Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ   
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерного проектирования

Кафедра проектирования информационно-компьютерных систем

Отчёт

по лабораторной работе №4

на тему:

**ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ (ООП)**

Проверил \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Ф.В. Усенко

(подпись)

Выполнил \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ П.М. Василевич

(подпись) гр. 214302

Минск, 2024

**Задание:** Напишите интерфейсы для различных типов транспорта (Drivable, Flyable, Sailable). Создайте классы, которые реализуют несколько интерфейсов (например, AmphibiousCar, FlyingBoat) и добавляют дополнительные методы для переключения режимов и управления.

**Решение**

Интерфейс Flying

public interface Flying {  
 String flyInfo();  
}

Интерфейс Swimming

public interface Swimming {  
 String swimInfo();  
}

Интерфейс Driving

public interface Driving {  
 String driveInfo();  
}

Интерфейс Diving

public interface Diving {  
 String diveInfo();  
}

Класс AmphibiousCar

class AmphibiousCar(var speed: Int = 0, var status: Char = 'n') : Driving, Swimming {

fun startDriving() {

if (status == 'r') println("Машинка-амфибия уже едет")

else {

speed = 60;

status = 'r'

println(

"Машинка-амфибия поехала. Ее скорость составляет $speed" +

" км/ч"

)

}

}

fun startSwimming() {

if (status == 's') println("Машинка-амфибия уже плывет")

else {

speed = 30;

status = 's'

println(

"Машинка-амфибия поплыла. Ее скорость составляет $speed" +

" км/ч"

)

}

}

override fun driveInfo(): String {

if (status == 'r') {

return "Машина-амфибия едет со скоростью $speed км/ч";

} else return "Машина-амфибия не едет"

}

override fun swimInfo(): String {

if (status == 's') {

return "Машина-амфибия плывет по поверхности скоростью $speed км/ч";

} else return "Машина-амфибия не плывет"

}

fun increaseSpeed(s: Int) {

when (status) {

'r' ->

(if (speed + s > 600) println("\tСкорость машины-амфибии не может превышать 600 км/ч")

else {

speed += s;

println("Скорость машины-амфибии увеличилась до $speed км/ч")

})

's' ->

(if (speed + s > 250) println("\tСкорость плывущей машины-амфибии не может превышать 250 км/ч")

else {

speed += s;

println("Скорость машины-амфибии увеличилась до $speed км/ч")

})

'n' -> (println("\tСначала начните движение"))

}

}

fun decreaseSpeed(s: Int) {

when (status) {

'r' ->

(if (speed - s < 0) {

status = 'n'

speed = 0

println("Машина остановилась")

} else {

speed -= s;

println("Скорость машины-амфибии снизилась до $speed км/ч")

})

's' ->

(if (speed - s < 0) println("\tСкорость плывущей машины-амфибии не может быть меньше 0 км/ч")

else {

speed -= s;

println("Скорость машины-амфибии снизилась до $speed км/ч")

})

'n' -> (println("\tСначала начните движение"))

}

}

}

Класс Plane

class Plane(var speed: Int = 0, var height: Int = 0, var status: Char = 'n') : Flying, Driving {

fun startDriving() {

if (status == 'r') println("Самолет уже едет")

else {

speed = 70;

height = 0

status = 'r'

println(

"Самолет поехал. Его скорость составляет $speed" +

" км/ч, его высота - $height м"

)}

}

fun startFlying(h: Int = 6000) {

if (status == 'f') println("Самолет уже летит")

else {

speed = 500;

height = h

status = 'f'

println(

"Самолет полетел. Его скорость составляет $speed" +

" км/ч, его высота - $height м"

)}

}

override fun driveInfo(): String {

if (status == 'r') {

return "Самолет едет со скоростью $speed км/ч";

} else return "Самолет не едет"

}

override fun flyInfo(): String {

if (status == 'f') {

return "Самолет летит на высоте $height м со скоростью $speed км/ч";

} else return "Самолет не летит"

}

fun increaseSpeed(s: Int) {

when (status) {

'r' ->

(if (speed + s > 70) println("\tСкорость самолета не может превышать 70 км/ч")

else {

speed += s;

println("Скорость самолета увеличилась до $speed км/ч")

})

'f' ->

(if (speed + s > 1500) println("\tСкорость летящего самолета не может превышать 1500 км/ч")

else {

speed += s;

println("Скорость самолета увеличилась до $speed км/ч")

})

'n' -> (println("\tСначала начните движение"))

