|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **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 **09.04.01/07 Интеллектуальный анализ больших**

**данных в системах поддержки принятия решений.**

**Отчет**

**по лабораторной работе № 4**

**Вариант № 12**

**Название:** Внутренние классы и интерфейсы

**Дисциплина:** Языки программирования для работы с большими данными

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ6-22М |  |  | Т.М. Курохтин |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  | П.В. Степанов |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

Москва, 2024

**Цель:** изучить и освоить работу внутренних классов и интерфейсов в Java.

**Задание 1:** cоздать класс City (город) с внутренним классом, с помощью объектов которого можно хранить информацию о проспектах, улицах, площадях.

Код программы:

class City {

enum class LocationType {

PROSPECT,

STREET,

SQUARE

}

inner class Location(

private val name: String,

private val type: LocationType

) {

override fun toString(): String {

return "${type.name}: $name"

}

}

fun createLocation(name: String, type: LocationType): Location {

return Location(name, type)

}

}

fun main() {

val city = City()

val prospect = city.createLocation("Проспект Мира", City.LocationType.PROSPECT)

val street = city.createLocation("Тульская улица", City.LocationType.STREET)

val square = city.createLocation("Площадь Ленина", City.LocationType.SQUARE)

println(prospect)

println(street)

println(square)

}

Результат работы программы представлен на рисунке 1.

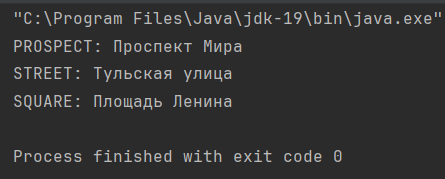


Рисунок 1 – Результат работы программы

**Задание 2:**  cоздать класс CD (mp3-диск) с внутренним классом, с помощью объектов которого можно хранить информацию о каталогах, подкаталогах и записях.

Код программы:

class CD {

inner class Track(

private val title: String,

private val artist: String,

private val duration: Int

) {

override fun toString(): String {

return "$title - $artist ($duration секунд)"

}

}

fun createTrack(title: String, artist: String, duration: Int): Track {

return Track(title, artist, duration)

}

}

fun main() {

val cd = CD()

val track1 = cd.createTrack("Песня 1", "Исполнитель 1", 180)

val track2 = cd.createTrack("Песня 2", "Исполнитель 2", 240)

val track3 = cd.createTrack("Песня 3", "Исполнитель 3", 200)

println(track1)

println(track2)

println(track3)

}

Результат работы программы показан на рисунке 2.

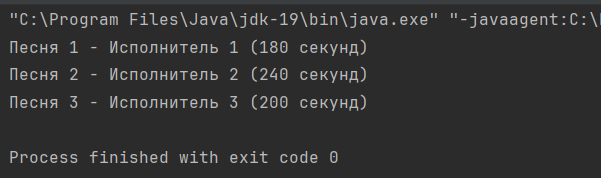


Рисунок 2 – Результат работы программы

**Задание 3:** абстрактный класс Книга (Шифр, Автор, Название, Год, Издательство). Подклассы Справочник и Энциклопедия.

Код программы:

abstract class Book(

val code: String,

val author: String,

val title: String,

val year: Int,

val publisher: String

) {

abstract fun getInfo(): String

}

class Directory(

code: String,

author: String,

title: String,

year: Int,

publisher: String,

private val category: String

) : Book(code, author, title, year, publisher) {

override fun getInfo(): String {

return "Справочник: $title, Автор: $author ($year), Издатель: $publisher, Категория: $category"

}

}

class Encyclopedia(

code: String,

author: String,

title: String,

year: Int,

publisher: String,

private val volume: Int

) : Book(code, author, title, year, publisher) {

override fun getInfo(): String {

return "Энциклопедия: $title, Автор: $author ($year), Издатель: $publisher, Том: $volume"

}

}

fun main() {

val directory = Directory("D001", "Автор 1", "Название 1", 2020, "Издатель 1", "Категория 1")

val encyclopedia = Encyclopedia("E001", "Автор 2", "Название 2", 2018, "Издатель 2", 3)

println(directory.getInfo())

println(encyclopedia.getInfo())

}

Результат работы программы представлен на рисунке 3.

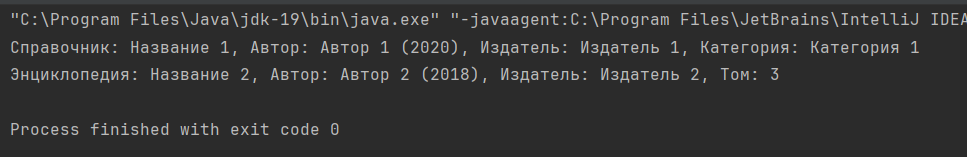
****

Рисунок 3 – Результат работы программы

**Задание 4:** interface Абитуриент <- abstract class Студент <- class Студент-Заочник.

Код программы:

interface Enrollee {

fun enroll()

}

abstract class Student : Enrollee {

abstract fun study()

}

class PartTimeStudent : Student() {

override fun study() {

println("Study as a part-time student...")

}

override fun enroll() {

println("Enrolling as a part-time student...")

}

}

fun main() {

val partTimeStudent = PartTimeStudent()

partTimeStudent.study()

partTimeStudent.enroll()

}

Результат работы программы показан на рисунке 4.

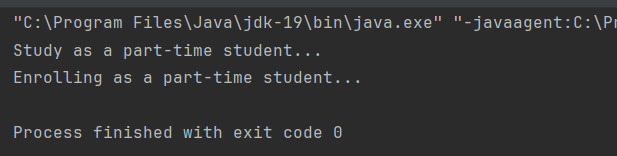


Рисунок 4 – Результат работы программы

**Вывод:** была освоена работа внутренних классов и интерфейсов в Java.