width must be a float between 0.0 and 20.0\nYou sent: " +wide);  }  **else**  **this**.width = wide; //set the width  }    //Get method for Length  **public** **float** getLength()  {  **return** **this**.length;  }    //Get method for Width  **public** **float** getWidth()  {  **return** **this**.width;  }    //Get method for Perimeter  **public** **float** getPerimeter()  {  **return** **this**.perimeter = (2 \* length) + (2 \* width);  }    //Get method for Area  **public** **float** getArea()  {  **return** **this**.area = width \* length;  }    //string to string  **public** String toString()  {  **return** "\nLength: " + **this**.length  +"\nWidth: " + **this**.width  +"\nPerimeter: " + **this**.perimeter  +"\nArea: " + **this**.area;  }  } |

# **Source Code – TestRectangle.java**

|  |
| --- |
| /\* McMahonMichael\_Wk2\_Q1\_MCT619  \* Q1. Create a class Rectangle. The class has attributes length and width, each of which defaults to 1.  \* It has methods that calculate the perimeter and the area of the rectangle. It has set and get methods  \* for both length and width. The set methods should verify that length and width are each floating-point  \* numbers larger than 0.0 and less than 20.0. Write a program to test class Rectangle.  \*  \* This is the Test Driver for the Simple Rectangle Class  \*/  **public** **class** TestRectangle {  **public** **static** **void** main(String[] args) {    //First test the Default Triangle Object without adding new parameters  Rectangle myRectangle1 = **new** Rectangle();    System.***out***.println("\nThe Default Rectangle Object has the following properties:"  + "\n"+"Length " +myRectangle1.getLength()  +"\nWidth:" +myRectangle1.getWidth()  +"\nPerimeter:" +myRectangle1.getPerimeter()  +"\nArea:" +myRectangle1.getArea());    //Second test the Triangle Object with new height and width parameters  Rectangle myRectangle2 = **new** Rectangle();  myRectangle2.setLength(5.0F);  myRectangle2.setWidth(15.0F);    System.***out***.println("\nThe Second Rectangle Object has the following properties:"  + "\n"+"Length " +myRectangle2.getLength()  +"\nWidth:" +myRectangle2.getWidth()  +"\nPerimeter:" +myRectangle2.getPerimeter()  +"\nArea:" +myRectangle2.getArea());      //UNCOMMENT BELOW TO TEST LENGTH GREATER THAN 20.0 ERROR  /\* Rectangle myRectangle3 = new Rectangle();  myRectangle3.setLength(21.1F);    System.out.println("\nThe Third Rectangle Object has the following properties:"  + "\n"+"Length " +myRectangle3.getLength()  +"\nWidth:" +myRectangle3.getWidth()  +"\nPerimeter:" +myRectangle3.getPerimeter()  +"\nArea:" +myRectangle3.getArea());  \*/    //UNCOMMENT BELOW TO TEST WIDTH GREATER THAN 20.0 ERROR  /\* Rectangle myRectangle4 = new Rectangle();  myRectangle3.setWidth(22.0F);  System.out.println("\nThe Fourth Rectangle Object has the following properties:"  + "\n"+"Length " +myRectangle4.getLength()  +"\nWidth:" +myRectangle4.getWidth()  +"\nPerimeter:" +myRectangle4.getPerimeter()  +"\nArea:" +myRectangle4.getArea());  \*/  //UNCOMMENT BELOW TO TEST LENGTH LESS THAN 0.0 ERROR  /\* Rectangle myRectangle5 = new Rectangle();  myRectangle5.setWidth(-1.0F);    System.out.println("\nThe Fifth Rectangle Object has the following properties:"  + "\n"+"Length " +myRectangle5.getLength()  +"\nWidth:" +myRectangle5.getWidth()  +"\nPerimeter:" +myRectangle5.getPerimeter()  +"\nArea:" +myRectangle5.getArea());  \*/  }    } |

Comments: Just one note above regarding storing extra attributes for the class. Otherwise, all of the requirements have been met and the documentation is sufficient.

# **Problem 2:**

Create a more sophisticated Rectangle class than the one you created above. This class stores only the Cartesian coordinates of the four corners of the rectangle. The constructor calls a set method that accepts four sets of coordinates and verifies that each of these is in the first quadrant with no single x- or y-coordinate larger than 20.0. The set method also verifies that the supplied coordinates specify a rectangle.

# **Approach**

To verify that the supplied coordinates specify a rectangle we need to first show that the shape is a parallelogram. We will need to verify that both pairs of opposite sides are parallel. To do this we need to calculate the slope of each side. If we can show that the slopes of the opposite sides are the same, then the opposite sides are parallel and the shape is indeed a parallelogram.

Then we prove that the parallelogram is a rectangle by showing that that one of the angles is a right angle. We have all of the slopes so we show that AB is perpendicular to BC because the slopes are negative reciprocals of each other. And if these are perpendicular it would follow that the other angles must also be right angles and be a rectangle.

My preference would have been to encapsulate the slope, parallelogram and perpendicular line work into separate methods but placed them in a single set method as per the wording of the question.

