Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Кафедра инженерной психологии и эргономики

Использование языка программирования Kotlin

Проверил:
Усенко Ф.В.

Выполнил:
Бородин А.Н.
гр. 310901

Минск 2024

Цель: Изучить синтаксис и основную логику языка kotlin.

Задание 3

Реализовать программу, описанную диаграммами. Дополнительно, написать тесты.

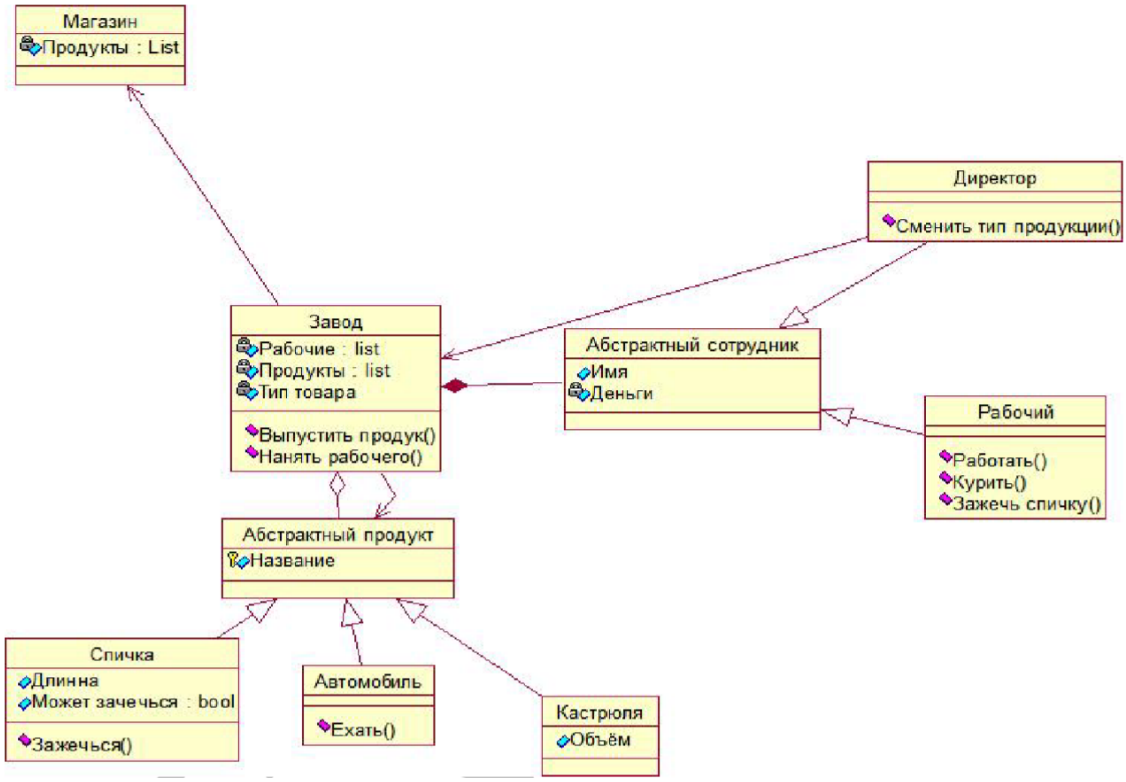

Рисунок 1 – Диаграмма классов

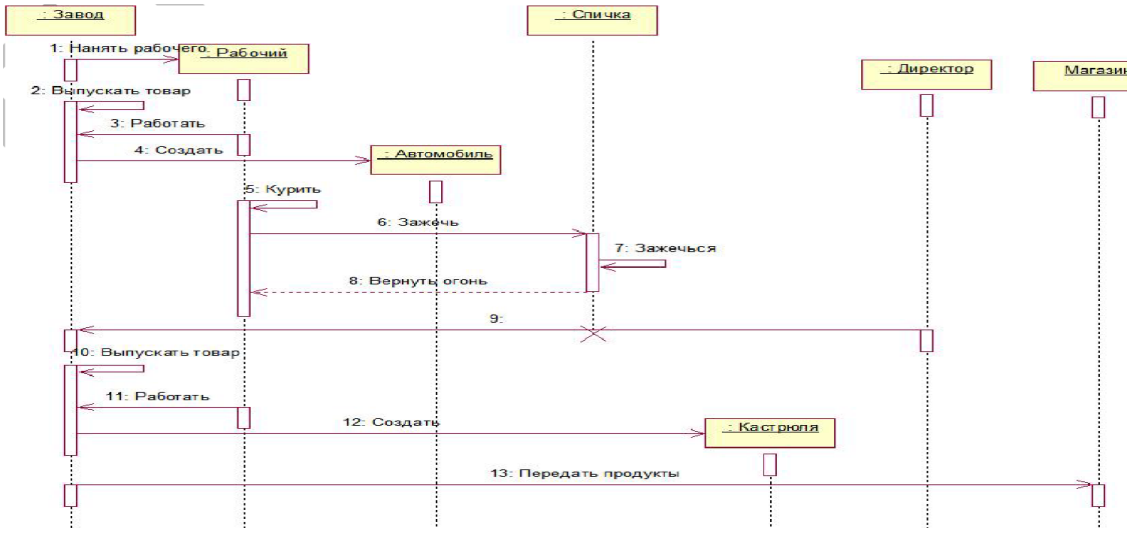

Рисунок 2 – Диаграмма последовательности

```
package io.github.nadevko.bsuir.MPL1

import kotlin.random.Random

abstract class Employee(val name: String) {
    private var money: Int = 0

    fun setMoney(money: Int): Int {
        val oldMoney = this.money
        this.money = money
        return oldMoney
    }

    fun getMoney() = money
}

class Worker(name: String) : Employee(name) {

    fun work(product: String, money: Int) {
        print("$name працуе над \"$product\"")
        setMoney(getMoney() + money)
        println(" і зарабляе $money (рахунак: ${getMoney()})")
    }

    fun smoke(match: Match) {
        println("\n$name ідзе паліць...")
        if (light(match)) {
            println("$name запальвае цыгарку...")
            println("$name паліць")
        } else {
            println("$name думае, ці не бросіць яму...")
        }
    }
```

```
    fun light(match: Match): Boolean {
        println("$name спрабуе запаліць запалку")
        return match.light()
    }
}

class Director(name: String) : Employee(name) {

    fun setProductType(factory: Factory, type: ProductType) {
        factory.setProductType(type)
        println(
                "$name змяніў тып вырабу на ${when (type) {
            ProductType.CAR -> "машыны"
            ProductType.POT -> "рондалі"
            ProductType.MATCH -> "запалкі"
        }}"
        )
    }
}

class Factory(private var productType: ProductType =
ProductType.CAR) {
    private val workers: MutableList<Worker> = mutableListOf()
    private val products: MutableList<Product> = mutableListOf()

    fun produce(name: String): Product {
        println("\nФабрыка пачала выпрабляць \"$name\"")
        val product =
                when (productType) {
                    ProductType.CAR -> Car(name)
                    ProductType.POT -> Pot(name, 3u)
                    ProductType.MATCH -> Match(name)
                }
        workers.forEach {
```

```
            it.work(
                    name,
                    when (productType) {
                        ProductType.CAR -> 2_000
                        ProductType.POT -> 500
                        ProductType.MATCH -> 5
                    }
            )
        }
        println("Фабрыка зрабіла \"$name\"")
        products.add(product)
        return product
    }

    fun hire(worker: Worker) = workers.add(worker)

    fun setProductType(type: ProductType) {
