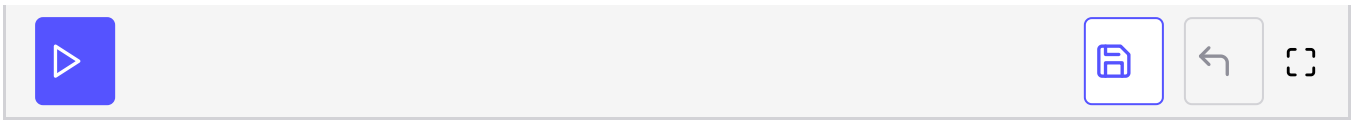
Solution Review: Advancing the Shapes Analysis

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

```
package main
import "fmt"

type Square struct {
        side float32
}

type Triangle struct {
        base   float32
        height float32
}

type AreaInterface interface {
        Area() float32
}

type PeriInterface interface {
        Perimeter() float32
}
func main() {
        var areaIntf AreaInterface
        var periIntf PeriInterface

        sq1 := new(Square)
        sq1.side = 5
        tr1 := new(Triangle)
        tr1.base = 3
        tr1.height = 5

        areaIntf = sq1
        fmt.Printf("The square has area: %f\n", areaIntf.Area())

        periIntf = sq1
        fmt.Printf("The square has perimeter: %f\n", periIntf.Perimeter())

        areaIntf = tr1
        fmt.Printf("The triangle has area: %f\n", areaIntf.Area())
}

func (sq *Square) Area() float32 {
        return sq.side * sq.side
}

func (sq *Square) Perimeter() float32 {
        return 4 * sq.side
}

func (tr *Triangle) Area() float32 {
```

```
func (tr *Triangle) Area() float32 {
        return 0.5 * tr.base*tr.height
}
```



Advancing the Shape Analysis

From **line 8** to **line 11**, we implement the `Triangle` type: from the formula for the *area*, we see that it needs two fields *base* and *height*, both of type `float32`.

See the implementation of `PeriInterface` from **line 17** to **line 18**: it needs a function `Perimeter()`, that also returns a `float32`.

From **line 48** to **line 50**, the `Triangle` type implements `AreaInterface` by defining the `Area()` method: given a parameter `tr` of type `*Triangle`, the area is given by the formula: `0.5 * tr.base * tr.height`.

From **line 44** to **line 46**, the `Square` type implements `PeriInterface` by defining the `Perimeter()` method: given a parameter `sq` of type `*Square`, the perimeter is given by the formula: `4 * sq.side`.

In the `main()` function, we define the variables `areaIntf` and `periIntf` of their respective interface types at **line 21** and **line 22**, respectively. From **line 24** to **line 28**, we define variables `sq1` and `tr1` of the `Square` and `Triangle` type with `new()`, and fill in their properties.

At **line 30**, we demonstrate that we can assign a `Square` variable `sq1` to a variable `areaIntf` of type `AreaInterface` because `Square` implements `AreaInterface`. So, we can call the `Area()` method on `areaIntf` (see **line 31**).

Similarly, at **line 33**, we assign a `Square` variable `sq1` to a variable `periIntf` of type `PeriInterface` as `Square` implements `PeriInterface`. So, we can call the `Perimeter()` method on `periIntf` (see **line 34**).

At **line 36**, we demonstrate that we can assign a `Triangle` variable `tr1` to a variable `areaIntf` of type `AreaInterface`, because `Triangle` implements `AreaInterface`. So, we can call the `Area()` method on `areaIntf` (see **line 37**).

That's it about the solution. In the next lesson, there's another challenge for you to solve.