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: Сверчков Е.А.

Студент группы 310902

Проверил: Усенко Ф.В.

Минск 2024

Индивидуальное задание №26

Добавить классы, наследующие от класса одежда, а именно Штаны и Рубашка. Добавить классу Портной фабричные методы, создающие эти классы. Добавить классу Ткань поле количество и организовать его работу.

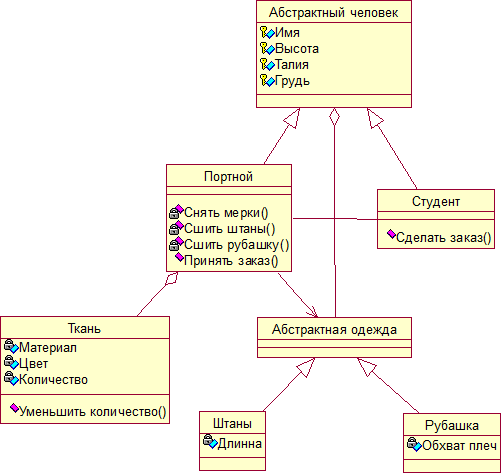


Рисунок 52 – Диаграмма классов

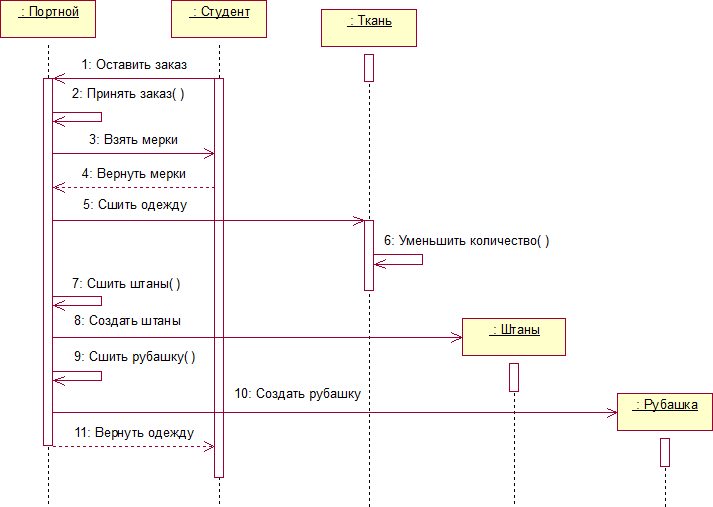


Рисунок 53 – Диаграмма последовательности

Листинг кода:

Содержимое файла Fabric.kt:

package org.example  
class Fabric(  
 val material: String = "",  
 val colour: String = "",  
 var amount: Double = 0.0  
) {  
 override fun toString() =  
 "Материал - ${material.toString()}\tЦвет - ${colour.toString()}\tКол-во (m^2) - ${amount.toString()}\n"  
  
 fun reduceFabric(amount: Double) {  
 this.amount -= amount  
 }  
}

Содержимое файла Clothes.kt:

package org.example  
  
abstract class Clothes {}  
  
class Pants(  
 val length: Double,  
 val waist: Double,  
 val fabric: Fabric  
) : Clothes() {  
 override fun toString() = "\nШтаны\nДлинна: ${length.toString()}\nТалия: ${waist.toString()}\nТкань: ${fabric.toString()}"  
}  
  
class Shirt(  
 val shoulderGirth: Double,  
 val chest: Double,  
 val waist: Double,  
 val fabric: Fabric  
) : Clothes() {  
 override fun toString() =  
 "\nРубашка\nОбхват плеч: ${shoulderGirth.toString()}\nТалия: ${waist.toString()}\nОбхват груди:${chest.toString()}\nТкань: ${fabric.toString()}"  
}

Содержимое файла Order.kt:

package org.example  
  
class Order(val fabric: Fabric, val amount: Double, val parameter: Double) {  
  
}

Содержимое файла Person.kt:

package org.example  
  
abstract class Person(  
 var Name: String = "Undefined",  
 var Height: Double = 0.0,  
 var Chest: Double = 0.0,  
 var Waist: Double = 0.0  
) {  
}  
  
class Student(  
 Name: String = "Undefined",  
 Height: Double = 0.0,  
 Chest: Double = 0.0,  
 Waist: Double = 0.0  
) : Person(Name, Height, Chest, Waist) {  
 var clothes: MutableList<Clothes> = mutableListOf()

fun makeOrder() = this

fun orderPants(read: () -> String?): Double {  
 println("Введите длину штанов сантиметрах")  
 val length = read()?.toDouble() ?: 0.0  
 return length  
 }  
 fun orderShirt(read: () -> String?): Double {  
 println("Введите обхват плеч в сантиметрах")  
 val shoulderGirth = read()?.toDouble() ?: 0.0  
 return shoulderGirth  
 }  
  
 fun reseiveClothes(clothes: Clothes) {  
 this.clothes.add(clothes)  
 println("Одежда получена")  
 }  
  
 override fun toString() =  
 "Имя: $Name\nРост: $Height\nТалия $Waist\nОбхват груди $Chest\nНабор одежды ${clothes.toString()}"  
}  
  
