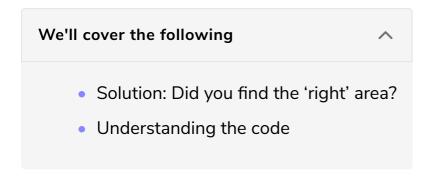
## Solution Review: Calculating the Area

In this review, solution of the challenge 'Calculating the Area' from the previous lesson is provided.



## Solution: Did you find the 'right' area? #

```
class rightAngleTriangle {
    //Define the member variables, constructor and
    // relevant area method
    private int length;
    private int height;
    public rightAngleTriangle(int 1, int h) {
        length = 1;
        height = h;
    }
    public double area() {
        return (length * height / 2.0);
    }
}
class challenge one {
    public static double test(rightAngleTriangle rt) {
        return rt.area();
    public static void main( String args[] ) {
        rightAngleTriangle one= new rightAngleTriangle(3,5);
        System.out.println("Area of right Angle traingle:" + test(one));
    }
}
```

## Understanding the code #

• Lines 4-5: The two **int** type variables **length** and **height** are created. Both

- variables are **private** to ensure they can only be accessed by **member** methods.
- Lines 7-9: The **overloaded constructor** is created. It takes as parameters, two **int** type variables. The **first** argument is assigned to the **length** and the **second** to the **height**. The constructor is **public** so that objects of type *rightAngleTriangle* can be made from any other class.
- Lines 12-14: This is the **method** for calculating the area of the triangle. Note that the method **does not** take in any parameters as it calculates the area using the *private members*. It calculates the area of the **right angle triangle** by the formula 0.5\* (length \* height). The **return type** for the method is double as it returns the **area** calculated.
- Lines 18-20: This creates a static method which simply takes in as a parameter, the rightAngleTriangle object and returns the answer of the area method.
  - The method was made **static** so it will belong to the whole class it is created in and not for every object of that class.

Let's display the message using the concepts of inheritance.