Solution Review: Coordinates of a Point

This lesson discusses the solution to the challenge given in the previous lesson.

```
package main
import (
        "fmt"
        "math"
)

type Point struct { /// struct of type Point
        X, Y float64
}

func (p *Point)Abs() float64 {  // method calculating absolute value
        return math.Sqrt(float64(p.X*p.X + p.Y*p.Y))
}

func (p *Point)Scale(s float64) {       // method to scale a point
        p.X = p.X * s
        p.Y = p.Y * s
        return
}

func main() {
        p1 := new(Point)
        p1.X = 3
        p1.Y = 4
        fmt.Printf("The length of the vector p1 is: %f\n", p1.Abs() ) // calling Abs() func

        p2:= &Point{4, 5}
        fmt.Printf("The length of the vector p2 is: %f\n", p2.Abs() ) // calling Abs() func

        p1.Scale(5) // calling Scale() fucn
        fmt.Printf("The length of the vector p1 is: %f\n", p1.Abs() ) // calling Abs() func
        fmt.Printf("Point p1 scaled by 5 has the following coordinates: X %f - Y %f", p1.X, p
}
```

Coordinates of a Point

In the above code, look at **line** 7. We make a struct of type `Point` that is a *2D* point. It has two fields `X` and `Y`, both of type *float64*.

Now, we need to write a method `Abs()`. See its header at **line 11**: `func (p`

`*Point)Abs() float64` . From its header, it's clear that this method can be called only by a pointer to the object of type `Point`. In the next line, the absolute of a point `p` is taken using the `Sqrt` function (from `math` package imported at **line 4**), and it is returned from the method.

Now, let's discuss the method `Scale()`. See its header at **line 15**: `func (p *Point)Scale(s float64)`. From its header, it's clear that this method can be called only by a pointer to the object of type `Point`. In the next line, both fields of `p` (`X` and `Y`) are multiplied with `s` (parameter) to scale the coordinates to the amount `s`.

Now, look at the `main` function. At **line 22**, we make a point `p1` using the `new()` function. In the next two lines, we set `X` and `Y` (**3** and **4**) fields of point `p1`. At **line 25**, we call the method `Abs()` on `p1` and print the resulting value, which is **5**.

Now, we make another point `p2`, using a literal expression at **line 27**. You may have noticed the operator `&`. This is because the method can be called by a *pointer*. In the next line, we call the method `Abs()` on point `p2` and print the resulting value, which is **6.403124**.

Now, at **line 30**, we are scaling point `p1` by a factor of **5**. In the next line, we again call the `Abs()` function on point `p1` to see the change and print the resulting value which is **25** this time. To see the scaled values of `X` and `Y` which are now **15** and **20**, we are printing them in the last line.

That's it for the solution. In the next lesson, you'll be attempting another challenge.