class Tailor : Person() {  
 lateinit var client: Student  
 lateinit var order: Order  
 var measurements: Double = 0.0

var fabrics: MutableSet<Fabric> = mutableSetOf(  
 Fabric("Хлопок", "Серый", 15.0),  
 Fabric("Хлопок", "Белый", 25.0),  
 Fabric("Джинса", "Джинса", 10.0),  
 Fabric("Лён", "Жёлтый", 45.0),  
 Fabric("Кожа", "Чёрный", 10.0)  
 )  
  
 fun getOrger(student: Student, read: () -> String?): Int {  
 client = student  
 println("Что шьём?\n1 - Штаны\n2 - Рубашка")  
 var i = 0  
 do {  
 try {  
 i = read()?.toInt() ?: 0  
 } catch (\_: Exception) {  
 println("Неверный ввод")  
 }  
 } while (i != 1 && i != 2)  
  
 if (i == 1) takePantsMeasurements(read)  
 else takeShirtMeasurements(read)  
  
 return i  
 }  
  
 private fun takeShirtMeasurements(read: () -> String?) {  
 this.measurements = client.orderShirt(read)  
 }  
  
 private fun takePantsMeasurements(read: () -> String?) {  
 this.measurements = client.orderPants(read)  
 }  
  
 fun orderPants(read: () -> String?): Boolean {  
 val length = measurements  
 val amountOfFabric = length \* client.Waist / 10000  
  
 val potentialFabrics = fabrics.filter { amountOfFabric <= it.amount }  
 if (potentialFabrics.isEmpty()) {  
 println("Недостаточно материала")  
 return false  
 }  
 println("Выберете материал из списка")  
 for (item in potentialFabrics) {  
 print(item)  
 }  
 var i: Int = 0  
 do {  
 try {  
 i = read()?.toInt() ?: 0  
 } catch (e: Exception) {  
 println("Неверный ввод")  
 continue  
 }  
 } while (i <= 0 || i > potentialFabrics.size)  
 this.order = Order(potentialFabrics.get(i - 1), amountOfFabric, length)  
 potentialFabrics.get(i - 1).reduceFabric(amountOfFabric)  
 return true  
 }  
  
 fun orderShirt(read: () -> String?): Boolean {  
 val shoulderGirth = measurements  
 val amountOfFabric =  
 ((client.Chest + client.Waist) \* client.Height / 4 + shoulderGirth \* client.Height / 10) / 10000  
 val potentialFabrics = fabrics.filter { amountOfFabric <= it.amount }  
  
 if (potentialFabrics.isEmpty()) {  
 println("Недостаточно материала")  
 return false  
 }  
  
 println("Выберете материал из списка")  
 for (item in potentialFabrics) {  
 print(item)  
 }  
 var i: Int = 0  
 do {  
 try {  
 i = read()?.toInt() ?: 0  
 } catch (e: Exception) {  
 println("Неверный ввод")  
 continue  
 }  
 } while (i <= 0 || i > potentialFabrics.size)  
 this.order = Order(potentialFabrics.get(i - 1), amountOfFabric, shoulderGirth)  
 potentialFabrics.get(i - 1).reduceFabric(amountOfFabric)  
 return true  
 }  
  
 fun suePants() = Pants(order.parameter, client.Waist, order.fabric)  
 fun sueShirt() = Shirt(order.parameter, client.Chest, client.Waist, order.fabric)  
}

Содержимое файла Main.kt:

package org.example  
fun main() {  
 app(:: readln)  
}  
  
fun app(read: () -> String?) {  
 val student = Student("Evgeniy", 179.0, 96.0, 800.0)  
 val tailor = Tailor()  
  
 println(student.toString())  
  
 while (true) {  
 println("Выберете действие\n1 - Просмотр данных\n2 - Заказать одежду\n0 - Выход")  
 var i = -1  
 do {  
  
 try {  
 i = read()?.toInt() ?: 0  
 } catch (e: NumberFormatException) {  
 println("Неверный ввод")  
 }  
 } while (i < 0 || i > 2)  
  
 when (i) {  
 1 -> println(student.toString())  
 2 -> {  
 when (tailor.getOrger(student.makeOrder(), read)) {  
 1 -> {  
 if (tailor.orderPants(read)) {  
 student.reseiveClothes(tailor.suePants())  
 }  
 }  
  
 2 -> {  
 if (tailor.orderShirt(read)) {  
 student.reseiveClothes(tailor.sueShirt())  
 }  
 }  
 }  
 }  
 0 -> return  
 }  
 }  
}

