Министерство образования Республики Беларусь

Учреждение образования

«Белорусский государственный университет

информатики и радиоэлектроники»

Кафедра инженерной психологии и эргономики

­­­­­

Современные языки программирования

Отчет по лабораторной работе №1

«Использование языка программирования Kotlin»

Выполнил: Позняк В. С.

Студент группы 310901

Преподаватель: Усенко Ф. В.

Минск 2024

Цель: Выполнить разработку приложения с использованием языка программирования Kotlin.

Задание.Вариант 19.

Классу Корабль добавить метод Вести пушечный огонь() и реализовать классы Зенитная пушка и Пушка главного калибра.

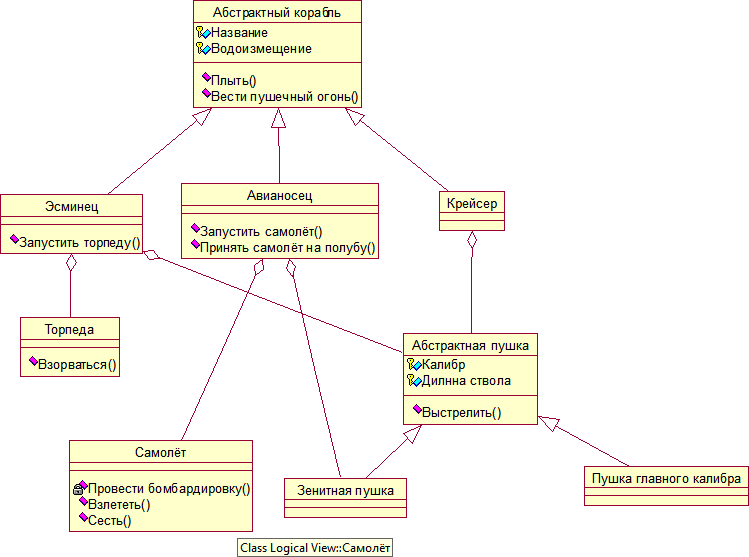


Рисунок 38 – Диаграмма классов

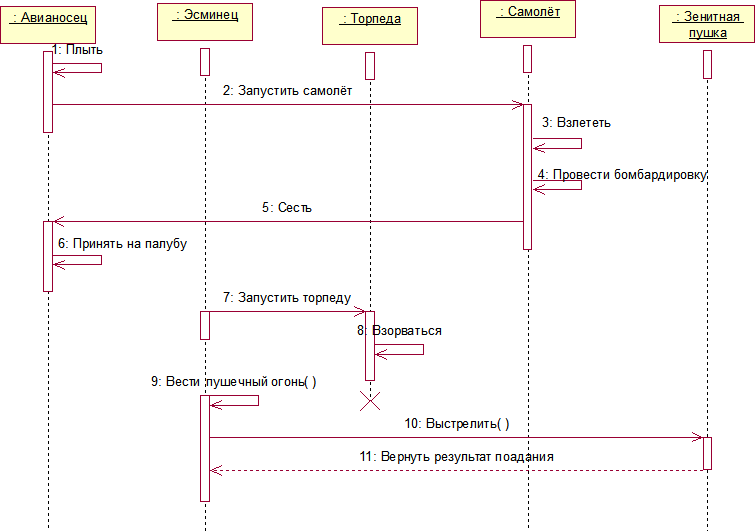


Рисунок 39 – Диаграмма последовательности

Листинг кода:

package AirUnits  
  
import WaterUnits.AircraftСarrier  
  
class Plane {  
  
 fun launch() {  
 *println*("Самолёт взлетел!")  
 Thread.sleep(3000)  
 }  
  
 fun land() {  
 *println*("Самолёт начал посадку!")  
 Thread.sleep(3000)  
 }  
  
 private fun bomb() {  
 *println*("Производится бомбардировка!")  
 Thread.sleep(3000)  
 }  
 public fun bomb\_public(){  
 bomb()  
  
 }  
  
}

package Guns  
  
import kotlin.random.Random  
  
class AntiAirGun(\_caliber: Double, \_barrelLength: Double) : Gun(\_caliber, \_barrelLength) {  
  
 override fun shoot() {  
 *println*("Ведётся огонь из зенитной пушки!")  
 Thread.sleep(1000)  
 if (Random.nextBoolean()) *println*("Попадание!") else *println*("Промах!")  
 }  
  
}

package Guns  
  
abstract class Gun(\_caliber: Double, \_barrelLength: Double) {  
 protected val caliber=\_caliber  
 protected val barrelLength=\_barrelLength  
 open fun shoot(){  
 *println*("Выстрел!")  
 }  
  