**Issues**

I had ran in into issue with lines which were 90 or 180 degrees because the slope was infinite which caused the calculation: slopeAB = diff\_Y\_AB / diff\_X\_AB to throw a divide by Zero exception - I could not resolve this issue fully

I also attempted to output the actual angle for line AB and line BC using the formula: double angle = (Math.toDegrees(Math.atan((this.slopeAB - this.slopeBC) / (1 - (this.slopeAB \* this.slopeBC))))); but has issues getting this to work correctly.

# **Design**

**I did not have enough time to create a flow chart**

TestRectangle2.java creates the object myRectangleX and sets the A, B, C, D cooridinates

We verify that each of these is in the first quadrant by ensuring both X and Y variables are above 0.0

We verify no single x- or y-coordinate larger than 20.0.

IF not throw illegal exception argument

Set the eight coordinates

Calculate the slope of all four lines AB, BC, CD, DA.

Verify if Slope of line AB is equal to Slope of line CD

Verify if Slope of line BC is equal to Slope of line DA

Verify if the shape is a parallelogram by checking that both sets of opposite slopes are equal

Verify that AB is perpendicular to BC and therefore a Right Angle

Get returns for all variables

EXIT

# **Testing**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Test** | **Expected Result** | **Actual Result** |
| 1 | Enter positive double numbers larger than 20.0 for Length | IllegalArgumentException | IllegalArgumentException |
| 2 | Enter negative double numbers less than 0.0 for Length | IllegalArgumentException | IllegalArgumentException |
| 3 | Enter positive double numbers larger than 20.0 for Width | IllegalArgumentException | IllegalArgumentException |
| 4 | Enter negative double numbers less than 0.0 for Width | IllegalArgumentException | IllegalArgumentException |
| 5 | Enter non double numbers for Length | Compilation Error | Compilation Error |
| 6 | Enter no variables when creating the object | Uses the default variables | Uses the default variables |
| 7 | Enter no variable for width when creating the object | Uses the default width variable | Uses the default width variable |
| 8 | Enter no variable for length when creating the object | Uses the default length variable | Uses the default length variable |
| 7 | Enter a letters into width/height | Compilation Error | Compilation Error |
| 10 | Enter symbols as width/height | Compilation Error | Compilation Error |
| 11 | Enter double numbers between 0.00 and 20.0 for width/height | Accept numbers and show result output | Accept numbers and show result output |
| 15 | Check shape is a rectangle | Rectangle is correct | Rectangle is correct |
| 16 | Check shape is not a rectangle | Not a Rectangle is correct | Not a Rectangle is correct |
| 17 | Programme closes at end | Programme closes at end | Programme closes at end |