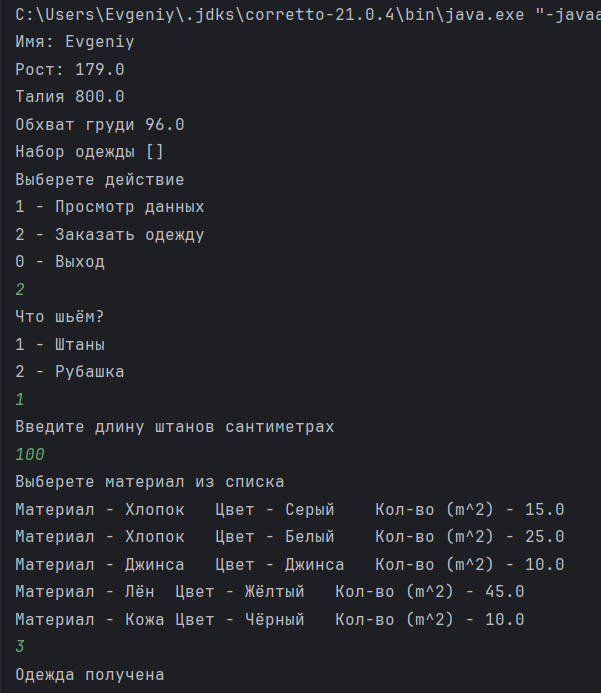
Скриншоты работающей программы:

Рисунок 1 – Добавление штанов

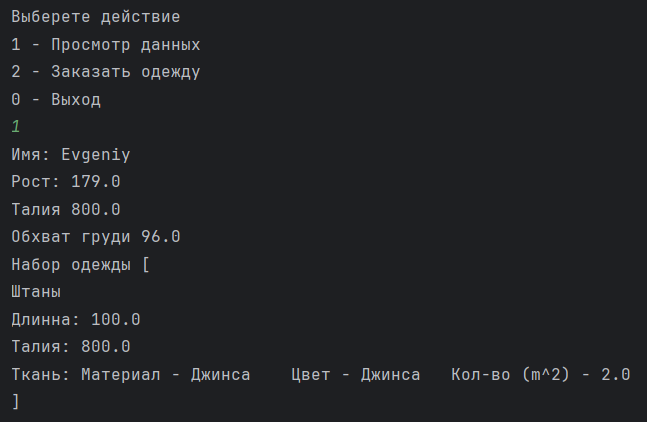


Рисунок 2 – Просмотр данных после добавления штанов

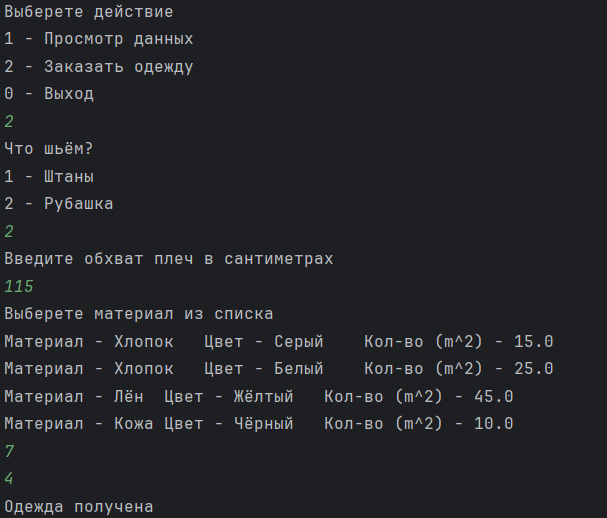


Рисунок 3 – Пошив рубашки

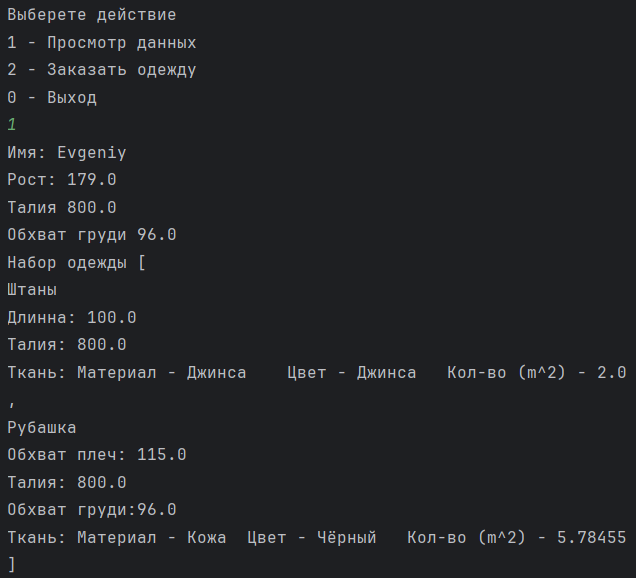


Рисунок 4 – Просмотр данных после добавления рубашки к набору одежды

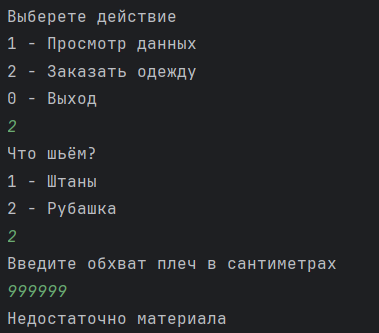


Рисунок 5 – Случай, при котором недостаточно материала для пошива

Вывод:

В ходе выполнения лабораторной были изучены базовые элементы языка, для выполнения поставленной задачи.