Министерство образования

Учреждение образования

«Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерного проектирования

Кафедра инженерной психологии и эргономики

Отчет по лабораторной работе №1

«Использование языка программирования Kotlin»

Выполнил: Крукович А.В.

Студент группы 310902

Проверил: Усенко Ф.В.

Минск 2024

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

Задание. Добавить классы Ангина и Грипп, наследующие от класса Абстрактная болезнь. Пациенту добавить горло и лёгкие. Также добавить метод принять лекарство () разрушающий Болезнь. Добавить класс Лекарство.

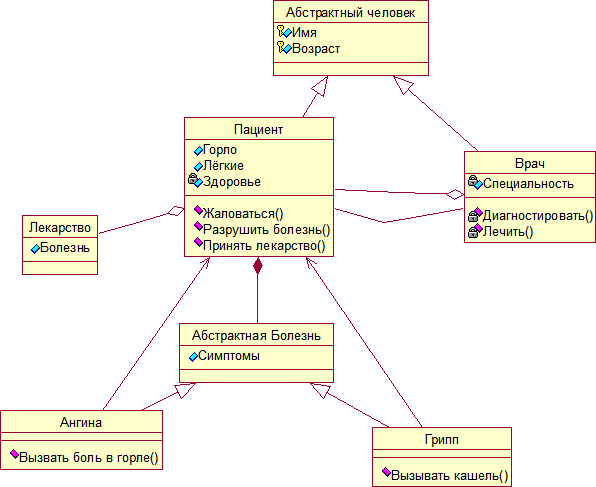


Диаграмма классов

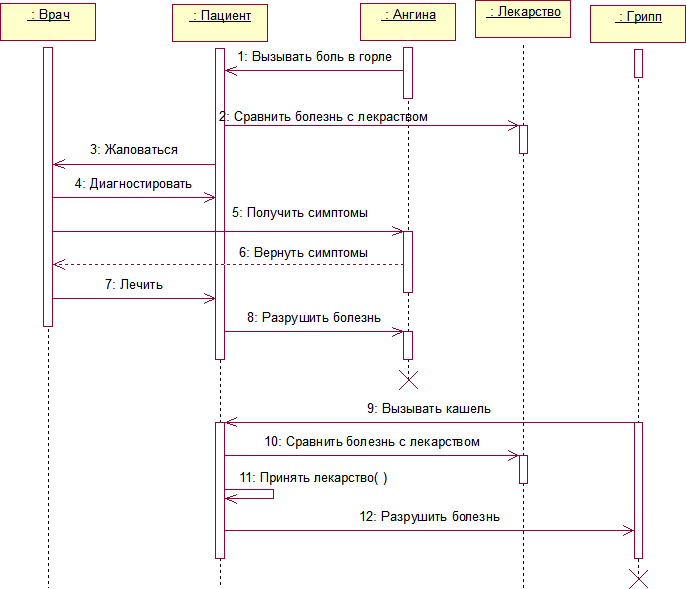


Диаграмма последовательности

**Код на языке Kotlin:**

**Main.kt**

***import patient.\*  
import pills.\*  
import desease.\*  
import doctor.\*  
  
//Специальность врачу. Функция лечить - приватная. Здоровье пациенту, приватное.  
fun main() {  
  
 val patient = Patient()  
 val doctor = Doctor()  
 val angina = Angina()  
 val flu = Flu()  
 val pills = Pills()  
 val options = listOf(0, 1, 2, 3)  
 val randomOption = options.random()  
 val randomDesease = when(randomOption){  
 1 -> angina.infectPatient(patient)  
 2 -> flu.infectPatient(patient)  
 3 -> {  
 flu.infectPatient(patient)  
 angina.infectPatient(patient)  
 }  
  
 else -> null  
 }  
  
 patient.Complain()  
 patient.infect()  
  
 doctor.checkPatient(patient)  
  
 patient.takePills(pills)  
  
 doctor.cure(patient)  
  
