|  |
| --- |
|  |
| МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ |
| Федеральное государственное бюджетное образовательное учреждение высшего образования  **"МИРЭА - Российский технологический университет"**  **РТУ МИРЭА** |
| Институт информационных технологий |
| Кафедра инструментального и прикладного программного обеспечения |

|  |  |
| --- | --- |
|  | |
| **ОТЧЕТ ПО ПРАКТИЧЕСКИМ РАБОТАМ № 1 — 16** | |
| **по дисциплине** | |
| **«Шаблоны программных платформ языка Джава»** | |
| Выполнил студент группы **ИКБО-30-22** | ***Сулейманов А.В.*** |
| Принял ассистент кафедры ИиППО | ***Ермаков С.Р.*** |

|  |  |  |
| --- | --- | --- |
| Практические работы выполнены | « » 2024 г. |  |
| «Зачтено» | « » 2024 г. |  |

Москва 2024

**Практическая работа №1**

1. **Постановка задачи и персональный вариант**

Тема: Встроенные функциональные интерфейсы Java. Возможности Java 8. Лямбда-выражения. Области действия, замыкания. Предикаты. Функции. Компараторы.

Постановка задачи: имплементировать интерфейс Predicate, определяющий, является ли данная строка email-адресом, используя регулярное выражение.

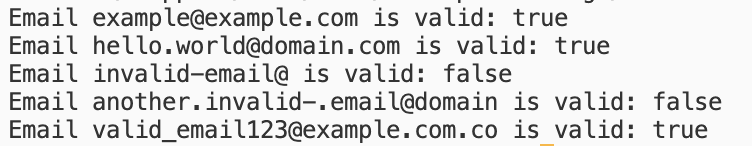
1. **Код и пояснение к коду**

Код: <https://github.com/stufade/java_pattern_homework/tree/main/hw_1>

Пояснения:

Класс EmailPredicate – класс, реализующий интерфейс Predicate<String>, определяющий, является ли данная строка email-адресом, используя регулярное выражение.

1. **Результат работы программы**

****

**Практическая работа №2**

1. **Постановка задачи и персональный вариант**

Тема: Работа со Stream API в Java 8.

Постановка задачи: в ходе выполнения практической работы должно быть реализовано:

1) класс Human (int age, String firstName, String lastName, LocalDate birthDate, int weight);

2) приложение, которое создает список из объектов класса Human, а затем производит сортировку по второй букве имени в обратном порядке, фильтрацию по весу больше, чем 60, сортировку по возрасту, и считает произведение всех возрастов.