# **Source Code – Rectangle2.java**

|  |
| --- |
| /\* McMahonMichael\_Wk2\_Q2\_MCT619  \* Create a more sophisticated Rectangle class than the one you created above. This class stores only the  \* Cartesian coordinates of the four corners of the rectangle. The constructor calls a set method that  \* accepts four sets of coordinates and verifies that each of these is in the first quadrant with no single  \* x- or y-coordinate larger than 20.0. The set method also verifies that the supplied coordinates specify  \* a rectangle.\*/  **public** **class** Rectangle2 {    **double** x1, y1, x2, y2, x3, y3, x4, y4; // The coordinates A(x1,y1), B(x2,y2), C(x3,y3), D(x4,y4)  **double** slopeAB, slopeBC, slopeCD, slopeDA; // Slope of lines between points of the rectangle  **boolean** isParallel\_AB\_CD, isParallel\_BC\_DA; //are they parallel lines  **boolean** isParallelogram; // is the shape a Parallelogram  **boolean** isRectangle; // is the shape a Rectangle  **double** angle; // the angle at the intersection of line AB and BC    //Rectangle default constructor  **public** Rectangle2()  {  setPoints(1.0, 4.0, 4.0, 7.0, 6.0, 5.0, 3.0, 2.0);  }    //Test the points and set point coordinates for the four points  **public** **void** setPoints(**double** x1, **double** y1, **double** x2, **double** y2, **double** x3, **double** y3, **double** x4, **double** y4)  {  **if**(x1 < 00.0 || x1 > 20.0  || y1 < 00.0 || y1 > 20.0  || x2 < 00.0 || x2 > 20.0  || y2 < 00.0 || y2 > 20.0  || x3 < 00.0 || x3 > 20.0  || y3 < 00.0 || y3 > 20.0  || x4 < 00.0 || x4 > 20.0  || y4 < 00.0 || y4 > 20.0)    {  **throw** **new** IllegalArgumentException("The point values must be between 0.0 and 20.0");  }  **else**  {  **this**.x1 = x1; //set x1 of point A  **this**.y1 = y1; //set y1 of point A  **this**.x2 = x2; //set x2 of point B  **this**.y2 = y2; //set y2 of point B  **this**.x3 = x3; //set x3 of point C  **this**.y3 = y3; //set y3 of point C  **this**.x4 = x4; //set x4 of point D  **this**.y4 = y4; //set y4 of point D  }    /\*To verify that the supplied coordinates specify a rectangle we need to first show that the shape is a parallelogram.  \* We will need to verify that both pairs of opposite sides are parallel. To do this we need to calculate the slope of  \* each side. If we can show that the slopes of the opposite sides are the same, then the opposite sides are parallel  \* and the shape is indeed a parallelogram\*/    //Calculate the slope of Line A-B  **double** diff\_Y\_AB = **this**.y1 - **this**.y2;  **double** diff\_X\_AB = **this**.x1 - **this**.x2;  **if** (diff\_Y\_AB == 0){  slopeAB = 0; //if slope is infinity so set to zero  }  **else** **if** (diff\_X\_AB == 0){  slopeAB = 0; //if slope is infinity so set to zero  }  **else**{  slopeAB = diff\_Y\_AB / diff\_X\_AB;  }      //Calculate the slope of Line B-C  **double** diff\_Y\_BC = **this**.y2 - **this**.y3;  **double** diff\_X\_BC = **this**.x2 - **this**.x3;  **if** (diff\_Y\_BC == 0){  slopeBC = 0;  }  **else** **if** (diff\_X\_BC == 0){  slopeBC = 0;  }  **else**{  slopeBC = diff\_Y\_BC / diff\_X\_BC;  }  //Calculate the slope of Line C-D  **double** diff\_Y\_CD = **this**.y3 - **this**.y4;  **double** diff\_X\_CD = **this**.x3 - **this**.x4;  **if** (diff\_Y\_CD == 0)  {  slopeCD = 0;  }  **else** **if** (diff\_X\_CD == 0)  {  slopeCD = 0;  }  **else**  {  slopeCD = diff\_Y\_CD / diff\_X\_CD;  }  //Calculate the slope of Line D-A  **double** diff\_Y\_DA = **this**.y4 - **this**.y1;  **double** diff\_X\_DA = **this**.x4 - **this**.x1;  **if** (diff\_Y\_DA == 0){  slopeDA = 0;  }  **else** **if** (diff\_X\_DA == 0){  slopeDA = 0;  }  **else**{  slopeDA = diff\_Y\_DA / diff\_X\_DA;  }    //Verify if Slope of line AB is equal to Slope of line CD  **if** (**this**.slopeAB == **this**.slopeCD){  isParallel\_AB\_CD = **true**;  }  **else**{  isParallel\_AB\_CD = **false**;  }    //Verify if Slope of line BC is equal to Slope of line DA  **if** (**this**.slopeBC == **this**.slopeDA){  isParallel\_BC\_DA = **true**;  }  **else**{  isParallel\_BC\_DA = **false**;  }    //Verify if the shape is a parallelogram  **if** ((**this**.isParallel\_AB\_CD == **true**) && (**this**.isParallel\_BC\_DA == **true**)){  isParallelogram = **true**;  }  **else**{  isParallelogram = **false**;  }    /\*Now prove that the parallelogram is a rectangle by showing that that one of the angles  is a right angle We have all of the slopes so we verify that AB is perpendicular to BC because  the slopes are negative reciprocals of each other. And if these are perpendicular it would  follow that the other angles must also be right angles and be a rectangle.\*/  **if** (**this**.slopeAB + **this**.slopeBC == 0){  isRectangle = **true**;  }  **else**{  isRectangle = **false**;  }  }          //Get method for Point A x1 coordinate  **public** **double** getPointAx1()  {  **return** **this**.x1;  }    //Get method for Point A y1 coordinate  **public** **double** getPointAy1()  {  **return** **this**.y1;  }    //Get method for Point B x2 coordinate  **public** **double** getPointBx2()  {  **return** **this**.x2;  }    //Get method for Point B y2 coordinate  **public** **double** getPointBy2()  {  **return** **this**.y2;  }    //Get method for Point C x3 coordinate  **public** **double** getPointBx3()  {  **return** **this**.x3;  }    //Get method for Point C y3 coordinate  **public** **double** getPointBy3()  {  **return** **this**.y3;  }    //Get method for Point D x4 coordinate  **public** **double** getPointBx4()  {  **return** **this**.x4;  }    //Get method for Point D y4 coordinate  **public** **double** getPointBy4()  {  **return** **this**.y4;  }    //Get method for slope of AB line  **public** **double** getSlopeOfAB()  {  **return** **this**.slopeAB;  }    //Get method for slope of BC line  **public** **double** getSlopeOfBC()  {  **return** **this**.slopeBC;  }    //Get method for slope of CD line  **public** **double** getSlopeOfCD()  {  **return** **this**.slopeCD;  }    //Get method for slope of DA line  **public** **double** getSlopeOfDA()  {  **return** **this**.slopeDA;  }    //Get method for slope of DA line  **public** **boolean** getParallel\_AB\_CD()  {  **return** **this**.isParallel\_AB\_CD;  }    //Get method for slope of DA line  **public** **boolean** getParallel\_BC\_DA()  {  **return** **this**.isParallel\_BC\_DA;  }    //Get method for slope of DA line  **public** **boolean** getParallelogram()  {  **return** **this**.isParallelogram;  }    //Get method for slope of DA line  **public** **boolean** getRectangle()  {  **return** **this**.isRectangle;  }    **public** String toString()  {  **return** "\nThe Default Rectangle Object has the following properties:"  + "\n"+"Point A (" +**this**.getPointAx1()+", "+**this**.getPointAy1()+")"  +"\nPoint B (" +**this**.getPointBx2()+", "+**this**.getPointBy2()+")"  +"\nPoint C (" +**this**.getPointBx3()+", "+**this**.getPointBy3()+")"  +"\nPoint D (" +**this**.getPointBx4()+", "+**this**.getPointBy4()+")"  +"\nSlope of A B Line: " +**this**.getSlopeOfAB()  +"\nSlope of B C Line: " +**this**.getSlopeOfBC()  +"\nSlope of C D Line: " +**this**.getSlopeOfCD()  +"\nSlope of D A Line: " +**this**.getSlopeOfDA()  +"\nSlope of AB and CD are equal: " +**this**.getParallel\_AB\_CD()  +"\nSlope of BC and DA are equal: " +**this**.getParallel\_BC\_DA()  +"\nThe shape is a parallelogram: " +**this**.getParallelogram()  +"\nThe shape is a Rectangle: " +**this**.getRectangle()  +"\nAngle: " +**this**.getAngle();  }    //COULD NOT REALLY GET THIS TO WORK - Get method for calculating if the intersection of line AB and BC make a right angle  **public** **double** getAngle()  {  **double** angle = (Math.*toDegrees*(Math.*atan*((**this**.slopeAB - **this**.slopeBC) / (1 - (**this**.slopeAB \* **this**.slopeBC)))));  **return** angle;  }  } |

