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: Судаков И. В.

Студент группы 310901

Преподаватель: Усенко Ф. В.

Минск 2024

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

**Вариант 25**: Добавить класс Пакет и организовать возможность помещать туда товар. Добавить классу Тележка метод Выдать товар(). Добавить класс Молоко.

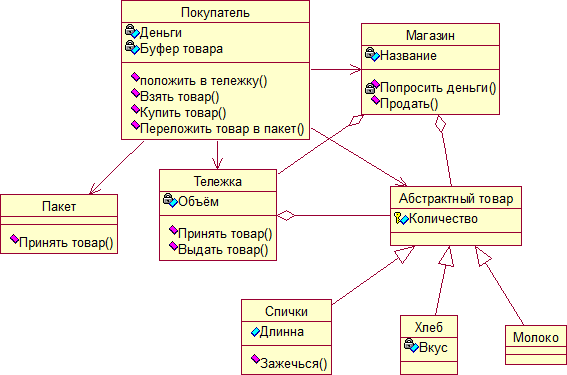


Рисунок 1 – Диаграмма классов

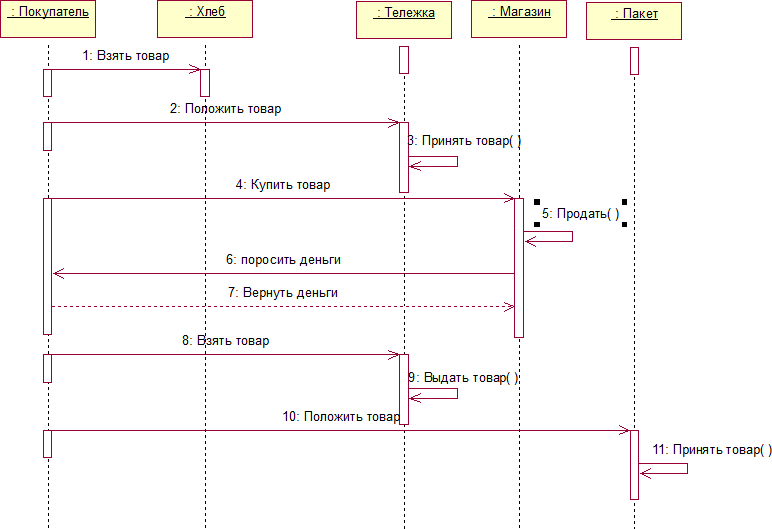


Рисунок 2 – Диаграмма последовательности

**Текст программы:**

//Main.kt

fun main() {

val buyerMoney = Money(20.0)

val buyer = Buyer(buyerMoney)

val store = Store("Магазин 1")

val matches = Matches(5, 10.0)

val whiteBread = Bread("Белый")

val ryeBread = Bread("Ржаной")

val milk = Milk(3)

while (true) {

println("Вас приветствует магазин!")

println("1. Взять товар")

println("2. Положить в тележку")

println("3. Купить товар")

println("4. Взять товар из тележки")

println("5. Положить товар в пакет")

println("6. Выход")

val choice = readLine()?.toIntOrNull()

when (choice) {

1 -> {

println("Какой товар вы хотите взять?")

println("1. Спички")

println("2. Хлеб")

println("3. Молоко")

val productChoice = readLine()?.toIntOrNull()

when (productChoice) {

1 -> buyer.takeProduct(matches)

2 -> {

println("Какой хлеб вы хотите взять?")

println("1. Белый хлеб")

println("2. Ржаной хлеб")

val breadChoice = readLine()?.toIntOrNull()

when (breadChoice) {

1 -> buyer.takeProduct(whiteBread)

2 -> buyer.takeProduct(ryeBread)

else -> println("Некорректный выбор хлеба.")

}

}

3 -> buyer.takeProduct(milk)

else -> println("Некорректный выбор товара.")

}

}

2 -> {

buyer.putInCart()

println("Товары добавлены в тележку.")

}

3 -> {

buyer.buyProduct(store)

println("Товар куплен.")