1. **Код и пояснение к коду**

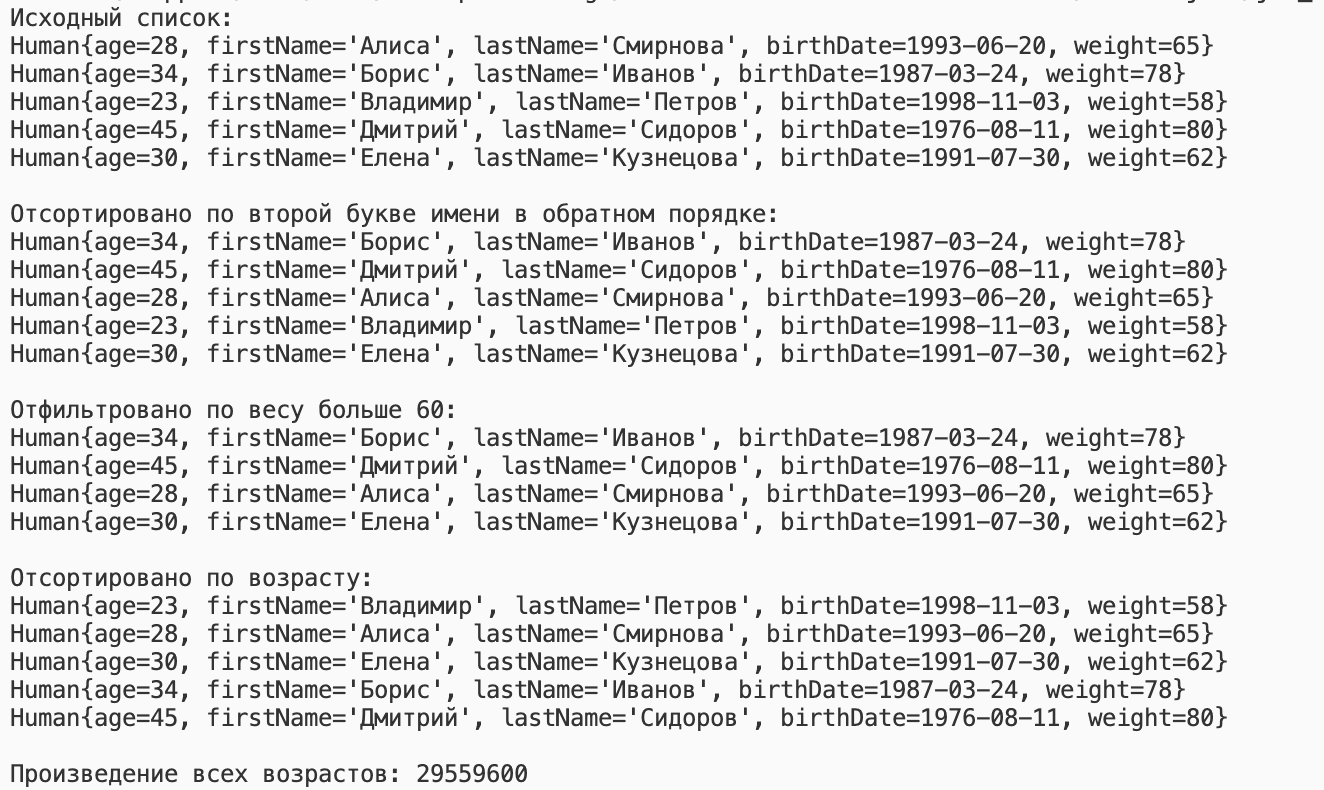
Код: <https://github.com/stufade/java_pattern_homework/tree/main/hw_2>

Пояснения:

Класс Human – реализация класса Human.

Класс HumanApp – класс, выполняющий операции, указанные в условии задания.

1. **Результат работы программы**

****

**Практическая работа №3**

1. **Постановка задачи и персональный вариант**

Тема: Знакомство с конкурентным программированием в Java. Потокобезопасность, ключевое слово syncrhonized, мьютексы, семафоры, мониторы, барьеры.

Постановка задачи: Map с использованием ключевого слова synchronized, Set с использованием Semaphore.

1. **Код и пояснение к коду**

Код: <https://github.com/stufade/java_pattern_homework/tree/main/hw_3>

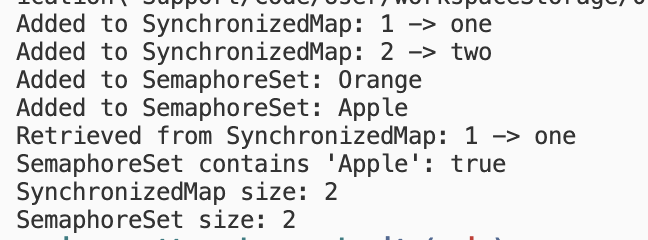
Пояснения:

Класс SynchronizedMap - Потокобезопасная реализация Map, для потокобезопасного доступа к ресурсу Map используется ключевое слово synchronized, оно гарантирует, что только один поток может выполнять методы этого объекта в данный момент времени.

Класс SemaphoreSet - Потокобезопасная реализация Set, в этом классе используется Semaphore для контроля доступа к ресурсу

Класс Main – производит тестирование.

1. **Результат работы программы**

****

**Практическая работа №4**

1. **Постановка задачи и персональный вариант**

Тема: Работа с ExecutorService, CompletableFuture.

