МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «Белгородский государственный
национальный исследовательский университет»

Лабораторная работа №1
дисциплина «Современные методы, среды и языки программирования»
по теме «Создание и реализация интерфейсов классов в программах на языке Java»

Выполнил: студент группы 12002135 Макаров Д.С.
Проверил:

Белгород 2021

Лабораторная работа №1

«Создание и реализация интерфейсов классов в программах на языке Java»

Цель работы:Научиться писать код определения классов и интерфейсов на языке Java в соответствии с основными принципами объектно-ориентированного программирования, а также выполнять реализацию созданных классов и интерфейсов для решения типовых задач.

Ход работы

Необходимо выполнить следующие задания на языке Java:

- Разработать программу, в которой в строки, помеченные комментариями //1 и //2, записать оптимальные вызовы метода execute():

```
public class Hope {
 public static void execute(){}
 private void taste(){ //1
 }
}
public class Dream {
 private Float taste(){ //2
 }
}
```

- Разработать программу с реализацией интерфейса:

```
interface Application {
 void hello();
}
```

- Разработать программу, в которой если возможно, создать подкласс абстрактного класса и обосновать свое решение:

```
abstract class Cook {
 protected abstract void sell();
}
```

- Разработать программу с реализацией метода hashCode для следующего класса:

```
class Quest {
 private int a;
 private short b;
}
```

- Разработать программу с реализацией метода equals для следующего класса

```
class Bus {
 protected String type;
}
```

- f) `interface` Корабль <- `abstract class` Военный Корабль <- `class` Авианосец.

Приложение

Содержимое файла task1.go

```
package main

import "fmt"

/*
Golang - структурный язык, но он позволяет создавать иерархию структур вкладывая их друг в друга
(тем самым реализовывать аналог наследования) и прикреплять функции к структурам (создавать методы).
*/

type Hope struct {
        hopeCount int
}

//В Java статическая функция, может выполняться без контекста (инстанса класса), в golang методы нельзя вызвать без
↪  экземпляра структуры.
//Поэтому static метод execute можно заменить на функцию.

func execute() {
        fmt.Println("Hope executed")
}

func (h Hope) taste() {
        execute()
        fmt.Println("Hope tasted")
}

type Dream struct {
        dream float32
}

func (d Dream) taste() float32 {
        execute()
        return d.dream
}

func main() {
        fmt.Println("task1")

        hope := Hope{}
        dream := Dream{}

        hope.taste()
        dream.taste()
}
```

Содержимое файла task2.go

```
package main

import "fmt"

type Application interface {
        hello()
}

type ApplicationStruct struct{}

func (as ApplicationStruct) hello() {
        fmt.Println("Hello World!")
}

func callHello(a Application) {
        a.hello()
}

func main() {
        fmt.Println("task2")
        as := ApplicationStruct{}
        callHello(as)
}
```

Содержимое файла task3.go

```go
package main

import "fmt"

type AbstractCook struct{}

type Cook struct {
        ac AbstractCook
}

func (c Cook) sell() {
        fmt.Println("sell")
}

func main() {
        fmt.Println("task3")
        c := Cook{}
        c.sell()
}
```

Содержимое файла task45.go

```go
package main

import "fmt"

/*
        В golang нет необходимости реализовывать методы hashCode и equals.
        Все базовые типа языка имеют реализацию hash функции.
        Структуры состоящие из базовых типов, так же могут быть хешированы и использованы в хеш таблицах (map)

        Так же сравнение структур одного типа выполняет компилятор, структуры различного типа сравнивать нельзя.
*/

type Quest struct {
        a int32
        b int16
}

type Bus struct {
        type_var string
}

func main() {
        fmt.Println("task4")

        quest1 := Quest{a: 10, b: 20}
        quest2 := Quest{a: 10, b: 20}
        quest3 := Quest{a: 20, b: 20}

        fmt.Println(quest1 == quest2)
        fmt.Println(quest2 == quest3)

        fmt.Println("task5")

        bus1 := Bus{type_var: "bus1"}
        bus2 := Bus{type_var: "bus2"}

        fmt.Println(bus1 == bus2)
}
```

Содержимое файла task6.go

```go
package main

import "fmt"

/*
        Вариант f
        interface Корабль <- abstract class Военный Корабль <- class Авианосец.
*/

type Ship interface {
        float()
}

type Battleship struct {
```

```
        mainGunGauge int
}

func (b Battleship) float() {
        fmt.Println("Battleship is floating")
}

// В golang нет наследования, структуры вкладываются друг в друга.

type Carrier struct {
        Battleship
        airstripCount int
}

func (c Carrier) float() {
        fmt.Println("Carier is floating")
}

func checkShip(s Ship) {
        fmt.Println("Ship ok!")
}

func main() {
        c := Carrier{airstripCount: 2}

        // Полиморфизм как в Java, C# и C++ тоже отсутствует.
        c.Battleship.float()
        c.float()

        checkShip(c)
}
```