# **Source Code – TestRectangle2.java**

|  |
| --- |
| /\* McMahonMichael\_Wk2\_Q2\_MCT619  \* Create a more sophisticated Rectangle class than the one you created above. This class stores only the  \* Cartesian coordinates of the four corners of the rectangle. The constructor calls a set method that  \* accepts four sets of coordinates and verifies that each of these is in the first quadrant with no single  \* x- or y-coordinate larger than 20.0. The set method also verifies that the supplied coordinates specify  \* a rectangle.\*/  **public** **class** TestRectangle2 {  **public** **static** **void** main(String[] args) {    //First test the Default Triangle Object without adding new parameters  Rectangle2 myRectangle1 = **new** Rectangle2();  //First test of a Triangle Object using defined parameters  System.***out***.println("\nThe Default Rectangle Object has the following properties:"  + "\n"+"Point A (" +myRectangle1.getPointAx1()+", "+myRectangle1.getPointAy1()+")"  +"\nPoint B (" +myRectangle1.getPointBx2()+", "+myRectangle1.getPointBy2()+")"  +"\nPoint C (" +myRectangle1.getPointBx3()+", "+myRectangle1.getPointBy3()+")"  +"\nPoint D (" +myRectangle1.getPointBx4()+", "+myRectangle1.getPointBy4()+")"  +"\nSlope of A B Line: " +myRectangle1.getSlopeOfAB()  +"\nSlope of B C Line: " +myRectangle1.getSlopeOfBC()  +"\nSlope of C D Line: " +myRectangle1.getSlopeOfCD()  +"\nSlope of D A Line: " +myRectangle1.getSlopeOfDA()  +"\nSlope of AB and CD are equal: " +myRectangle1.getParallel\_AB\_CD()  +"\nSlope of BC and DA are equal: " +myRectangle1.getParallel\_BC\_DA()  +"\nThe shape is a parallelogram: " +myRectangle1.getParallelogram()  +"\nThe shape is a Rectangle: " +myRectangle1.getRectangle()  +"\nAngle: " +myRectangle1.getAngle()  );    //This is the Rectangle I had issues with because slope of all lines is infinity (90 and 180 degrees)  //Second test of a Triangle Object adding new parameters  /\*  Rectangle2 myRectangle2 = new Rectangle2();    myRectangle2.setPoints(1.0, 2.0, 1.0, 4.0, 5.0, 4.0, 5.0, 2.0);  System.out.println("\nThe Default Rectangle Object has the following properties:"  + "\n"+"Point A (" +myRectangle2.getPointAx1()+", "+myRectangle2.getPointAy1()+")"  +"\nPoint B (" +myRectangle2.getPointBx2()+", "+myRectangle2.getPointBy2()+")"  +"\nPoint C (" +myRectangle2.getPointBx3()+", "+myRectangle2.getPointBy3()+")"  +"\nPoint D (" +myRectangle2.getPointBx4()+", "+myRectangle2.getPointBy4()+")"  +"\nSlope of A B Line: " +myRectangle2.getSlopeOfAB()  +"\nSlope of B C Line: " +myRectangle2.getSlopeOfBC()  +"\nSlope of C D Line: " +myRectangle2.getSlopeOfCD()  +"\nSlope of D A Line: " +myRectangle2.getSlopeOfDA()  +"\nSlope of AB and CD are equal: " +myRectangle2.getParallel\_AB\_CD()  +"\nSlope of BC and DA are equal: " +myRectangle2.getParallel\_BC\_DA()  +"\nThe shape is a parallelogram: " +myRectangle2.getParallelogram()  +"\nThe shape is a Rectangle: " +myRectangle2.getRectangle()  +"\nAngle: " +myRectangle2.getAngle()  );  \*/  }  } |

# **Problem 3:**

Provide methods to calculate the length, width, perimeter and area (the length is the larger of the two dimensions). Include a predicate method (methods like this, that test a quality and return true/false, are called predicate methods) isSquare which determines whether the rectangle is a square. Write a program to test class Rectangle.