Постановка задачи: реализовать собственную имплементацию ExecutorService с единственным параметром конструктора – количеством потоков.

1. **Код и пояснение к коду**

Код: <https://github.com/stufade/java_pattern_homework/tree/main/hw_4>

Пояснения:

Класс SimpleExecutorService реализован с помощью BlockingQueue<Runnable> taskQueue и List<WorkerThread> workerThreads для управления потоками.

1. **Результат работы программы**

****

**Практическая работа №5**

1. **Постановка задачи и персональный вариант**

Тема: познакомиться с паттернами проектирования, их определением и классификацией. Обзор паттернов GoF. Паттерн Синглтон.

Постановка задачи: реализовать паттерн Singleton как минимум 3-мя способами

1. **Код и пояснение к коду**

Код: <https://github.com/stufade/java_pattern_homework/tree/main/hw_5>

Пояснения:

Класс Singleton1 реализован через статический метод.

Класс Singleton2 реализован через приватный класс и метод.

Класс Singleton3 реализован через enum.

1. **Результат работы программы**

****

**Практическая работа №6**

1. **Постановка задачи и персональный вариант**

Тема: знакомство с реализацией порождающих паттернов проектирования.

Постановка задачи: написать реализацию паттернов «Фабричный метод», «Абстрактная фабрика», «Строитель», «Прототип».

1. **Код и пояснение к коду**

Код: <https://github.com/stufade/java_pattern_homework/tree/main/hw_6>

Пояснения:

Класс AbsctractFactoryMethod реализует паттерн «Абстрактная фабрика».

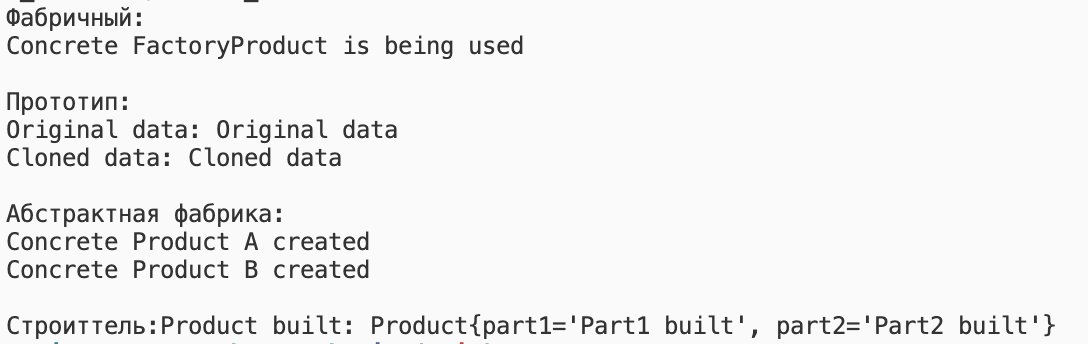
Класс BuilderMethod реализует паттерн «Строитель».

Класс AbsctractFactoryMethod реализует паттерн «Фабричный метод».

Класс PrototypeMethod реализует паттерн «Прототип».

Класс Main производит тестирование.

1. **Результат работы программы**

****

**Практическая работа №7**

1. **Постановка задачи и персональный вариант**

Тема: реализация структурных паттернов проектирования.

Постановка задачи: написать реализацию паттернов Легковес, Компоновщик.

1. **Код и пояснение к коду**

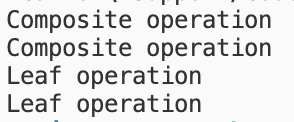
Код: <https://github.com/stufade/java_pattern_homework/tree/main/hw_7>

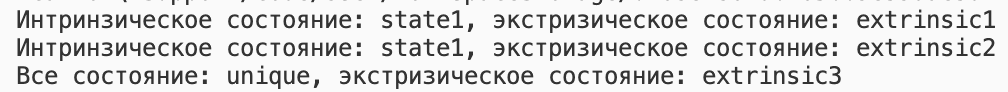
Пояснения:

Класс CompositeDemo реализует паттерн «Компоновщик» и производит тестирование.

