Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ **ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ**

КАФЕДРА **КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.04.01 Информатика и вычислительная техника**

МАГИСТЕРСКАЯ ПРОГРАММА **09.04.01/12 Интеллектуальный анализ больших данных в системах поддержки принятия решений**

О Т Ч Е Т

по лабораторной работе № 4

Название: Внутренние классы, интерфейсы

Дисциплина: Языки программирования для работы с большими данными

Студент	ИУ6-23М		Д.В. Пешков
	(Группа)	(Подпись, дата)	(И.О. Фамилия)
Преподаватель			П.В. Степанов
		(Подпись, дата)	(И.О. Фамилия)

Москва, 2024

Цель работы

Целью лабораторной работы является изучение внутренних классов и интерфейсов в языке Kotlin.

Задание

Вариант 1

10. Создать класс Cinema (кино) с внутренним классом, с помощью объектов которого можно хранить информацию об адресах кинотеатров, фильмах и времени сеансов.

11. Создать класс City (город) с внутренним классом, с помощью объектов которого можно хранить информацию о проспектах, улицах, площадях.

Вариант 2

Реализовать абстрактные классы или интерфейсы, а также наследование и полиморфизм для следующих классов

10. interface Фильм <- class Отечественный Фильм <- class Комедия.

11. Абстрактный класс Книга (Шифр, Автор, Название, Год, Издательство). Подклассы Справочник и Энциклопедия.

Ход работы

Составлены программы для выполнения всех требуемых задач. Каждая из задач была покрыта набором unit-тестов JUnit, был настроен CI для проверки прохождения тестов на каждое изменение в коде.

Фрагмент программного кода приведен в листинге 1.

Листинг 1 — Фрагмент задания 10 из варианта 2

```
package org.lab4

/**
 * Реализовать абстрактные классы или интерфейсы, а также наследование и полиморфизм для следующих классов
 *
 *    10.         interfaceФильм<-classОтечественныйФильм<-classКомедия.
 */
```

```
interface Film {
    fun play()
}

abstract class DomesticFilm : Film {
    abstract val country: String
}

class Comedy(override val country: String) : DomesticFilm()
{
    override fun play() {
        println("Playing a comedy film from $country")
    }
}

fun main() {
    val comedyFilm = Comedy("Russia")
    comedyFilm.play()
}
```

Полные программные коды программ доступны в репозитории: https://github.com/DPeshkoff/PLfBD.

Вывод

В ходе выполнения данной лабораторной работы были получены требуемые компетенции. Были изучены внутренние классы и интерфейсы в языке Kotlin.