Solution Review: Make a Rectangle

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

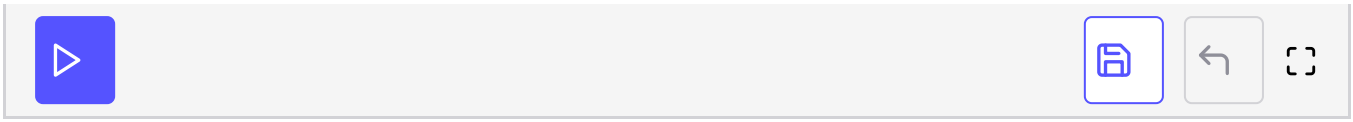
```
package main
import "fmt"

type Rectangle struct { // struct of type Rectangle
    length, width int
}

func (r *Rectangle) Area() int {    // method calculating area of rectangle
    return r.length * r.width
}

func (r *Rectangle) Perimeter() int { // method calculating perimeter of rectangle
    return 2* (r.length + r.width)
}

func main() {
    r1 := Rectangle{4, 3}
    fmt.Println("Rectangle is: ", r1)
    fmt.Println("Rectangle area is: ", r1.Area()) // calling method of area
    fmt.Println("Rectangle perimeter is: ", r1.Perimeter()) // calling method of perimeter
}
```



Make a Rectangle

In the above code, at **line 4**, we make a struct `Rectangle` containing two fields `length` and `width` of type *int*. Then, we have two important methods `Area()` and `Perimeter()`. Look at the header of `Area` method at **line 8** as: `func (r *Rectangle) Area() int` . It returns the area of rectangle `r`, i.e., `r.Length*r.Width`. Now, look at the header of `Perimeter` method at **line 12** as: `func (r *Rectangle) Perimeter() int` . It returns the perimeter of rectangle `r`, i.e., `2*(r.Length+r.Width)`. Let's see the `main` function. At **line 17**, we make a rectangle `r1` with literal expression `r1 := Rectangle{3,4}`. In the next line, we are printing `r1`, which will automatically print the fields of `r` enclosed within *braces*({}). At **line 19**, we are calling method `Area()` on `r1` and printing the returned value, which is **12**, the area of `r1`. At **line 20**, we are calling

method `Perimeter()` on `r1` and printing the returned value, which is **14**, the perimeter of `r1`.

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