Московский государственный технический университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»
Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Парадигмы и конструкции языков программирования»

Отчет по лабораторной работе № 3
«Работа со структурами и методами в Go»

Выполнил:
студент группы ИУ5-31Б
Князев А.М.

Проверил:
преподаватель каф. ИУ5
Нардид А.Н.

Москва, 2024 г.

Описание задания

1. Создать структуру *Rectangle* с полями:

- *width* (ширина)
- *height* (высота)

2. Реализовать функцию *validateDimensions* для проверки корректности значений ширины и высоты:

- Значения должны быть **положительными числами**.
- В случае некорректных данных функция возвращает сообщение об ошибке.

3. Реализовать функции и методы для работы со структурой *Rectangle*:

- **Функция *getArea***: вычисление площади прямоугольника.
- **Метод *Perimeter***: вычисление периметра прямоугольника.

4. Создать вложенную структуру *ColoredRectangle*, которая включает в себя:

- Поле типа *Rectangle*.
- Дополнительное поле *color* (цвет прямоугольника).

5. Проверить работу программы на нескольких прямоугольниках:

- Инициализация значениями при объявлении.
- Инициализация с использованием переменных.
- Инициализация со значениями по умолчанию (0, 0).

6. Реализовать сравнение двух прямоугольников.

Текст программы

Файл *lab3.go*

```go
package main

import (
    "errors"
    "fmt"
)

type Rectangle struct {
    width  float64
    height float64
```

```
}

func validateDimensions(width, height float64) error {
    if width <= 0 || height <= 0 {
        return errors.New("ширина и высота должны быть положительными числами")
    }
    return nil
}

func getArea(r Rectangle) float64 {
    return r.width * r.height
}

func (r Rectangle) Perimeter() float64 {
    return 2 * (r.width + r.height)
}

type ColoredRectangle struct {
    Rectangle Rectangle
    color     string
}

func main() {

    rect1 := Rectangle{width: 5, height: 10}
    fmt.Println("Rectangle 1:", rect1)

    width := 7.5
    height := 12.0
    if err := validateDimensions(width, height); err != nil {
        fmt.Println("Ошибка:", err)
        return
    }
    rect2 := Rectangle{width: width, height: height}
    fmt.Println("Rectangle 2:", rect2)

    var rect3 Rectangle
    if err := validateDimensions(rect3.width, rect3.height); err != nil {
        fmt.Println("Ошибка в Rectangle 3:", err)
    } else {
        fmt.Println("Rectangle 3:", rect3)
    }

    fmt.Println("Area of rect1:", getArea(rect1))
    fmt.Println("Perimeter of rect1:", rect1.Perimeter())

    fmt.Println("Area of rect2:", getArea(rect2))
    fmt.Println("Perimeter of rect2:", rect2.Perimeter())

    coloredRect := ColoredRectangle{Rectangle: Rectangle{width: 3, height: 4},
color: "red"}
```

```
    if err := validateDimensions(coloredRect.Rectangle.width,
coloredRect.Rectangle.height); err != nil {
        fmt.Println("Ошибка:", err)
    } else {
        fmt.Println("Colored Rectangle:", coloredRect)
        fmt.Println("Area of coloredRect:", getArea(coloredRect.Rectangle))
        fmt.Println("Perimeter of coloredRect:",
coloredRect.Rectangle.Perimeter())
    }
    rect4 := Rectangle{width: 5, height: 10}
    rect5 := Rectangle{width: 5, height: 10}
    fmt.Println("rect4 == rect5:", rect4 == rect5)
}
```

Экранные формы с примерами выполнения программы