Класс Flyweight реализует паттерн «Легковес» и производит тестирование.

1. **Результат работы программы**

****

****

**Практическая работа №8**

1. **Постановка задачи и персональный вариант**

Тема: реализация структурных паттернов проектирования.

Постановка задачи: написать реализацию паттернов Посетитель, Команда.

1. **Код и пояснение к коду**

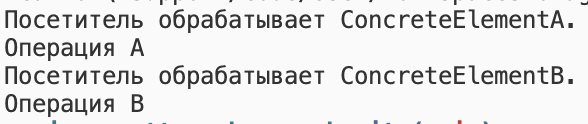
Код: <https://github.com/stufade/java_pattern_homework/tree/main/hw_8>

Пояснения:

Класс VisitorDemo реализует паттерн «Посетитель» и производит тестирование.

Файл CommandDemo реализует паттерн «Команда» и производит тестирование.

1. **Результат работы программы**

****

****

**Практическая работа №9**

1. **Постановка задачи и персональный вариант**

Тема: Знакомство с системой сборки приложения. Gradle.

Постановка задачи: создать приложение, которое выводит какое-то сообщение в консоль. Создать Gradle Task, который создает jar-файл приложения, переносит его в отдельную папку, в которой хранится Dockerfile для jar, а затем создает Docker контейнер из данного jar-файла и запускает его.

1. **Код и пояснение к коду**

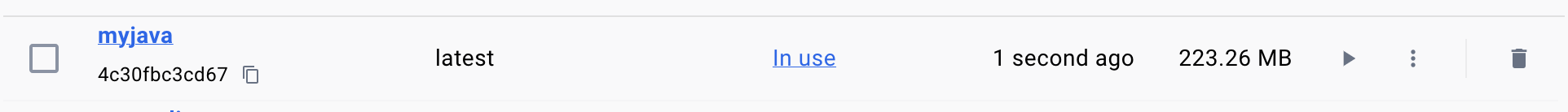
Код: <https://github.com/stufade/java_pattern_homework/tree/main/hw_9>

Пояснения:

Написаны таски runDocker, buildDocker, moveJar, createJar для выполнения условия задания.

1. **Результат работы программы**

****

****

**Практическая работа №10**

1. **Постановка задачи и персональный вариант**

Тема: Введение в Spring. Container. Bean. Внедрение зависимостей, основанных на конструкторах и сеттерах. Конфигурация бинов. Автоматическое обнаружение и связывание классов.

Постановка задачи: создать приложение, в котором создается ApplicationContext и из него берётся бин с названием, переданным в качестве аргумента к приложению, и вызывается метод интерфейса, который он имплементирует. Нужно создать по одному бину для каждого класса, определить им название. Проверить, что вызывается при вводе названия каждого из бинов. Интерфейс Programmer с методом doCoding(), его имплементации: Junior, Middle, Senior.

1. **Код и пояснение к коду**

Код: <https://github.com/stufade/java_pattern_homework/tree/main/hw_10>

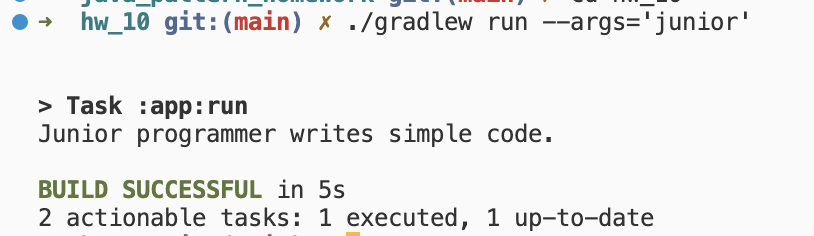
Пояснения:

Классы Junior, Middle и Senior и интерфейс Programmer реализованы в соответствии с постановкой задачи.

