Министерство образования Республики Беларусь

Учреждение образования

«Белорусский государственный университет

информатики и радиоэлектроники»

Учебная дисциплина «Современные языки программирования»

Отчёт

по лабораторной работе №1

«Использование языка программирования Kotlin»

Выполнил студент группы 310101

Криницкий М.А.

Проверил:

Усенко Ф. В.

Минск 2024

Цель работы: Выполнить разработку приложения с использованием языка программирования Kotlin.

Индивидуальное задание: Добавить класс Гладиолус. Добавить класс Лепесток, принадлежащий классу Бутон и класс Лист, принадлежащий классу Цветок. Обеспечить создание Объектов Лепесток одновременно с объектами Бутон. Реализовать метод Засохнуть() у класса Цветок.

Листинг программы с комментариями

fun main() {  
  
 val Menu=Menu() //создание объекта класса меню  
 Menu.menu() //вызов метода меню  
  
}

class Menu {  
 private val flowers: MutableList<Flower> = mutableListOf() //создание изменяемого списка  
 fun menu()  
 {  
 println("Введите ваше имя:")  
 val Human=Human(readLine()!!) // ввод имени человека и передача его в конструктор класса Human  
 println("Здраствуйте, " + Human.getName())  
// :? - оператором Элвиса, используемый для проверки на null  
// !! - используется для явного указания того, что объект, переменная не ссылается на null  
 var t:Boolean=true  
  
 while(t) //цикл  
 {  
  
 println(  
 "Выберите цветок, который хотите посадить:\n" +  
 "1) Роза\n" +  
 "2) Гладиолус\n" +  
 "3) Фиалка"  
 )  
 val number\_of\_flower = readLine()!!.toInt()  
 if (number\_of\_flower == 1) {  
 Human.setFlower(number\_of\_flower, flowers)  
 } else if (number\_of\_flower == 2) {  
 Human.setFlower(number\_of\_flower, flowers)  
 } else if (number\_of\_flower == 3) {  
 Human.setFlower(number\_of\_flower, flowers)  
 } else {  
 println("Такого цветка нет!")  
 }  
  
 val test:Boolean=true //создание булевой переменной  
 var k:Int=0  
  
  
 while(test) { //выбор пункта меню и вызов соответствующих методов  
 println("1. Нажмите '1', чтобы полить цветок")  
 println("2.Нажмите '2', чтобы понюхать цветок")  
 println("3.Нажмите '3', чтобы посадить новый цветок")  
 println("4.Нажмите '4', чтобы выйти из программы")  
  
 val choose = readLine()!!.toInt() //явное указание типа считываемой информации  
  
 if (choose == 1) {  
 Human.waterFlowers(flowers, choose,k) //вызов метода класса Human, к которому принадлежит объект Human  
 k++  
 }  
 else if(choose ==1 && k>=1)  
 {  
 Human.waterFlowers(flowers,choose,k)  
 k++  
 }  
 else if (choose == 2 && k>=1) {  
 Human.smellFlower(flowers, choose)  
 k++  
 }  
 else if(choose==2 && k==0)  
 {  
 println("Сначала полейте цветок!")  
 }  
 else if (choose == 3) {  
  
 break  
 }  
 else if (choose == 4) {  
  
 for (flower in flowers) {  
 flower.dryUp()  
 }  
 t=false  
 break  
 }  
 else {  
 println("Такого пункта нет!")  
 }  
  
 }  
  
 }  
 }  
}

class Human (private val name:String) {  
  
 fun getName():String  
 {  
 return name  
 }  
  
  
 fun setFlower(number\_of\_flower:Int,list:MutableList<Flower>){  
  
 if (number\_of\_flower==1)  
 {  
 val your\_flower=Rose(number\_of\_flower)  
 your\_flower.set()  
 list.add(your\_flower) //добавление объекта в изменяемый список  
  
 }  
 else if (number\_of\_flower==2)  
 {  
 val your\_flower=Gladiolus(number\_of\_flower)  
 your\_flower.set()  
 list.add(your\_flower)  
 }  
 else if (number\_of\_flower==3)  
 {  
 val your\_flower=Violet(number\_of\_flower)  
 your\_flower.set()  
 list.add(your\_flower)  
 }  
 else {  
 println("Такого цветка нет!")  
 }  
  
  
 }  
  
  
  
 fun waterFlowers(flower:MutableList<Flower>,number:Int,k:Int ){  
 val flower=flower.last()  
 if(k==0) {  
  
 println("Цветок успешно полит!")  
 flower.releaseList()  
 flower.releaseBud(number)  
  
 }  
 else  
 {  
 flower.increaseHeight()  
 }  
  
 }  
 fun smellFlower(flower:MutableList<Flower>,number:Int ){  
 val flower=flower.last()  
 flower.releaseBud(number)  
  
 }  
  
}

abstract class Flower (){  
  
 abstract var height:Int  
 abstract fun releaseList()  
 abstract fun increaseHeight()  
 abstract fun releaseBud(number:Int)  
 abstract fun dryUp()  
  
}

class Petal() {  
 private var color:String="Цвет"  
 fun setColor(color:String){  
 this.color = color  
 }  
 fun getColor():String{  
 return color  
 }  
}

class List {  
 public var length:Int=1  
}

class Bud(val number\_of\_flower:Int){  
 var petal:Petal? = Petal() //создание объекта лепесток - "petal"  
 fun realisePetal()  
 {  
  
 if(number\_of\_flower==1) {  
 petal?.setColor("Красный")  
 println("Появились лепестки, цвета: ${petal?.getColor()}")  
 }  
 else if(number\_of\_flower==2)  
 {  
 petal?.setColor("Нежно-сиреневый")  
 println("Появились лепестки, цвета: ${petal?.getColor()}")  
 }  
 else if(number\_of\_flower==3)  
 {  
 petal?.setColor("Фиолетовый")  
 println("Появились лепестки, цвета: ${petal?.getColor()}")  
 }  
 }  
  
  
 public val smell:String="Приятный"  
 public fun ThrowPetal(){  
  
 petal=null //обнуление ссылки на объкт "petal"  
 println("Лепестки завяли")  
  
