**Go Interfaces**

Go provides another datatype called as interface

It represents a set of method signatures

The struct data type implements these interface to have a method definition for the method of the interface

Syntex

type interface\_type name{

method name1[return type]

method name2[return type]

method name3[return type]

}

type struct\_name struct{

variables

}

// implement interface methods

func (struct\_name\_variable struct\_name) method\_name1() [return type]{

// Method implementation

}

func (struct\_name\_variable struct\_name) method\_name() [return type] {

// Method implementation  
}

package main

import (

   "fmt"

   "math"

)

/\* define an interface \*/

type Shape interface {

   area() float64

}

/\* define a circle \*/

type Circle struct {

   x, y, radius float64

}

/\* define a rectangle \*/

type Rectangle struct {

   width, height float64

}

/\* define a method for circle (implementation of Shape.area())\*/

func (circle Circle) area() float64 {

   return math.Pi \* circle.radius \* circle.radius

}

/\* define a method for rectangle (implementation of Shape.area())\*/

func (rect Rectangle) area() float64 {

   return rect.width \* rect.height

}

/\* define a method for shape \*/

func getArea(shape Shape) float64 {

   return shape.area()

}

func main() {

   circle := Circle{x: 0, y: 0, radius: 5}

   rectangle := Rectangle{width: 10, height: 5}

   fmt.Printf("Circle area: %f\n", getArea(circle))

   fmt.Printf("Rectangle area: %f\n", getArea(rectangle))

}

Circle area: 78.539816

Rectangle area: 50.000000