}

4 -> {

val productFromCart = buyer.takeFromCart()

if (productFromCart != null) {

buyer.takeProduct(productFromCart)

println("${productFromCart.name} взят из тележки.")

} else {

println("Тележка пуста.")

}

}

5 -> {

val productFromBufferToBag = buyer.takeFromBuffer()

if (productFromBufferToBag != null) {

buyer.putInBag(productFromBufferToBag)

println("${productFromBufferToBag.name} взят из буфера.")

} else {

println("Буфер пуст.")

}

}

6 -> {

println("Выход из программы.")

return

}

else -> println("Некорректный выбор. Попробуйте снова.")

}

}

}

// Bag.kt

class Bag {

private val products = mutableListOf<Product>()

fun acceptProduct(product: Product) {

products.add(product)

println("${product.showInfo()} добавлено в пакет.")

}

fun releaseProduct(): Product? {

return if (products.isNotEmpty()) {

val product = products.removeAt(products.size - 1)

println("${product.showInfo()} выдан из пакета.")

product

} else {

println("Пакет пуст.")

null

}

}

}

// Bread.kt

class Bread(private val flavor: String, quantity: Int = 1) : Product(quantity) {

override val name: String

get() = "Хлеб: вкус = $flavor"

override fun showInfo(): String {

return "Хлеб: вкус = $flavor, количество = $quantity"

}

}

// Buyer.kt

class Buyer( private val money: Money) {

private val productBuffer = ProductBuffer()

private val cart = Cart(10)

private val bag = Bag()

fun takeProduct(product: Product) {

productBuffer.addProduct(product)

}

fun putInCart() {

val productsToAdd = productBuffer.getProducts()

productsToAdd.forEach { cart.acceptProduct(it) }

productBuffer.clearBuffer()

}

fun buyProduct(store: Store) {

val productsToBuy = cart.getProducts()

productsToBuy.forEach { product ->

store.sellToBuyer(this, product)

}

}

fun takeFromCart(): Product? {

return cart.releaseProduct()

}

fun putInBag(product: Product) {

bag.acceptProduct(product)

}

fun hasEnoughMoney(amount: Double): Boolean {

return money.hasEnough(amount)

}

fun spendMoney(amount: Double) {

money.spend(amount)

}

fun takeFromBuffer(): Product? {

return productBuffer.releaseProduct()

}

}

// Cart.kt

class Cart(private val capacity: Int) {

private val productsInCart = mutableListOf<Product>()

fun acceptProduct(product: Product) {

if (productsInCart.size < capacity) {

productsInCart.add(product)

println("${product.showInfo()} добавлено в тележку.")

} else {

println("Тележка полна!")

}

}

fun releaseProduct(): Product? {

return if (productsInCart.isNotEmpty()) {

val product = productsInCart.removeAt(productsInCart.size - 1)

println("${product.showInfo()} выдан из тележки.")

product

} else {

println("Тележка пуста.")

null

}

}

fun getProducts(): List<Product> {

return productsInCart.toList()

}

}

// Matches.kt

class Matches(quantity: Int, val length: Double) : Product(quantity) {

fun ignite() {

println("Спички зажжены!")

}

override val name: String

get() = "Спички (длина: $length см, количество: $quantity)"

override fun showInfo(): String {

return "Спички: длина = $length см, количество = $quantity"

}

}

// Milk.kt

class Milk(quantity: Int) : Product(quantity) {

override val name: String

get() = "Молоко (количество: $quantity)"

override fun showInfo(): String {

return "Молоко: количество = $quantity"

}

}

// Money.kt

class Money(private var amount: Double) {

fun spend(amount: Double) {

this.amount -= amount

println("Потрачено: $amount. Осталось: ${this.amount}")

}

fun hasEnough(amount: Double): Boolean {

return this.amount >= amount

}

}

// Product.kt

abstract class Product(val quantity: Int) {

abstract val name: String

abstract fun showInfo(): String

}

// ProductBuffer.kt

class ProductBuffer {

private val products = mutableListOf<Product>()

fun addProduct(product: Product) {

products.add(product)

println("${product.showInfo()} добавлено в буфер.")