}

}

fun decreaseSpeed(s: Int) {

when (status) {

'r' ->

(if (speed - s < 0) {

status = 'n'

speed = 0

height = 0

println("Самолет остановился")

} else {

speed -= s;

println("Скорость самолета снизилась до $speed км/ч")

})

'f' ->

if (speed - s < 50) println("\tСкорость летящего самолета не может быть меньше 50 км/ч")

else {

speed -= s;

println("Скорость самолета снизилась до $speed км/ч")

}

'n' -> (println("\tСначала начните движение"))

}

}

fun decreaseHeight(h: Int) {

when (status) {

'n' -> println("\tСначала начните движение")

'r' -> println("\tЧтобы уменьшить высоту полета самолета, он должен быть в воздухе")

else -> (if (height - h < 0) startDriving()

else {

height -= h;

println("Высота полета самолета уменьшилась до $height м")

})

}

}

fun increaseHeight(h: Int) {

when (status) {

'n' -> println("\tСначала начните движение")

'r' -> startFlying(h)

else -> (if (height + h > 10000) println("\tВысота самолета не может превышать 10000 м")

else {

height += h;

println("Высота полета самолета увеличилась до $height м")

})

}

}

}

Класс Submarine

class Submarine(var speed: Int = 0, var depth: Int = 0, var status: Char = 'n') : Swimming, Diving {  
 fun startDiving(d: Int = 1000) {  
 if (status=='d') println("Субмарина уже нырнула")  
 else{  
 speed = d;  
 depth = 1000;  
 status = 'd'  
 println(  
 "Субмарина нырнула на глубину. Ее скорость составляет $speed" +  
 " км/ч, глубина - $depth м"  
 )  
 }  
 }  
  
 fun startSwimming() {  
 if (status=='s') println("Субмарина уже плывет")  
 else{  
 speed = 20;  
 depth = 0  
 status = 's'  
 println(  
 "Субмарина поплыла по поверхности. Ее скорость составляет $speed" +  
 " км/ч, глубина - $depth м"  
 )}  
 }  
  
 override fun diveInfo(): String {  
 if (status == 'd') {  
 return "Субмарина плывет на глубине $depth м со скоростью $speed км/ч";  
 } else return "Субмарина не плывет на глубине"  
 }  
  
 override fun swimInfo(): String {  
 if (status == 's') {  
 return "Субмарина плывет по поверхности скоростью $speed км/ч";  
 } else return "Субмарина не плывет"  
 }  
  
 fun increaseSpeed(s: Int) {  
 when (status) {  
 'd' -> (  
 if (speed + s > 120) println("\tСкорость нырнувшей субмарины не может превышать 120 км/ч")  
 else {  
 speed += s;  
 println("Скорость субмарины увеличилась до $speed км/ч")  
 })  
  
  
 's' ->  
 (if (speed + s > 30) println("\tСкорость плывущей субмарины не может превышать 30 км/ч")  
 else {  
 speed += s;  
 println("Скорость субмарины увеличилась до $speed км/ч")  
 })  
  
 'n' -> (println("\tСначала начните движение"))  
 }  
 }  
  
 fun decreaseSpeed(s: Int) {  
 when (status) {  
 'd' ->  
 (if (speed - s < 0) println("\tСкорость нырнувшей субмарины не может быть меньше 0 км/ч")  
 else {  
 speed -= s;  
 println("Скорость субмарины снизилась до $speed км/ч")  
 })  
  
 's' ->  
 (if (speed - s < 0) println("\tСкорость плывущей субмарины не может быть меньше 0 км/ч")  
 else {  
 speed -= s;  
 println("Скорость субмарины снизилась до $speed км/ч")  
 })  
  
 'n' -> (println("\tСначала начните движение"))  
 }  
 }  
  
 fun decreaseDepth(h: Int) {  
 if (status != 'd') println("\tСначала нужно нырнуть")  
 else {  
 if (depth - h < 0) startSwimming()  
 else {  
 depth -= h;  
 println("Субмарина поднялась до глубины $depth м")  
 }  
 }  
 }  
  
 fun increaseDepth(h: Int) {  
 when (status) {  
 'n' -> println("\tСначала начните движение")  
 's' -> startDiving(h)  
 else -> (if (depth + h > 12000) println("\tСубмарина не может опуститься глубже 12000 м")  
 else {  
 depth += h;  
 println("Субмарина опустилась до глубины $depth м")  
 })  
 }  
 }  
}

