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: Майданюк А.С.

Студент группы 310901

Преподаватель: Усенко Ф. В.

Минск 2024

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

**Вариант 15**: Добавить класс Абстрактный файл, от которого наследует класс Программа. Добавить класс Видеофайл, и организовать его взаимодействие с классом Плеер и классом Браузер.

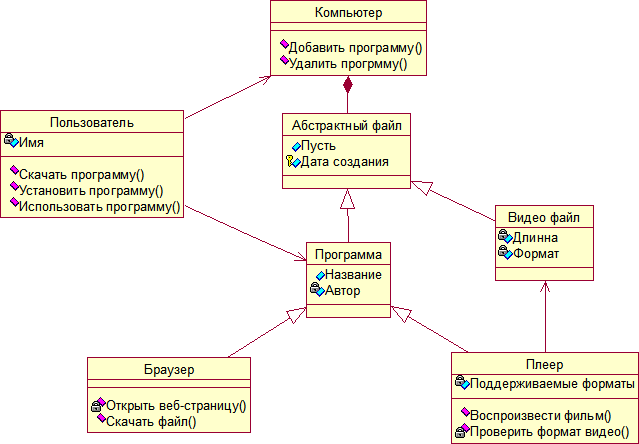


Рисунок 1 – Диаграмма классов

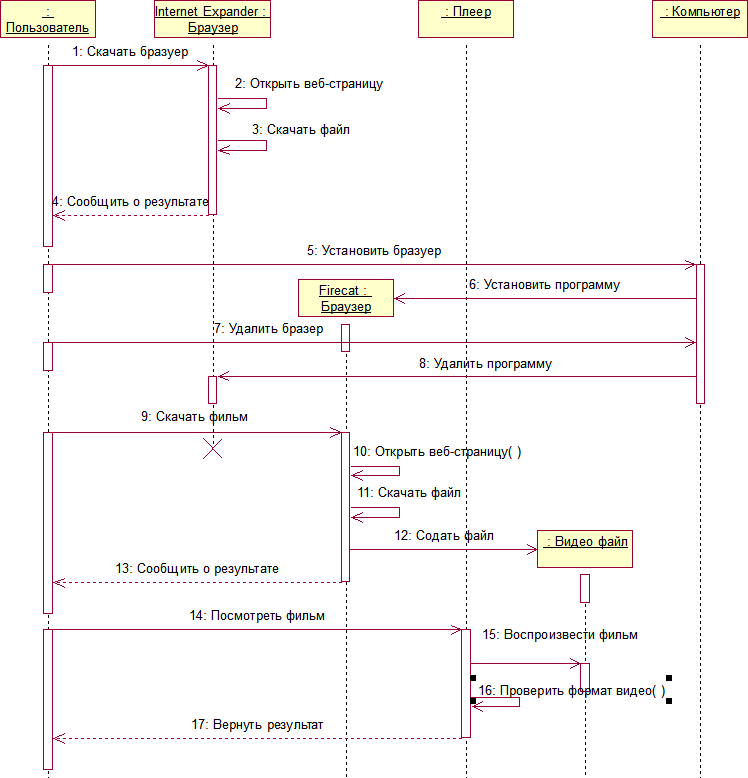


Рисунок 2 – Диаграмма последовательности

**Текст программы:**

// Абстрактный класс AbstractFile

abstract class AbstractFile(val path: String, protected val creationDate: String) {

//детали файла

abstract fun getDetails(): String//детали файла0

}

// Класс Program, наследующий AbstractFile

class Program(val name: String, private val author: String, path: String, creationDate: String) : AbstractFile(path, creationDate) {

override fun getDetails(): String {

return "Program: $name, Author: $author"

}

}

// Класс VideoFile, наследующий AbstractFile

class VideoFile(private val length: Int, private val format: String, path: String, creationDate: String) : AbstractFile(path, creationDate) {

override fun getDetails(): String {

return "Video File: Format: $format, Length: $length seconds"

}

//формат видео

fun getFormat(): String {

return format

}

}

// Класс Player

class Player(private val supportedFormats: List<String>) {

//воспроизведение программы

fun playProgram(program: Program) {

println("Playing program: ${program.getDetails()}")

}

//воспроизведение видео

fun playVideo(video: VideoFile) {

if (checkFormat(video)) {

println("Playing video: ${video.getDetails()}")

} else {

println("Unsupported video format: ${video.getFormat()}")

}

}

//проверка поддержки формата

private fun checkFormat(video: VideoFile): Boolean {

return supportedFormats.contains(video.getFormat())

}

}

// Класс Browser

class Browser {

fun downloadFile(file: AbstractFile) {

println("Downloading file: ${file.getDetails()}")

}

private fun openWebPage(url: String) {

println("Opening web page: $url")

}

}

// Класс Computer

class Computer {

private val programs = mutableListOf<Program>()

private val files = mutableListOf<AbstractFile>()

fun addProgram(program: Program) {

programs.add(program)

files.add(program)

println("Added program: ${program.getDetails()}")

}

fun removeProgram(program: Program) {

programs.remove(program)

files.remove(program)

println("Removed program: ${program.getDetails()}")

}

fun addFile(file: AbstractFile) {

files.add(file)

println("Added file: ${file.getDetails()}")

}

}

// Класс User

class User(private val name: String) { // Имя приватное

fun downloadProgram(browser: Browser, program: Program) {

browser.downloadFile(program)

}

fun installProgram(computer: Computer, program: Program) {

computer.addProgram(program)

}

fun useProgram(program: Program) {

println("$name is using ${program.getDetails()}")

}

fun useComputer(computer: Computer) {

println("$name is using the computer.")