# **Approach**

Once I had determined how to get the distance between two coordinated using the formula: lengthAB = Math.sqrt((x2 - x1)\*(x2 - x1) + (y2 - y1)\*(y2 - y1)); I then worked out which was the length and width variable and built upon the work I did in WK 1 for area and perimeter. For the isSquare I checked if the sides were the same length, if the shape was a Parallelogram and if one corner was a right angle

# **Design**

**I did not have enough time to create a flow chart**

Design is exactly as Rectangle2.java with the addition of the following

Calculate the distance between AB

Calculate the distance between BC

Calculate the distance between CD

Calculate the distance between DA

Verify which side is the Length

Verify which side if the width

Calculate the Area

Calculate the Perimeter

To check is object is a Square

* Check if both width and height are the same
* Check that object is a Parallelogram
* Check that the object has a right angle

EXIT

# **Testing**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Test** | **Expected Result** | **Actual Result** |
| 1 | Enter positive double numbers larger than 20.0 for Length | IllegalArgumentException | IllegalArgumentException |
| 2 | Enter negative double numbers less than 0.0 for Length | IllegalArgumentException | IllegalArgumentException |
| 3 | Enter positive double numbers larger than 20.0 for Width | IllegalArgumentException | IllegalArgumentException |
| 4 | Enter negative double numbers less than 0.0 for Width | IllegalArgumentException | IllegalArgumentException |
| 5 | Enter non double numbers for Length | Compilation Error | Compilation Error |
| 6 | Enter no variables when creating the object | Uses the default variables | Uses the default variables |
| 7 | Enter no variable for width when creating the object | Uses the default width variable | Uses the default width variable |
| 8 | Enter no variable for length when creating the object | Uses the default length variable | Uses the default length variable |
| 7 | Enter a letters into width/height | Compilation Error | Compilation Error |
| 10 | Enter symbols as width/height | Compilation Error | Compilation Error |
| 11 | Enter double numbers between 0.00 and 20.0 for width/height | Accept numbers and show result output | Accept numbers and show result output |
| 15 | Check shape is a rectangle | Rectangle is correct | Rectangle is correct |
| 16 | Check shape is not a rectangle | Not a Rectangle is correct | Not a Rectangle is correct |
|  | Check that all lengths are correct | All lengths are correct | All lengths are correct |
|  | Check Area is correct | Area is correct | Area is correct |
|  | Check Perimeter is correct | Perimeter is correct | Perimeter is correct |
|  | Check shape is a square | Shape is a square | Shape is a square |
|  | Check shape is not a square | Shape is not a square | Shape is not a square |
| 17 | Programme closes at end | Programme closes at end | Programme closes at end |