        productType = type
    }

    fun transfer(store: Store): MutableList<Product> {
        println("Фабрыка перадае товары на склад")
        store.transfer(this.products.toMutableList())
        this.products.clear()
        return products
    }
}

class Store {
    private val products: MutableList<Product> = mutableListOf()

    fun transfer(products: MutableList<Product>) {
        this.products.addAll(products)
        list()
```

```
    }

    fun list() {
        println("\nТовары на складзе магазіна:")
        products.forEach { println(it.getNaming()) }
    }
}

abstract class Product(protected val name: String) {
    fun getNaming() = name
}

enum class ProductType {
    CAR,
    POT,
    MATCH
}

class Car(name: String) : Product(name) {

    fun drive() = println("Машына \"$name\" ў руху")
}

class Pot(name: String, volume: UInt) : Product(name) {
    val volume = volume
}

class Match(name: String) : Product(name) {
    var isLightable = true
    var length = 5

    fun light(): Boolean {
        length -= 1
        if (length == 0) {
```

```
            println("$name занадта кароткая")
            isLightable = false
        }
        if (!isLightable) {
            println("$name немагчыма запаліць")
            return false
        }
        if (Random.nextDouble() < 0.3) {
            println("$name не запальваецца")
            return false
        }
        println("$name загараецца...")
        isLightable = false
        return true
    }
}

fun main() {
    val match = Match("Запалачка")
    println("Будуецца фабрыка...")
    val factory = Factory()
    println("Пабудаваны ўсефабрычны мегакамбінат")
    val worker =
            Worker(
                    listOf(
                                    "Іваныч",
                                    "Пятровіч",
                                    "Мікалаіч",
                                    "Лёха",
                                    "Васёк",
                                    "Пятрок",
                                    "Мішаня",
                            )
                            .random()
```

```
            )
    val director =
            Director(
                    listOf(
                                    "Іван Іванавіч",
                                    "Пётр Пятровіч",
                                    "Мікалай Мікалаевіч",
                                    "Аляксей Аляксееў",
                                    "Васіль Васільеў",
                                    "Пётр Пятровіч",
                                    "Міхайла Міхайлавіч",
                            )
                            .random()
            )
    val store = Store()
    factory.hire(worker)
    factory.produce("Рэнова логан")
    worker.smoke(match)
    director.setProductType(factory, ProductType.POT)
    factory.produce("Рондаль на 3 літры")
    factory.transfer(store)
}
```