}

fun clearBuffer() {

products.clear()

println("Буфер очищен.")

}

fun getProducts(): List<Product> {

return products.toList()

}

fun releaseProduct(): Product? {

return if (products.isNotEmpty()) {

val product = products.removeAt(products.size - 1)

println("${product.showInfo()} выдан из буфера.")

product

} else {

println("Пакет пуст.")

null

}

}

}

// Store.kt

class Store(private val name: String) {

fun sell(product: Product): Double {

val price = when (product) {

is Matches -> 1.0

is Bread -> 2.0

is Milk -> 1.5

else -> 0.0

}

println("Магазин $name продает: ${product.showInfo()} за $price.")

return price

}

fun sellToBuyer(buyer: Buyer, product: Product) {

val price = sell(product)

askForMoney(buyer, price)

}

private fun askForMoney(buyer: Buyer, amount: Double) {

println("Магазин просит $amount у покупателя.")

if (buyer.hasEnoughMoney(amount)) {

buyer.spendMoney(amount)

println("Покупка успешна!")

} else {

println("Недостаточно денег у покупателя.")

}

}

}

Результат работы программы представлен на рисунках 3,4,5.

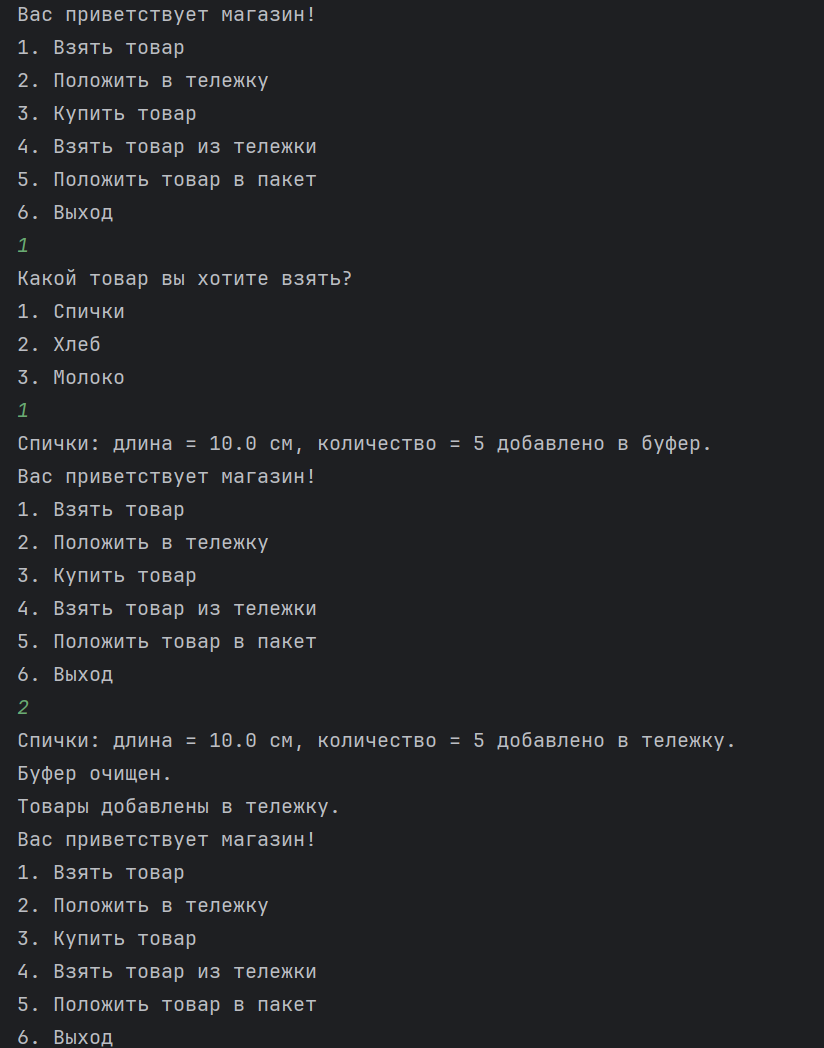


Рисунок 3 – Результат работы программы.

