**AP Computer Science - Triangle Class Assignment**

This assignment requires you to design an Triangle Class.

**Data Fields** - All data fields are private. *numTriangles* is also static.

|  |  |  |
| --- | --- | --- |
| **Data Field** | **Data Type** | **Purpose** |
| sideA | double | length of side A - This will always be the shortest side |
| sideB | double | length of side B - This will always be the middle length side |
| sideC | double | length of side C - This will always be the longest side |
| angles | double [] | array of size 3 stores the angles opposite sideA, sideB, sideC (in that order) |
| area | double | stores the area of the triangle (calculated with Heron's formula) |
| numTriangles | int | static data field stores the total number of triangles created |
| attempts | int | data field that stores the number of attempts it took to create this triangle |
| CONVERSION | double | This constant will be used to convert our angles from radians to degrees. |

**Constructors -** Your triangle will have two constructors:

**Constructor # 1**

public Triangle() - No-arg constructor - Will generate a random valid, right triangle

* Assign random integer values (1 to 100) for *sideA*, *sideB*, and *sideC* until a valid right triangle is created. Sort the sides by length from smallest to biggest, with sideA being smallest and sideC being largest. Since sides are of type double, they can hold an integer value without need for any conversion or casting.
* increase *attempts* by 1 every time you attempt new integer values for the side lengths.
* once a valid right triangle has been generated, calculate the angles opposite *sideA*, *sideB*, and *sideC* using the *calculateAngles* method. Put them in the *angles* array in the appropriate order. (angle opposite A first, angle opposite B, then angle opposite C)
* calculate *area* using the calculateArea method
* Once a valid right triangle is created, increase numTriangles by 1.
* Output the triangle using the *output* method.

**Constructor # 2**

public Triangle(double a, double b, double c) - User sends in 3 side lengths.

* set attempts to 1. (Only 1 attempt will be made)
* Sort the 3 lengths in order from smallest to biggest using the *sortSides* method.
* calculate the angles opposite *sideA*, *sideB*, and *sideC* using the *calculateAngles* method. Put them in the array in the appropriate order.
* calculate the *area* using the *calculateArea* method
* if it's a valid triangle, increase *numTriangles* by 1.
* Output the triangle using the *output* method.

**Methods**

|  |  |  |  |
| --- | --- | --- | --- |
| **Method Name** | **Parameters** | **Return Type** | **Purpose** |
| generateTriangle | none | void | This will assign random values (1-100) for sideA, sideB and sideC until the resulting combination of side lengths creates a valid right triangle. Each time is tries a new combination, attempts should be incremented by 1. Once it finds a combination of side lengths that works, numTriangles should be incremented by 1. |
| sortSides | none | void | This method put sideA, sideB and sideC in increasing order. |
| calculateAngles | none | none | *angles* data field is populated using the Law of Cosines and the radians to degrees constant, CONVERSION |
| isValid | none | boolean | If the sum of the values in the *angles* array is 180, true is returned, else false |
| isRight | none | boolean | if any of the angles in the angles array = 90, true is returned, else false. |
| calculateArea | none | double | *area* is calculated using Heron's formula |
| output | none | none | See runner expected output. |
| getAttempts | none | int | Will return the value of attempts datafield |

**Formulas**

**Heron's Formula** - Allows you to calculate the area of a triangle given its 3 side lengths (a, b, c).



Step 1 - Calculate the semiperimeter (s) of the triangle.



Step 2 - Calculate the area using s, a, b and c.

**Law of Cosines** - Given a triangle with sides A, B and C, we can find the opposite angles of each of these sides. We will assume that A, B and C are in increasing order by length.

B

A

 This is the angle opposite side A (denoted ∠A). It is calculated as the angle created between B and C.

C

$∠A= cos^{-1}(\frac{B^{2}+C^{2}-A^{2}}{2BC})$ \* (180/π) cos-1 is calculated with Math.acos() in java

$∠B= cos^{-1}(\frac{A^{2}+C^{2}-B^{2}}{2AC})$ \* (180/π) Multiplying by 180/π converts from radians to degrees

$∠C=180- ∠A- ∠B$ Remember that these are stored in the angles array! ∠A = angles[0] ∠B = angles[1] ∠C = angles[2]

**TriangleRunner** - Use this exact code for your runner:

**public** **class** TriangleRunner {

**public** **static** **void** main(String[] args) {

 Triangle t = **new** Triangle();

 Triangle t2 = **new** Triangle();

 Triangle t3 = **new** Triangle(11,99,22);

 Triangle t4 = **new** Triangle(14, 10, 6);

 Triangle t5 = new Triangle(3, 4, 5);

 **int** total = t.getAttempts() +2.getAttempts()+t3.getAttempts()+t4.getAttempts()+t5.getAttempts();

 System.*out*.println(Triangle.*numTriangles* + " triangles created after " + total + " attempts. ");

 }

}

**Expected Output**

The output will vary based on the random side lengths created for first two triangles. The 3rd and 4th triangles will use the second constructor and your output should make this one exactly.

If the no-arg constructor is used to create the triangle, you will output that is it a right triangle and how many attempts it took (randomly picking side lengths) to create. It will also output the lengths of sides A, B and C and the opposite angle for each side. It will output the sum of the angles (should show 180.0) and its area.

If the constructor with 3 double values is used it will either output "Attempt made…" or "Triangle # x created" based on whether or not it was successful. If unsuccessful, it will output "Not a valid triangle". If successful, it should output the same information as in the no-arg constructor. Be sure to not output "right triangle" unless one of its angles is 90 degrees.

Triangle # 1 created.

This is a right triangle and took 564 attempts to create.

A = 15.0 and has an opposite angle of 36.86989764584401

B = 20.0 and has an opposite angle of 53.13010235415599

C = 25.0 and has an opposite angle of 90.0

Sum of angles = 180.0

The area of this triangle is 150.0

Triangle # 2 created.

This is a right triangle and took 9481 attempts to create.

A = 15.0 and has an opposite angle of 22.61986494804042

B = 36.0 and has an opposite angle of 67.38013505195957

C = 39.0 and has an opposite angle of 90.0

Sum of angles = 180.0

The area of this triangle is 270.0

Attempt made to make triangle with sides 11.0 22.0 99.0

Sum of angles = NaN

Not a valid triangle

Triangle # 3 created.

A = 6.0 and has an opposite angle of 21.78678929826181

B = 10.0 and has an opposite angle of 38.21321070173819

C = 14.0 and has an opposite angle of 120.00000000000001

Sum of angles = 180.0

The area of this triangle is 25.98076211353316

Triangle # 4 created.

This is a right triangle and took 1 attempts to create.

A = 3.0 and has an opposite angle of 36.86989764584401

B = 4.0 and has an opposite angle of 53.13010235415599

C = 5.0 and has an opposite angle of 90.0

Sum of angles = 180.0

The area of this triangle is 6.0

4 triangles were created after 6373 attempts.

**Details**

Constants, like CONVERSION, use the *final* keyword when declared and use all capital letters.