}

}

// Пример использования

fun main() {

val user = User("Angelina")

val browser = Browser()

val computer = Computer()

val program = Program("Photo Editor", "Tom Noll", "/apps/photo\_editor", "2023-09-20")

val videoFile = VideoFile(120, "mp4", "/videos/my\_video.mp4", "2023-09-21")

// Пользователь скачивает и устанавливает программу

user.downloadProgram(browser, program)

user.installProgram(computer, program)

user.useProgram(program)

// Пользователь использует компьютер

user.useComputer(computer)

// Добавляем видеофайл в компьютер

computer.addFile(videoFile)

// Создаем плеер и воспроизводим видео и программу

val player = Player(listOf("mp4", "avi"))

player.playProgram(program)

player.playVideo(videoFile)}

Результат работы программы представлен на рисунке 3

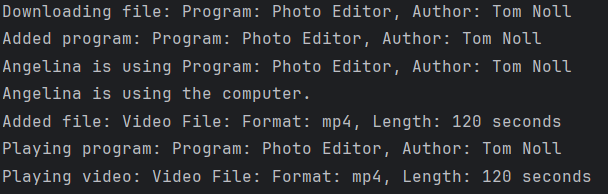


Рисунок 3 – Результат работы программы

**Ответы на вопросы к лабораторной работе:**

* 1. Что такое *JVM*?

JVM (Java Virtual Machine) – это виртуальная машина для запуска программ написанных на языках Java, Kotlin и др. Она управляет системной памятью и обеспечивает переносимую среду выполнения для приложения

1. Пакеты в *Kotlin*?

Используются для организации кода в логическую единицу и предотвращения конфликтов имен. В начале файла с помощью слова package.

1. Классы в *Kotlin*.

Объявляются с исп ключевого слова class. Класс может иметь первичный и вторичный конструкторы, свойства и методы.

1. Как объявить переменную в *Kotlin*? Различие между *var* и *val*.

Объявляются переменные с помощью var и val. Var исп для объявления изменяемых переменных. Val исп для объявления неизменяемых переменных.

1. Сколько конструкторов доступно в *Kotlin*?

Класс может иметь один первичный и один или несколько вторичных конструкторов.

1. Для чего служат абстрактные классы?

Абстрактный класс используется для определения общих характеристик, которые должны быть реализованы в подклассах.

1. Какой порядок вызова конструкторов при наследовании?

Сначала вызывается конструктор базового класса, затем конструктор производного класса.

1. Как переопределить метод родительского класса?

Передается с исп ключевого слова override.

1. Что такое оператор «*Elvis*» в *Kotlin* и как он используется?

Оператор «Elvis» (?:) исп для предоставления значения по умолчанию, если выражение слева от оператора равно null.

1. Как объявить и использовать лямбда-выражения в *Kotlin*?

Лямбда-выражения объявляются с использованием синтаксиса {параметр -> тело}. Предназначен для определения функциональных выражений.

1. Что такое модификаторы доступа (*access modifiers*) в *Kotlin*?

Они определяют видимость класса, методов и свойств. Public – доступен везде (по умолчанию), private – доступен только внутри файла или класса, protected – доступен в классе и его подклассах, internal – доступен внутри модуля.

1. Как работает оператор *when* в *Kotlin* и какие его особенности?

Он позволяет сравнить значение переменной с набором возможных значений и выполнить данный блок кода. Может быть исп как выражение, возвращающее значение, или как оператор, выполняющий блоки кода. Может объединять несколько условий в одну ветку с помощью запятой. Поддерживает проверку значений на принадлежность диапазону или коллекции с помощью операторов in и !in. When не требует break для выхода из блока кода, т.к. выполнение автоматически прекращается. Может сравнить значение с результатом произвольных выражений.

1. Как создать и использовать классы и объекты в *Kotlin*?

Классы создаются с исп ключевого слова class, может содержать свойства (переменные) и методы. Объекты с исп ключевого слова var или val и вызова конструктора класса. Класс может иметь один первичный конструктор и один или несколько вторичных конструкторов. Вторичные конструкторы объявляются с исп клбчевого слова constructur. Инициализаторы исп для выполнения кода при. Создании объекта. Они объявляются с исп ключ слова init.

1. Что такое *inline*-функции и зачем их применять?

Inline-функция исп для оптимизации кода, уменьшая накладные расходы на вызов функций.

1. Как обрабатывать исключения с помощью блока *try-catch* в *Kotlin*?

В блок try помещается код, который может вызвать исключение. Catch перехватывает и обрабатывает исключение если оно возникает.

1. Как работает оператор in для проверки наличия элемента в коллекции?

Оператор in исп для проверки наличия элемента в коллекции. Возвращает true, если элемент присутствует в коллекции и false если нет.

1. Как объявить и использовать расширения для стандартных классов в *Kotlin*?

Расширение позволяет добавлять новые функции и свойства к сущ кассам без необходимости наследования или исп паттернов. С помощью расширений (extensions).

1. Что такое функциональные типы данных (*function types*) и как их применять?

Функциональные типы данных представляют собой функции и могут быть использованы как параметры или возвращаемые значения.

**Вывод**: Изучена работа языка *Kotlin* и выполнена разработка приложения.