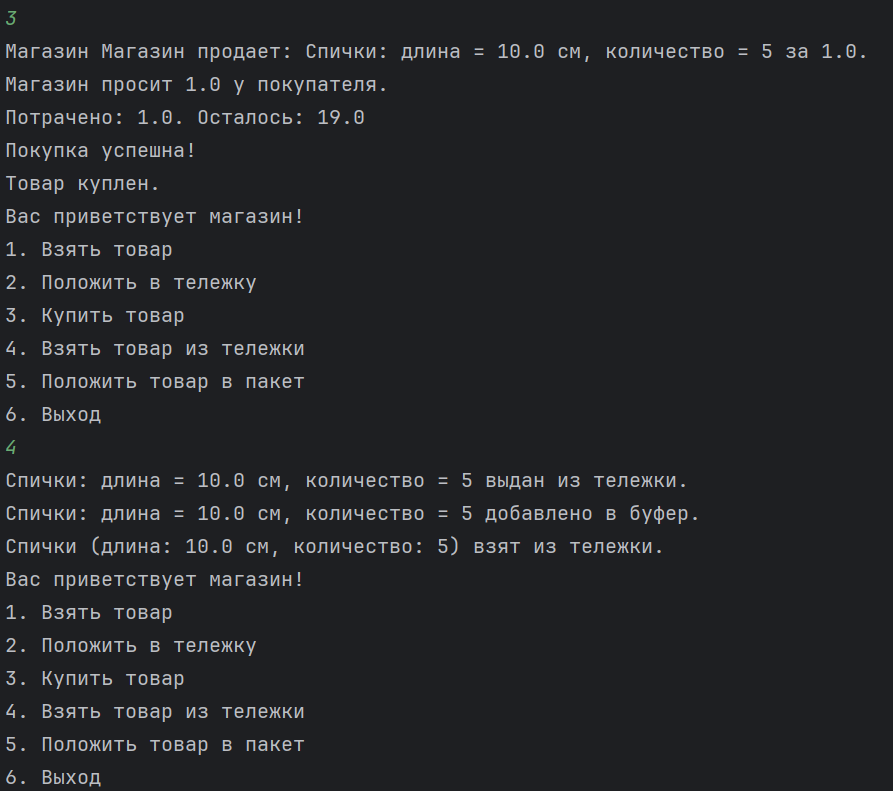


Рисунок 4 – Результат работы программы.

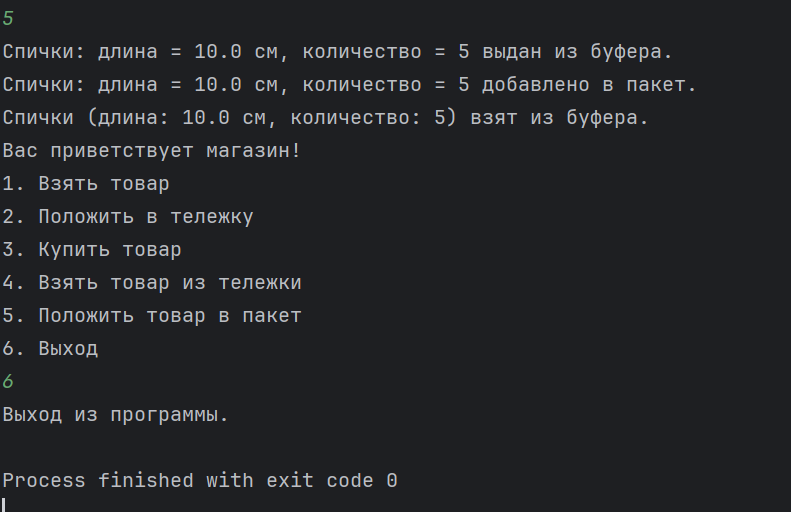


Рисунок 5 – Результат работы программы.

**Выводы**: Изучены основы языка Kotlin и выполнена разработка приложения на данном языке.