 }  
  
}

class Gladiolus(number\_of\_flower:Int) :Flower(){  
 var bud:Bud?=Bud( number\_of\_flower)  
 var list:List?= List()  
 override var height: Int=7  
 override fun releaseList() {  
 val list = List()  
 println("Появились первые листья гладиолуса!")  
 println("Длина листа: ${list!!.length}")  
 }  
  
 fun set()  
 {  
 println("Гладиолус посажен!")  
 }  
  
 override fun increaseHeight() {  
 val max\_height=70  
 if (height<max\_height) {  
 height+=4  
 println("Ваш цветок увеличился на 4 см: Длина цветка - $height")  
 }  
 else {println("Цветок больше не выростет: Длина цветка: ${height}")}  
  
 val max\_length=10  
 if (list!!.length<max\_length) {  
 list!!.length+=6  
 println("Листья цветка увеличились на 1 см: Длина листьев - ${list!!.length}")  
 }  
 else {println("Листья больше не увеличатся: Длина листьев: ${list!!.length}")}  
 }  
  
 override fun releaseBud(number:Int) {  
  
 if(number==1) {  
 println("Появился бутон Гладиолуса!")  
 bud!!.realisePetal()  
 }  
 else if(number==2){  
 println("Запах ${bud!!.smell}")  
 }  
 }  
  
 override fun dryUp() {  
 println("Гладиолус засыхает")  
 bud?.ThrowPetal()  
 bud=null  
 println("Бутон гладиолуса засох")  
 list=null  
 println("Листья гладиолуса засохли\n")  
 println("------------------------------------\n")  
  
 }  
}

Скриншоты работы программы представлены на рисунках 1,2.

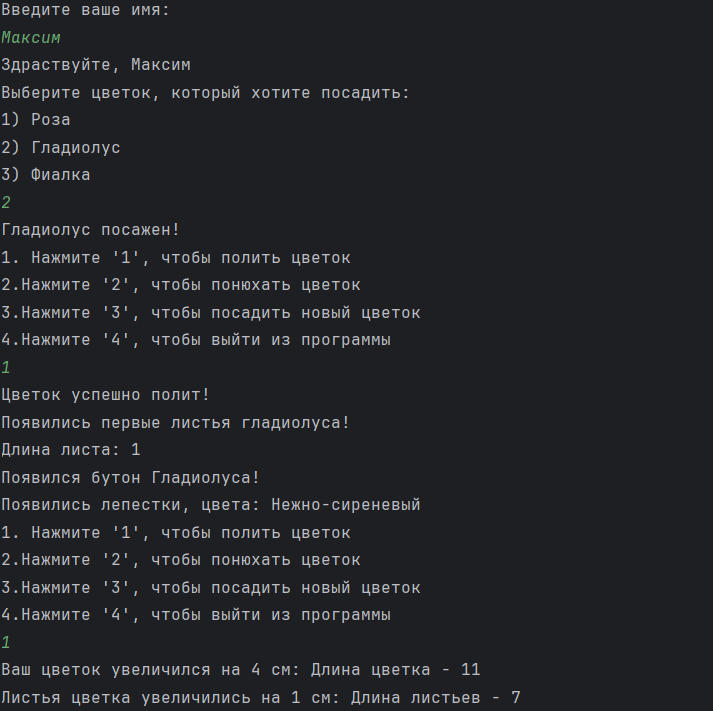


Рис.1 – Скриншот работы программы

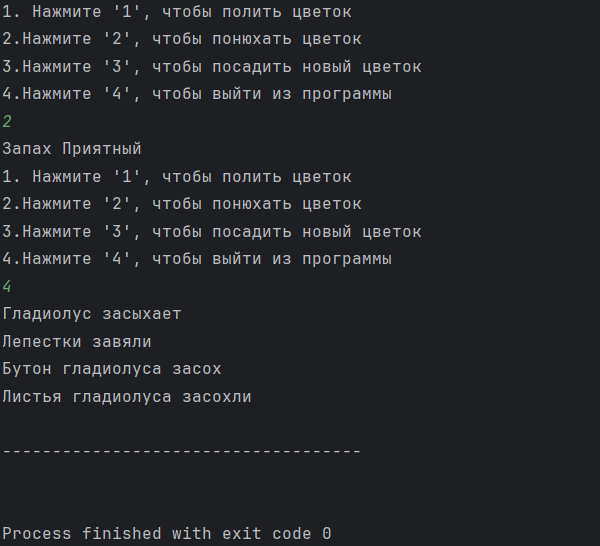


Рис.2 – Скриншот работы программы

**Вывод**

В ходе разработки приложения на Kotlin была создана иерархия классов, отражающая структуру гладиолуса. Реализованы классы Гладиолус, Бутон, Лепесток и Лист, которые отражают различные аспекты цветка. Установлены связи между классами, позволяющие лепесткам создаваться одновременно с бутоном. Также реализован метод Засохнуть() в классе Гладиолус, демонстрирующий поведение растения при засыхании. Использованы коллекции для динамического управления листьями и лепестками, а инкапсуляция обеспечивает контроль доступа к внутренним данным класса.