Класс AppConfig – класс с конфигурацией Spring и бины для каждого типа программиста.

Программа запускается командой ./gradlew run –args=’arg’, где arg – название нужного бина.

1. **Результат работы программы**

****

**Практическая работа №11**

1. **Постановка задачи и персональный вариант**

Тема: Использование Sprting boot.

Постановка задачи: создать приложение с использованием Spring Boot Starter Initializr (https://start.spring.io/) с такими зависимостями:

– Spring Web;

– Lombok;

– Validation;

– Spring boot Actuator.

Запустить приложение и удостовериться, что не появилось никаких ошибок. Добавить все эндпоинты в Actuator, сделать HTTP-запрос на проверку состояния приложения. Собрать jar-файл приложения, запустить и проверить состояние при помощи REST-запроса.

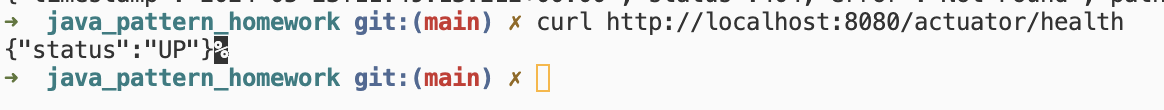
1. **Код и пояснение к коду**

Код: <https://github.com/stufade/java_pattern_homework/tree/main/hw_11>

Пояснения:

Была добавлена строчка в файл application.properties для включения всех эндпоинтов в Actuator - management.endpoints.web.exposure.include=\*.

1. **Результат работы программы**

****

**Практическая работа №12**

1. **Постановка задачи и персональный вариант**

Тема: Работа с жизненным циклом компонентов. Аннотации PostConstruct, PreDestroy.

Постановка задачи: создать приложение, которое при запуске берет данные из одного файла, хеширует, а при остановке приложения удаляет исходный файл, оставляя только файл с захешированными данными. Названия первого и второго файла передаются в качестве аргументов при запуске. При отсутствии первого файла создает второй файл и записывает в него строку null. Реализовать с использованием аннотаций PostConstruct, PreDestroy.

1. **Код и пояснение к коду**

Код: <https://github.com/stufade/java_pattern_homework/tree/main/hw_12>

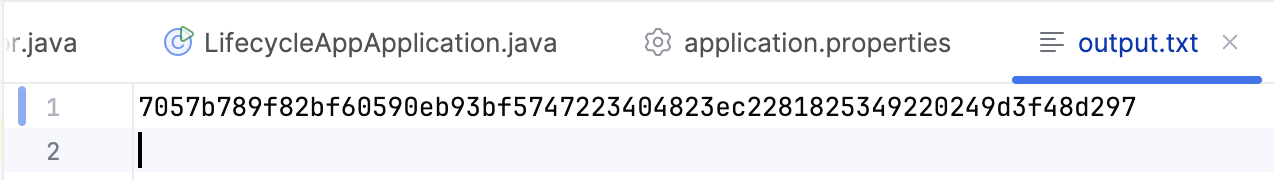
Пояснения:

Класс FileProcessor создан для работы с файламифайлами.

Аннотация @PostConstruct применена к методу processFile(), который производит запись в файл output.txt.

Аннотация @PreDestroy применена к методу cleanUp(), который удаляет исходный файл input.txt.

1. **Результат работы программы**

****

**Практическая работа №13**

1. **Постановка задачи и персональный вариант**

Тема: Конфигурирование приложения. Environment.

Постановка задачи: создать файл application.yml в папке resources, добавить в него такие свойства:

– student.name – имя студента;

– student.last\_name – фамилия студента;

– student.group – название группы студента.

При запуске приложения выведите данные свойства в консоль при помощи интерфейса Environment или аннотации Value.