}***

**Doctor.kt**

***package doctor  
import patient.Patient  
  
class Doctor {  
 private val speciality: String = "pediatrician"  
 private fun diagnoseDisease(patient: Patient): String {  
 return when {  
 patient.throat == "Sore" && patient.lungs == "Healthy" -> "Angina"  
 patient.lungs == "Inflamed" && patient.throat == "Healthy" -> "Flu"  
 patient.lungs == "Inflamed" && patient.throat == "Sore" -> "Complex case of both Angina and Flu"  
 else -> "Healthy"  
 }  
 }  
  
 fun checkPatient(patient: Patient) {  
 println("Checking the patient...")  
 println("Throat status: ${patient.throat}")  
 println("Lungs status: ${patient.lungs}")  
 val diagnosis = diagnoseDisease(patient)  
 println("Diagnosis: $diagnosis")  
  
 if(patient.throat == "Healthy" && patient.lungs == "Healthy") {  
 println("The patient is healthy.")  
 System.exit(0)  
 }  
  
 }  
  
 private fun curePatient(patient: Patient) {  
 if (patient.throat == "Sore") {  
 println("Curing the patient from Angina...")  
 patient.throat = "Healthy"  
  
 val diagnosis = diagnoseDisease(patient)  
 println("Diagnosis: $diagnosis")  
 }  
  
 }  
 fun cure(patient: Patient) {  
 curePatient(patient)  
 }  
}***

**Desease.kt**

***package desease  
import patient.Patient  
  
abstract class Desease{  
 open var sympthoms: String = "Undifined"  
  
  
}  
  
class Angina: Desease(){  
 override var sympthoms: String = "Sore throat"  
 fun infectPatient(patient: Patient) {  
  
 patient.throat = "Sore"  
  
  
 }  
  
  
  
}  
class Flu: Desease(){  
 override var sympthoms: String = "Cough"  
 fun infectPatient(patient: Patient) {  
 patient.lungs = "Inflamed"  
 }  
  
  
}***

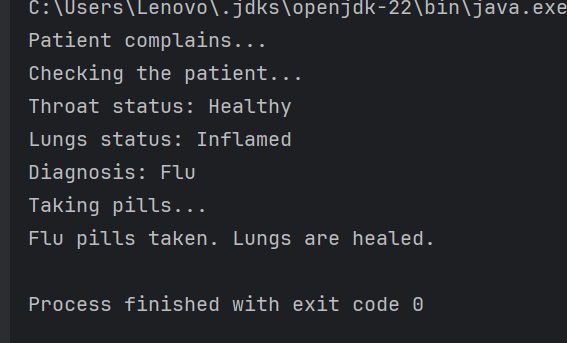
**Patient.kt**

***package patient  
import pills.\*  
  
interface Patient1 {  
  
  
 val throat: String  
 val lungs: String  
 fun Complain()  
 fun takePills(pills: Pills1)  
  
}  
 abstract class Person{  
 protected var name: String = "Undefined"  
 protected var age: Int = 0  
  
 }  
 class Patient: Patient1 {  
  
 override var throat: String = "Healthy"  
 override var lungs: String = "Healthy"  
 private var health: String = "Healthy"  
  
 override fun Complain(){  
 println("Patient complains...")  
 }  
 fun infect(){  
 if(throat != "Healthy" || lungs !="Healthy"){  
 health = "Infected"  
 }  
 }  
  
  
 override fun takePills(pills: Pills1) {  
 if(lungs == "Inflamed") {  
 println("Taking pills...")  
 when (pills.deseaseType) {  
  
  
 "Flu" -> {  
 lungs = "Healthy"  
 println("Flu pills taken. Lungs are healed.")  
 }  
  
  
 else -> println("No specific pills to take.")  
 }  
 }  
 }  
 }***

**Pills.kt**

***package pills  
  
  
interface Pills1 {  
 val deseaseType: Any  
  
  
}  
class Pills: Pills1 {  
 override var deseaseType: String = "Flu"  
}***

**Скриншот работающей программы:**

****

**Рисунок 1 - Результат выполнения программы**

**Вывод: В этой лабораторной работе мы получили базовые знания о языке программирования Kotlin и написали программу на этом языке, разбив ее функционал на несколько пакетов с учетом логики и сделав валидацию вводимых значений.**