# **Source Code – Rectangle3.java**

|  |
| --- |
| /\* McMahonMichael\_Wk2\_Q3\_MCT619  \* Provide methods to calculate the length, width, perimeter and area (the length is the larger of the two dimensions).  \* Include a predicate method (methods like this, that test a quality and return true/false, are called predicate methods)  \* isSquare which determines whether the rectangle is a square. Write a program to test class Rectangle.\*/  **public** **class** Rectangle3 {    **double** x1, y1, x2, y2, x3, y3, x4, y4; // The coordinates A(x1,y1), B(x2,y2), C(x3,y3), D(x4,y4)  **double** slopeAB, slopeBC, slopeCD, slopeDA; // Slope of lines between points of the rectangle  **boolean** isParallel\_AB\_CD, isParallel\_BC\_DA; //are they parallel lines  **boolean** isParallelogram; // is the shape a Parallelogram  **boolean** isRectangle; // is the shape a Rectangle  **double** angle; // the angle at the intersection of line AB and BC  **double** lengthAB, lengthBC, lengthCD, lengthDA; //the Length of each line  **double** length, width;  **double** area, perimeter; //the area and perimeter  **boolean** isSquare;    //Rectangle default constructor  **public** Rectangle3()  {  setPoints(1.0, 4.0, 4.0, 7.0, 6.0, 5.0, 3.0, 2.0);  }    //Test the points and set point coordinates for the four points  **public** **void** setPoints(**double** x1, **double** y1, **double** x2, **double** y2, **double** x3, **double** y3, **double** x4, **double** y4)  {  **if**(x1 < 00.0 || x1 > 20.0  || y1 < 00.0 || y1 > 20.0  || x2 < 00.0 || x2 > 20.0  || y2 < 00.0 || y2 > 20.0  || x3 < 00.0 || x3 > 20.0  || y3 < 00.0 || y3 > 20.0  || x4 < 00.0 || x4 > 20.0  || y4 < 00.0 || y4 > 20.0)    {  **throw** **new** IllegalArgumentException("The point values must be between 0.0 and 20.0");  }  **else**  {  **this**.x1 = x1; //set x1 of point A  **this**.y1 = y1; //set y1 of point A  **this**.x2 = x2; //set x2 of point B  **this**.y2 = y2; //set y2 of point B  **this**.x3 = x3; //set x3 of point C  **this**.y3 = y3; //set y3 of point C  **this**.x4 = x4; //set x4 of point D  **this**.y4 = y4; //set y4 of point D  }    /\*To verify that the supplied coordinates specify a rectangle we need to first show that the shape is a parallelogram.  \* We will need to verify that both pairs of opposite sides are parallel. To do this we need to calculate the slope of  \* each side. If we can show that the slopes of the opposite sides are the same, then the opposite sides are parallel  \* and the shape is indeed a parallelogram\*/    //Calculate the slope of Line A-B  **double** diff\_Y\_AB = **this**.y1 - **this**.y2;  **double** diff\_X\_AB = **this**.x1 - **this**.x2;  **if** (diff\_Y\_AB == 0){  slopeAB = 0; //if slope is infinity so set to zero  }  **else** **if** (diff\_X\_AB == 0){  slopeAB = 0; //if slope is infinity so set to zero  }  **else**{  slopeAB = diff\_Y\_AB / diff\_X\_AB;  }      //Calculate the slope of Line B-C  **double** diff\_Y\_BC = **this**.y2 - **this**.y3;  **double** diff\_X\_BC = **this**.x2 - **this**.x3;  **if** (diff\_Y\_BC == 0){  slopeBC = 0;  }  **else** **if** (diff\_X\_BC == 0){  slopeBC = 0;  }  **else**{  slopeBC = diff\_Y\_BC / diff\_X\_BC;  }  //Calculate the slope of Line C-D  **double** diff\_Y\_CD = **this**.y3 - **this**.y4;  **double** diff\_X\_CD = **this**.x3 - **this**.x4;  **if** (diff\_Y\_CD == 0)  {  slopeCD = 0;  }  **else** **if** (diff\_X\_CD == 0)  {  slopeCD = 0;  }  **else**  {  slopeCD = diff\_Y\_CD / diff\_X\_CD;  }  //Calculate the slope of Line D-A  **double** diff\_Y\_DA = **this**.y4 - **this**.y1;  **double** diff\_X\_DA = **this**.x4 - **this**.x1;  **if** (diff\_Y\_DA == 0){  slopeDA = 0;  }  **else** **if** (diff\_X\_DA == 0){  slopeDA = 0;  }  **else**{  slopeDA = diff\_Y\_DA / diff\_X\_DA;  }    //Verify if Slope of line AB is equal to Slope of line CD  **if** (**this**.slopeAB == **this**.slopeCD){  isParallel\_AB\_CD = **true**;  }  **else**{  isParallel\_AB\_CD = **false**;  }    //Verify if Slope of line BC is equal to Slope of line DA  **if** (**this**.slopeBC == **this**.slopeDA){  isParallel\_BC\_DA = **true**;  }  **else**{  isParallel\_BC\_DA = **false**;  }    //Verify if the shape is a parallelogram  **if** ((**this**.isParallel\_AB\_CD == **true**) && (**this**.isParallel\_BC\_DA == **true**)){  isParallelogram = **true**;  }  **else**{  isParallelogram = **false**;  }    /\*Now prove that the parallelogram is a rectangle by showing that that one of the angles  is a right angle We have all of the slopes so we verify that AB is perpendicular to BC because  the slopes are negative reciprocals of each other. And if these are perpendicular it would  follow that the other angles must also be right angles and be a rectangle.\*/  **if** (**this**.slopeAB + **this**.slopeBC == 0){  isRectangle = **true**;  }  **else**{  isRectangle = **false**;  }  }          //Get method for Point A x1 coordinate  **public** **double** getPointAx1()  {  **return** **this**.x1;  }    //Get method for Point A y1 coordinate  **public** **double** getPointAy1()  {  **return** **this**.y1;  }    //Get method for Point B x2 coordinate  **public** **double** getPointBx2()  {  **return** **this**.x2;  }    //Get method for Point B y2 coordinate  **public** **double** getPointBy2()  {  **return** **this**.y2;  }    //Get method for Point C x3 coordinate  **public** **double** getPointBx3()  {  **return** **this**.x3;  }    //Get method for Point C y3 coordinate  **public** **double** getPointBy3()  {  **return** **this**.y3;  }    //Get method for Point D x4 coordinate  **public** **double** getPointBx4()  {  **return** **this**.x4;  }    //Get method for Point D y4 coordinate  **public** **double** getPointBy4()  {  **return** **this**.y4;  }    //Get method for slope of AB line  **public** **double** getSlopeOfAB()  {  **return** **this**.slopeAB;  }    //Get method for slope of BC line  **public** **double** getSlopeOfBC()  {  **return** **this**.slopeBC;  }    //Get method for slope of CD line  **public** **double** getSlopeOfCD()  {  **return** **this**.slopeCD;  }    //Get method for slope of DA line  **public** **double** getSlopeOfDA()  {  **return** **this**.slopeDA;  }    //Get method for slope of DA line  **public** **boolean** getParallel\_AB\_CD()  {  **return** **this**.isParallel\_AB\_CD;  }    //Get method for slope of DA line  **public** **boolean** getParallel\_BC\_DA()  {  **return** **this**.isParallel\_BC\_DA;  }    //Get method for slope of DA line  **public** **boolean** getParallelogram()  {  **return** **this**.isParallelogram;  }    //Get method for slope of DA line  **public** **boolean** getRectangle()  {  **return** **this**.isRectangle;  }    //COULD NOT REALLY GET THIS TO WORK - Get method for calculating if the intersection of line AB and BC make a right angle  **public** **double** getAngle()  {  **double** angle = (Math.*toDegrees*(Math.*atan*((**this**.slopeAB - **this**.slopeBC) / (1 - (**this**.slopeAB \* **this**.slopeBC)))));  **return** angle;  }    //Get method for Length of AB Line  **public** **double** getLengthOfAB()  {  **return** lengthAB = Math.*sqrt*((x2 - x1)\*(x2 - x1) + (y2 - y1)\*(y2 - y1));  }    //Get method for Length of BC Line  **public** **double** getLengthOfBC()  {  **return** lengthBC = Math.*sqrt*((x3 - x2)\*(x3 - x2) + (y3 - y2)\*(y3 - y2));  }    //Get method for Length of CD Line  **public** **double** getLengthOfCD()  {  **return** lengthCD = Math.*sqrt*((x4 - x3)\*(x4 - x3) + (y4 - y3)\*(y4 - y3));  }    //Get method for Length of DA Line  **public** **double** getLengthOfDA()  {  **return** lengthDA = Math.*sqrt*((x4 - x1)\*(x4 - x1) + (y4 - y1)\*(y4 - y1));  }    //Get method to determine which side is Length  **public** **double** getLength()  {  **if** (**this**.lengthAB > **this**.lengthBC){  **return** **this**.length = **this**.lengthAB;  }  **else** {  **return** **this**.length = **this**.lengthBC;  }  }  //Get method to determine which side is Width  **public** **double** getWidth()  {  **if** (**this**.lengthAB > **this**.lengthBC){  **return** **this**.width = **this**.lengthBC;  }  **else** {  **return** **this**.width = **this**.lengthAB;  }  }    //Get method for Perimeter  **public** **double** getPerimeter()  {  **return** **this**.perimeter = (2 \* **this**.length) + (2 \* **this**.width);  }    //Get method for Area  **public** **double** getArea()  {  **return** **this**.area = **this**.width \* **this**.length;  }    //verify if the shape is a square by testing if all sides are equal length and has a right angle.  **public** **boolean** isSquare()  {  **if** ((**this**.width == **this**.length) && (**this**.slopeAB + **this**.slopeBC == 0)) {  isSquare = **true**;  }  **else**{  isSquare = **false**;  }  **return** isSquare;  }    **public** String toString()  {  **return** "\nThe Default Rectangle Object has the following properties:"  + "\n"+"Point A (" +**this**.getPointAx1()+", "+**this**.getPointAy1()+")"  +"\nPoint B (" +**this**.getPointBx2()+", "+**this**.getPointBy2()+")"  +"\nPoint C (" +**this**.getPointBx3()+", "+**this**.getPointBy3()+")"  +"\nPoint D (" +**this**.getPointBx4()+", "+**this**.getPointBy4()+")"  +"\nSlope of A B Line: " +**this**.getSlopeOfAB()  +"\nSlope of B C Line: " +**this**.getSlopeOfBC()  +"\nSlope of C D Line: " +**this**.getSlopeOfCD()  +"\nSlope of D A Line: " +**this**.getSlopeOfDA()  +"\nSlope of AB and CD are equal: " +**this**.getParallel\_AB\_CD()  +"\nSlope of BC and DA are equal: " +**this**.getParallel\_BC\_DA()  +"\nThe shape is a parallelogram: " +**this**.getParallelogram()  +"\nThe shape is a Rectangle: " +**this**.getRectangle()  +"\nAngle: " +**this**.getAngle()  +"\nLength of A B Line: " +**this**.getLengthOfAB()  +"\nLength of B C Line: " +**this**.getLengthOfBC()  +"\nLength of C D Line: " +**this**.getLengthOfCD()  +"\nLength of D A Line: " +**this**.getLengthOfDA()  +"\nLength: " +**this**.getLength()  +"\nWidth: " +**this**.getWidth()  +"\nPerimeter of Rectangle: " +**this**.getPerimeter()  +"\nArea of Rectangle: " +**this**.getArea()  +"\nIs the shape a square: " +**this**.isSquare();  }  } |