}

package Guns  
  
class MainGun(\_caliber: Double, \_barrelLength: Double) : Gun(\_caliber, \_barrelLength){  
  
 override fun shoot(){  
 *println*("Ведётся огонь из главной пушки!")  
 Thread.sleep(1000)  
 }  
  
}

package WaterUnits  
  
import AirUnits.Plane  
import Guns.AntiAirGun  
  
class AircraftСarrier(\_name: String, \_displacement: Double) : Ship(\_name, \_displacement) {  
  
 fun launchPlane() {  
 *println*("Самолёт начал взлёт!")  
 Thread.sleep(1000)  
 Plane().launch()  
 }  
  
 fun landPlane() {  
 *println*("Авианосец принял на палубу!")  
 Thread.sleep(1000)  
 }  
  
 fun antiAirShoot() {  
 AntiAirGun(15.0,32.5).shoot()  
 }  
}

package WaterUnits  
  
class cruiser : Ship("ijrr", 65.7) {  
  
}

package WaterUnits  
  
import Guns.AntiAirGun  
import Guns.MainGun  
  
class Destroyer(\_name: String, \_displacement: Double) : Ship(\_name, \_displacement){  
  
 private class Torpedo{  
  
 fun explode() {  
 *println*("Взрыв!")  
 }  
 }  
  
 fun shootTorpedo() {  
 *println*("Эсминец стреляет торпедой")  
 Thread.sleep(1000)  
 Torpedo().explode()  
 }  
  
 override fun shoot() {  
 *println*("Эсминец стреляет")  
 MainGun(40.0,75.0).shoot()  
 }  
  
 fun antiAirShoot() {  
 AntiAirGun(15.0,32.5).shoot()  
 }  
}

package WaterUnits  
  
import Guns.MainGun  
  
abstract class Ship (\_name: String, \_displacement: Double) {  
  
 protected val name = \_name  
 protected val displacement=\_displacement  
 open fun swim() {  
 *println*("Корабль $name плывёт")  
 Thread.sleep(1000)  
 }  
  
 open fun shoot() {  
 *println*("Корабль стреляет")  
 MainGun(40.0,75.0).shoot()  
 }  
}

import AirUnits.Plane  
import WaterUnits.AircraftСarrier  
import WaterUnits.Destroyer  
  
fun main() {  
  
 val aircraftСarrier = AircraftСarrier("Auram", 150.0)  
 aircraftСarrier.swim()  
 aircraftСarrier.launchPlane()  
 Plane().bomb\_public()  
 Plane().land()  
 aircraftСarrier.landPlane()  
 val destroyer = Destroyer("Stinger", 75.5)  
 destroyer.shootTorpedo()  
 destroyer.shoot()  
 destroyer.antiAirShoot()  
}

На рисунке 1 результат выполнения программы.

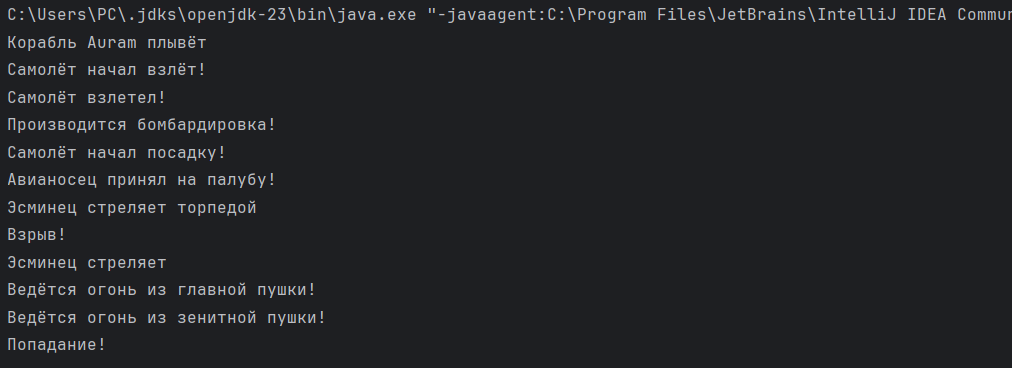


Рисунок 1 – Результат выполнения программы

Вывод: в ходе выполнения лабораторной работы номер 1 по языку программирования kotlin мы познакомились с основами языка и научились на практике применять основные его конструкции. Было потрачено время на изучение UML диаграмм для корректного понимая задания лабораторной рабыты. При выполнении дополнительного задания познакомились с библиотекой Junit5 для написания тестов. При выполнении возникли трудности с понииманием работы конструкторов в Kotlin, в остальном работа выполнялась гладко. Затрачено часов – 6.