Класс SuperCar

class SuperCar(var speed: Int = 0, var height: Int = 0, var depth: Int = 0, var status: Char = 'n') : Driving, Flying,  
 Swimming, Diving {  
 fun startDiving(d:Int=800) {  
 if (status == 'd') println("Суперкар уже нырнул")  
 else {  
 speed = 70;  
 depth = d;  
 height = 0  
 status = 'd'  
 println(  
 "Суперкар нырнул на глубину. Его скорость составляет $speed" +  
 " км/ч, глубина - $depth м"  
 )}  
 }  
  
 fun startDriving() {  
 if (status == 'r') println("Суперкар уже едет")  
 else {  
 speed = 60;  
 height = 0  
 depth = 0  
 status = 'r'  
 println(  
 "Суперкар поехал. Его скорость составляет $speed" +  
 " км/ч, его высота и глубина - $height м"  
 )}  
 }  
  
 fun startFlying(h: Int = 4500) {  
 if (status == 'f') println("Суперкар уже летит")  
 else {  
 speed = 400;  
 height = h  
 depth = 0  
 status = 'f'  
 println(  
 "Суперкар полетел. Его скорость составляет $speed" +  
 " км/ч, его высота - $height м"  
 )}  
 }  
  
 fun startSwimming() {  
 if (status == 's') println("Суперкар уже плывет")  
 else {  
 speed = 45;  
 depth = 0  
 height = 0  
 status = 's'  
 println(  
 "Суперкар поплыл по поверхности. Его скорость составляет $speed" +  
 " км/ч, глубина и высота - $depth м"  
 )}  
 }  
  
 override fun diveInfo(): String {  
 if (status == 'd') {  
 return "Суперкар плывет на глубине $depth м со скоростью $speed км/ч";  
 } else return "Cуперкар не плывет на глубине"  
 }  
  
 override fun driveInfo(): String {  
 if (status == 'r') {  
 return "Суперкар едет со скоростью $speed км/ч";  
 } else return "Cуперкар не едет"  
 }  
  
 override fun flyInfo(): String {  
 if (status == 'f') {  
 return "Суперкар летит на высоте $height м со скоростью $speed км/ч";  
 } else return "Cуперкар не летит"  
 }  
  
 override fun swimInfo(): String {  
 if (status == 's') {  
 return "Суперкар плывет по поверхности скоростью $speed км/ч";  
 } else return "Cуперкар не плывет"  
 }  
  
 fun increaseSpeed(s: Int) {  
 when (status) {  
 'd' -> (  
 if (speed + s > 90) println("\tСкорость нырнувшего суперкара не может превышать 90 км/ч")  
 else {  
 speed += s;  
 println("Скорость суперкара увеличилась до $speed км/ч")  
 })  
  
 'r' ->  
 (if (speed + s > 1200) println("\tСкорость суперкара не может превышать 1200 км/ч")  
 else {  
 speed += s;  
 println("Скорость суперкара увеличилась до $speed км/ч")  
 })  
  
 'f' ->  
 (if (speed + s > 2000) println("\tСкорость летящего суперкара не может превышать 2000 км/ч")  
 else {  
 speed += s;  
 println("Скорость суперкара увеличилась до $speed км/ч")  
 })  
  
 's' ->  
 (if (speed + s > 300) println("\tСкорость плывущего суперкара не может превышать 300 км/ч")  
 else {  
 speed += s;  
 println("Скорость суперкара увеличилась до $speed км/ч")  
 })  
  
 'n' -> (println("\tСначала начните движение"))  
 }  
 }  
  
 fun decreaseSpeed(s: Int) {  
 when (status) {  
 'd' ->  
 (if (speed - s < 0) println("\tСкорость нырнувшего суперкара не может быть меньше 0 км/ч")  
 else {  
 speed -= s;  
 println("Скорость суперкара снизилась до $speed км/ч")  
 })  
  
 'r' ->  
 (if (speed - s < 0) {  
 status = 'n'  
 speed = 0  
 depth = 0  
 height = 0  
 println("Машина остановилась")  
 } else {  
 speed -= s;  
 println("Скорость суперкара снизилась до $speed км/ч")  
 })  
  
 'f' ->  
 if (speed - s < 50) println("\tСкорость летящего суперкара не может быть меньше 50 км/ч")  
 else {  
 speed -= s;  
 println("Скорость суперкара снизилась до $speed км/ч")  
 }  
  