Рисунок 3 – Main.kt

```
package io.github.nadevko.bsuir.MPL1

import org.junit.jupiter.api.Assertions.*
import org.junit.jupiter.api.Test

class MainTest {

    @Test
    fun `test factory hires workers correctly`() {
```

```
        val factory = Factory()
        val worker = Worker("Test Worker")
        assertTrue(factory.hire(worker), "Worker should be hired
successfully.")
    }

    @Test
    fun `test factory produces products correctly`() {
        val factory = Factory(ProductType.CAR)
        val product = factory.produce("Test Car")
        assertNotNull(product, "Factory should produce a
product.")
        assertTrue(product is Car, "Product should be of type
Car.")
    }

    @Test
    fun `test factory transfers products to store`() {
        val factory = Factory()
        val store = Store()
        factory.produce("Test Product 1")
        factory.produce("Test Product 2")
        val transferredProducts = factory.transfer(store)
        assertEquals(0, transferredProducts.size, "Factory's
product list should be empty after transfer.")
    }

    @Test
    fun `test store receives transferred products`() {
        val factory = Factory()
        val store = Store()
        factory.produce("Test Product 1")
        factory.produce("Test Product 2")
        factory.transfer(store)
```

```
        val output = captureOutput { store.list() }
        assertTrue(output.contains("Test Product 1"), "Store
should contain 'Test Product 1'.")
        assertTrue(output.contains("Test Product 2"), "Store
should contain 'Test Product 2'.")
    }

    private fun captureOutput(block: () -> Unit): String {
        val originalOut = System.out
        val outputStream = java.io.ByteArrayOutputStream()
        System.setOut(java.io.PrintStream(outputStream))
        try {
            block()
        } finally {
            System.setOut(originalOut)
        }
        return outputStream.toString()
    }
}
```

Рисунок 4 – MainTest.kt

```
Будуецца фабрыка...
Пабудаваны ўсефабрычны мегакамбінат

Фабрыка пачала вырабляць "Рэнова логан"
Пятрок працуе над "Рэнова логан" і зарабляе 2000 (рахунак: 2000)
Фабрыка зрабіла "Рэнова логан"

Пятрок ідзе паліць...
Пятрок спрабуе запаліць запалку
Запалачка загараецца...
Пятрок запальвае цыгарку...
Пятрок паліць
```

```
Мікалай Мікалаевіч змяніў тып вырабу на рондалі

Фабрыка пачала вырабляць "Рондаль на 3 літры"
Пятрок працуе над "Рондаль на 3 літры" і зарабляе 500 (рахунак:
2500)
Фабрыка зрабіла "Рондаль на 3 літры"
Фабрыка перадае товары на склад

Товары на складзе магазіна:
Рэнова логан
Рондаль на 3 літры
```

Рисунок 5 – Вывод программы

Вывод: освоен базис языка программирования kotlin.