# **Source Code – TestRectangle3.java**

|  |
| --- |
| /\* McMahonMichael\_Wk2\_Q3\_MCT619  \* Provide methods to calculate the length, width, perimeter and area (the length is the larger of the two dimensions).  \* Include a predicate method (methods like this, that test a quality and return true/false, are called predicate methods)  \* isSquare which determines whether the rectangle is a square. Write a program to test class Rectangle.\*/  **public** **class** TestRectangle3 {    **public** **static** **void** main(String[] args) {    //First test the Default Triangle Object without adding new parameters  Rectangle3 myRectangle1 = **new** Rectangle3();    //First test of a Triangle Object using defined parameters  System.***out***.println("\nThe Default Rectangle Object has the following properties:"  + "\n"+"Point A (" +myRectangle1.getPointAx1()+", "+myRectangle1.getPointAy1()+")"  +"\nPoint B (" +myRectangle1.getPointBx2()+", "+myRectangle1.getPointBy2()+")"  +"\nPoint C (" +myRectangle1.getPointBx3()+", "+myRectangle1.getPointBy3()+")"  +"\nPoint D (" +myRectangle1.getPointBx4()+", "+myRectangle1.getPointBy4()+")"  +"\nSlope of A B Line: " +myRectangle1.getSlopeOfAB()  +"\nSlope of B C Line: " +myRectangle1.getSlopeOfBC()  +"\nSlope of C D Line: " +myRectangle1.getSlopeOfCD()  +"\nSlope of D A Line: " +myRectangle1.getSlopeOfDA()  +"\nSlope of AB and CD are equal: " +myRectangle1.getParallel\_AB\_CD()  +"\nSlope of BC and DA are equal: " +myRectangle1.getParallel\_BC\_DA()  +"\nThe shape is a parallelogram: " +myRectangle1.getParallelogram()  +"\nThe shape is a Rectangle: " +myRectangle1.getRectangle()  +"\nAngle: " +myRectangle1.getAngle()  +"\nLength of A B Line: " +myRectangle1.getLengthOfAB()  +"\nLength of B C Line: " +myRectangle1.getLengthOfBC()  +"\nLength of C D Line: " +myRectangle1.getLengthOfCD()  +"\nLength of D A Line: " +myRectangle1.getLengthOfDA()  +"\nLength: " +myRectangle1.getLength()  +"\nWidth: " +myRectangle1.getWidth()  +"\nPerimeter of Rectangle: " +myRectangle1.getPerimeter()  +"\nArea of Rectangle: " +myRectangle1.getArea()  +"\nIs the shape a square: " +myRectangle1.isSquare()  );  }  } |