 's' ->  
 (if (speed - s < 0) println("\tСкорость плывущего суперкара не может быть меньше 0 км/ч")  
 else {  
 speed -= s;  
 println("Скорость суперкара снизилась до $speed км/ч")  
 })  
  
 'n' -> (println("\tСначала начните движение"))  
 }  
 }  
  
 fun decreaseHeight(h: Int) {  
 if (status != 'f') println("\tСначала нужно взлететь")  
 else {  
 if (height - h < 0) println("\tВысота суперкара не может быть меньше 0 м")  
 else {  
 height -= h;  
 println("Высота полета суперкара снизилась до $height м")  
 }  
 }  
 }  
  
 fun increaseHeight(h: Int) {  
 when (status) {  
 'n' -> println("\tСначала начните движение")  
 's', 'r' -> startFlying(h)  
 'd' -> println("\tСначала нужно всплыть")  
 else -> (if (height + h > 9000) println("\tВысота полета суперкара не может превышать 9000 м")  
 else {  
 height += h;  
 println("Высота полета суперкара увеличилась до $height м")  
 })  
 }  
 }  
  
 fun decreaseDepth(h: Int) {  
 if (status != 'd') println("\tСначала нужно нырнуть")  
 else {  
 if (depth - h < 0) startSwimming()  
 else {  
 depth -= h;  
 println("Суперкар поднялся до глубины $depth м")  
 }  
 }  
 }  
  
 fun increaseDepth(h: Int) {  
 when (status) {  
 'n' -> println("\tСначала начните движение")  
 's' -> startDiving(h)  
 'f','r' -> println("\tЧтобы опуститься на глубину, надо быть на/в воде")  
 else -> (if (height + h > 9000) println("\tГлубина погружения суперкара не может превышать 9000 м")  
 else {  
 height += h;  
 println("Высота полета суперкара увеличилась до $height м")  
 })  
 }  
 }  
}

Класс Main

fun main() {

val supercar = SuperCar()

println(supercar.driveInfo());

println(supercar.swimInfo());

println(supercar.flyInfo());

println(supercar.diveInfo());

supercar.startFlying();

println(supercar.flyInfo());

supercar.increaseSpeed(100)

supercar.increaseHeight(9000)

supercar.increaseHeight(1000)

supercar.decreaseHeight(4000)

supercar.startSwimming()

supercar.increaseDepth(400)

supercar.increaseSpeed(15)

supercar.decreaseSpeed(600)

supercar.decreaseDepth(5000)

supercar.startDriving()

supercar.driveInfo()

supercar.decreaseSpeed(500)

val plane = Plane()

plane.decreaseHeight(40)

plane.startDriving()

println(plane.flyInfo())

println(plane.driveInfo())

plane.increaseHeight(5000)

plane.decreaseSpeed(10000)

plane.decreaseSpeed(300)

plane.decreaseHeight(12000)

plane.decreaseSpeed(500)

val amphibiousCar = AmphibiousCar()

amphibiousCar.startSwimming()

amphibiousCar.decreaseSpeed(500)

amphibiousCar.startDriving()

amphibiousCar.decreaseSpeed(20)

amphibiousCar.decreaseSpeed(900)

val submarine = Submarine()

submarine.startSwimming()

println(submarine.diveInfo())

println(submarine.swimInfo())

submarine.increaseSpeed(20)

submarine.increaseDepth(300)

submarine.startDiving()

submarine.decreaseDepth(12000)

}

**Контролные вопросы**

1. Что такое класс в Kotlin, и как он объявляется?

Класс в Kotlin объявляется с использованием ключевого слова class. Класс может содержать свойства и методы, а также быть шаблоном для создания объектов.

Пример:

class Car(val brand: String, val model: String, var year: Int) {

fun startEngine() {

println("Engine started for $brand $model")

}

}

Здесь класс Car имеет свойства brand, model и year, а также метод startEngine.

1. Как создать объект класса в Kotlin? Приведите пример.

Объекты создаются с помощью вызова конструктора класса. В Kotlin ключевое слово new не требуется.

Пример:

val myCar = Car("Toyota", "Corolla", 2020)

myCar.startEngine() // Выводит "Engine started for Toyota Corolla"

1. Что такое первичный конструктор, и как он используется для инициализации свойств класса?

В Kotlin первичный конструктор объявляется прямо в заголовке класса и используется для инициализации свойств класса.

Пример:

class Car(val brand: String, val model: String, var year: Int)