1. **Код и пояснение к коду**

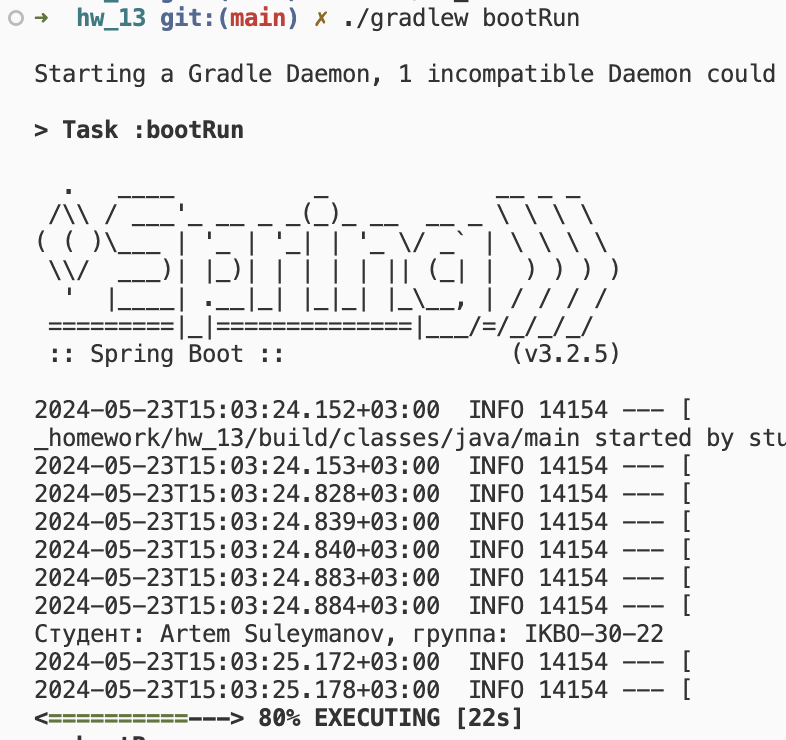
Код: <https://github.com/stufade/java_pattern_homework/tree/main/hw_13>

Пояснения:

Создан application.yml в соответствии с постановкой задачи.

Класс Application приложения имплементирует использует Environment для получения данных из application.yml.

1. **Результат работы программы**

****

**Практическая работа №14**

1. **Постановка задачи и персональный вариант**

Тема: Знакомство со Spring MVC. Работа с Rest API в Spring.

Постановка задачи: Создать отдельный репозиторий Git. Создать простой html-документ, который будет содержать вашу фамилию, имя, номер группы, номер варианта. Создать контроллер, который будет возвращать данный статический документ при переходе на url «/home». Создать класс Student с полями firstName, lastName, middleName. Создать класс University с полями name, creationDate. Создать классыконтроллеры для создания, удаления объектов и получения всех объектов каждого типа. Сами объекты хранить в памяти.

1. **Код и пояснение к коду**

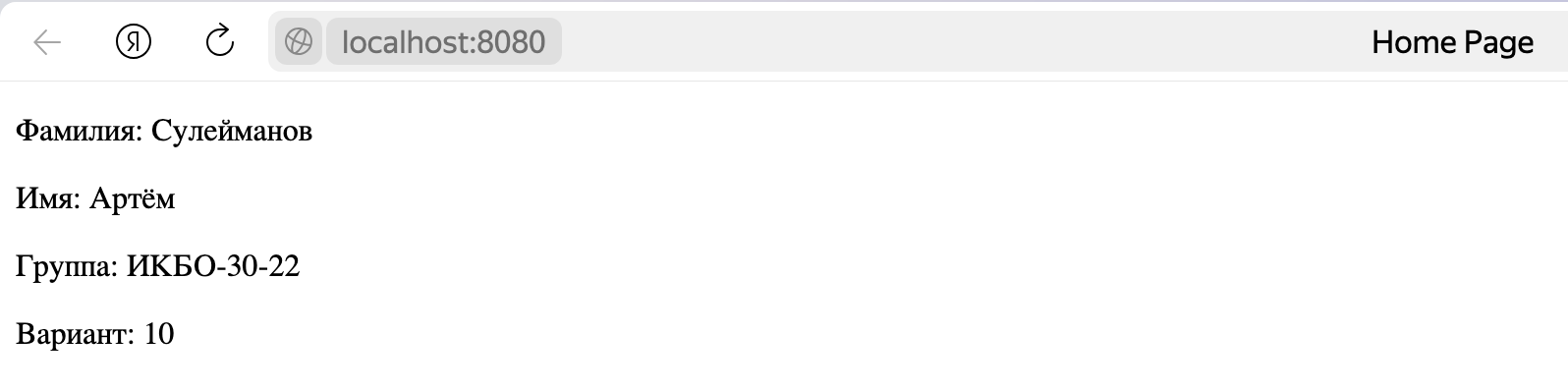
Код: <https://github.com/stufade/java_pattern_homework/tree/main/hw_14>