# **References:**

[1] *Calculating the angle between two lines without having to calculate the slope? (Java)*. (2016). *Stackoverflow.com*. Retrieved 4 September 2016, from <http://stackoverflow.com/questions/3365171/calculating-the-angle-between-two-lines-without-having-to-calculate-the-slope>

[2] *Calculating the distance between two points*. (2016). *Stackoverflow.com*. Retrieved 4 September 2016, from <http://stackoverflow.com/questions/929773/calculating-the-distance-between-two-points>

[3] *Calculating the distance between two points problem (Beginning Java forum at Coderanch)*. (2016). *Coderanch.com*. Retrieved 4 September 2016, from <https://coderanch.com/t/537897/java/java/Calculating-distance-points>

[4] *cartesian slope calculation error Java*. (2016). *Stackoverflow.com*. Retrieved 4 September 2016, from <http://stackoverflow.com/questions/19447047/cartesian-slope-calculation-error-java>

[5] *Checking if a list of coordinates defines a closed, exact rectangle*. (2016). *Codereview.stackexchange.com*. Retrieved 4 September 2016, from <http://codereview.stackexchange.com/questions/54488/checking-if-a-list-of-coordinates-defines-a-closed-exact-rectangle>

[6] *CmSc 150 Lab02*. (2016). *Faculty.simpson.edu*. Retrieved 4 September 2016, from <http://faculty.simpson.edu/lydia.sinapova/www/cmsc150/cmsc150-05/Labs/Lab02_Identifiers.htm>

[7] *Creating a Class Then Testing that Class*. (2016). *Stackoverflow.com*. Retrieved 4 September 2016, from <http://stackoverflow.com/questions/29980466/creating-a-class-then-testing-that-class>

[8] *Deitel-8thEd-Ex9.12-Rectangle - C++ Programming*. (2016). *Sites.google.com*. Retrieved 4 September 2016, from <https://sites.google.com/site/pdcppnotes/classes/03c-ex9-12-deitel-8thed-rectangle>

[9] Deitel, H. & Deitel, P. (2012). *Java How to Program* (9th ed.). Upper Saddle River, NJ: Prentice Hall.

[10] *Design a class named Rectangle to represent a rectangle.*. (2016). *DaniWeb*. Retrieved 4 September 2016, from <https://www.daniweb.com/programming/software-development/threads/280390/design-a-class-named-rectangle-to-represent-a-rectangle>

[11] *Java Beginner Programming Tutorial - 30 - Objects & Classes Rectangle*. (2016). *YouTube*. Retrieved 4 September 2016, from <https://www.youtube.com/watch?v=8UC2pUGoVpc>

[12] *Java Ski Slope - Part 1 (Slope Algorithm)*. (2016). *YouTube*. Retrieved 4 September 2016, from <https://www.youtube.com/watch?v=ri6Pky7QUBk>

[13] *Java: Calculating the angle between two points in degrees*. (2016). *Stackoverflow.com*. Retrieved 4 September 2016, from <http://stackoverflow.com/questions/9970281/java-calculating-the-angle-between-two-points-in-degrees>

[14] *Point (Java Platform SE 7 )*. (2016). *Docs.oracle.com*. Retrieved 4 September 2016, from <https://docs.oracle.com/javase/7/docs/api/java/awt/Point.html>

[15] *Prove it is a Rectangle*. (2016). *Softschools.com*. Retrieved 4 September 2016, from <http://www.softschools.com/math/geometry/proofs/prove_it_is_a_rectangle/>

[16] Shanahan, P. (2016). *Math.atan() | Coding Forums*. *Thecodingforums.com*. Retrieved 4 September 2016, from <http://www.thecodingforums.com/threads/math-atan.641682/>

[17] *The Basics Of Classes in Java*. (2013). *Captain DeadBones Chronicles*. Retrieved 4 September 2016, from <http://thelivingpearl.com/2013/10/17/the-basics-of-classes-in-java/>

[18] *Trying to write a program to calculate slope of a line - need help please. • /r/learnprogramming*. (2016). *reddit*. Retrieved 4 September 2016, from <https://www.reddit.com/r/learnprogramming/comments/2g2lp0/trying_to_write_a_program_to_calculate_slope_of_a/>

[19] *Write a program that prompts user to enter two points (work included)*. (2016). *Stackoverflow.com*. Retrieved 4 September 2016, from <http://stackoverflow.com/questions/21689391/write-a-program-that-prompts-user-to-enter-two-points-work-included>