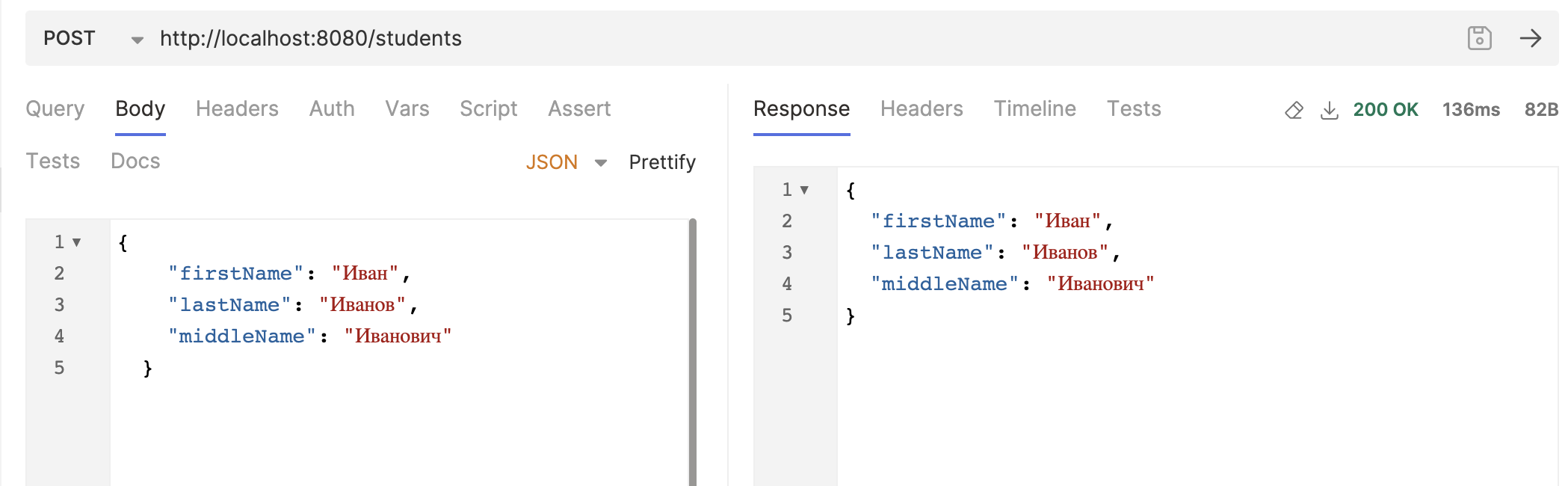
Пояснения:

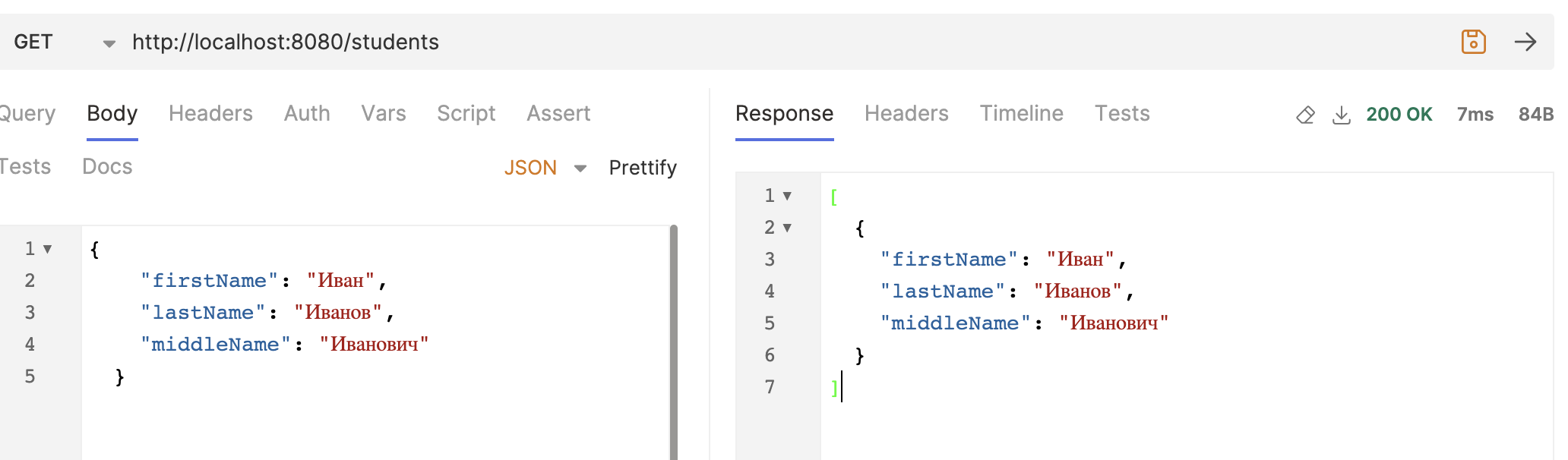
В папке controllers находятся классы контроллеров.

В папке models находятся модели Student и University.

1. **Результат работы программы**

****

****

****

**Практическая работа №15**

1. **Постановка задачи и персональный вариант**

Тема: Использование Hibernate в Spring framework.

Постановка задачи: изменить программу с предыдущего задания так, чтобы объекты хранились в базе данных PostgreSQL вместо памяти компьютера.

1. **Код и пояснение к коду**

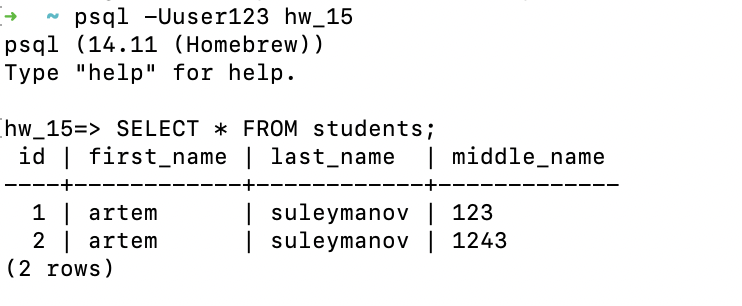
Код: <https://github.com/stufade/java_pattern_homework/tree/main/hw_15>

Пояснения:

Добавлены строчки в application.properties для подключения СУБД PostgreSQL.

Добавлена папка repositories, где находятся JPA-репозитории для моделей.

1. **Результат работы программы**

****

**Практическая работа №16**

1. **Постановка задачи и персональный вариант**

Тема: изучение видов связей между сущностями в Hibernate. Использование транзакций.

Постановка задачи: создать связь Один-ко-многим между сущностями из предыдущего задания и проверить работу lazy loading.

1. **Код и пояснение к коду**

Код: <https://github.com/stufade/java_pattern_homework/tree/main/hw_16>

Пояснения:

Изменены модели для добавления связи один-ко-многим.

1. **Результат работы программы**

