**Министерство науки и высшего образования Российской Федерации**

Федеральное государственное автономное образовательное учреждение

высшего образования «Уральский федеральный университет

имени первого Президента России Б.Н. Ельцина»

Институт радиоэлектроники и информационных технологий - РТФ

Центр ускоренного обучения

Отчет по лабораторной работе № 1

по дисциплине «Технологии Программирования»

Старший преподаватель: Н.А. Архипов

Студент гр. РИВ-220938у М.С. Ермаков

Екатеринбург 2023

**1. Цель работы:**

**Цель:** Создание простейшего веб-приложения с помощью фреймворка Spring Boot

**Учебные вопросы:**

1) Реализовать простейшее Spring Boot приложение

2) Реализовать дополнительный функционал в разработанном по примеру Spring Boot приложении

3) Ответить письменно на вопросы в соответствии с номером в списке

4) Оформить отчет и прикрепить его на сайте https://edu.itlearn.ru/ в соответствующем курсе, в соответствующем разделе

**2. Описание задачи:**

Реализация дополнительного функционала

Необходимо самостоятельно доработать приложение:

1. Создайте GET-метод **updateArrayList**(String s), который по url “/update-array” принимает аргумент и записывает его в ArrayList<String >, реализуйте логику - если это первый вызов метода, то создается пустой ArrayList, если он не пустой, то туда записывается значение.

2. Создайте GET метод **showArrayList**(), который по url “/show-array” возвращается все элементы хранящиеся в ArrayList <String >, созданном в п. 1

3. Создайте GET-метод **updateHashMap**(String s), который по url “/update-map” принимает аргумент и записывает его в HashMap<String, String>, реализуйте логику - если это первый вызов метода, то создается пустой HashMap, если он не пустой, то туда записывается значение.

4. Создайте GET метод **showHashMap** (), который по url “/show-map” возвращается все элементы хранящиеся в HashMap<String, String>, созданном в п. 3

5. Создайте GET метод **showAllLenght** (), который по url “/show-all-lenght” возвращает текст, в котором указано количество элементов в ArrayList и HashMap

Вопросы

Письменно ответьте на вопросы, если ваш номер в списке группы четный, то нужно ответить на четные вопросы, если нечетный, то нужно ответить на нечетные вопросы.

1. Чем отличаются системы сборки maven и gradle?

2. Чем отличаются языки программирования Java и Kotlin?

3. Чем отличаются языки программирования Java и Groovy?

4. Чем отличаются языки программирования Kotlin и Groovy?

5. Что нового добавилось в Java 11 в отличии от Java 8?

6. Что нового добавилось в Java 17 в отличии от Java 11?

**3. Ход выполнения:**

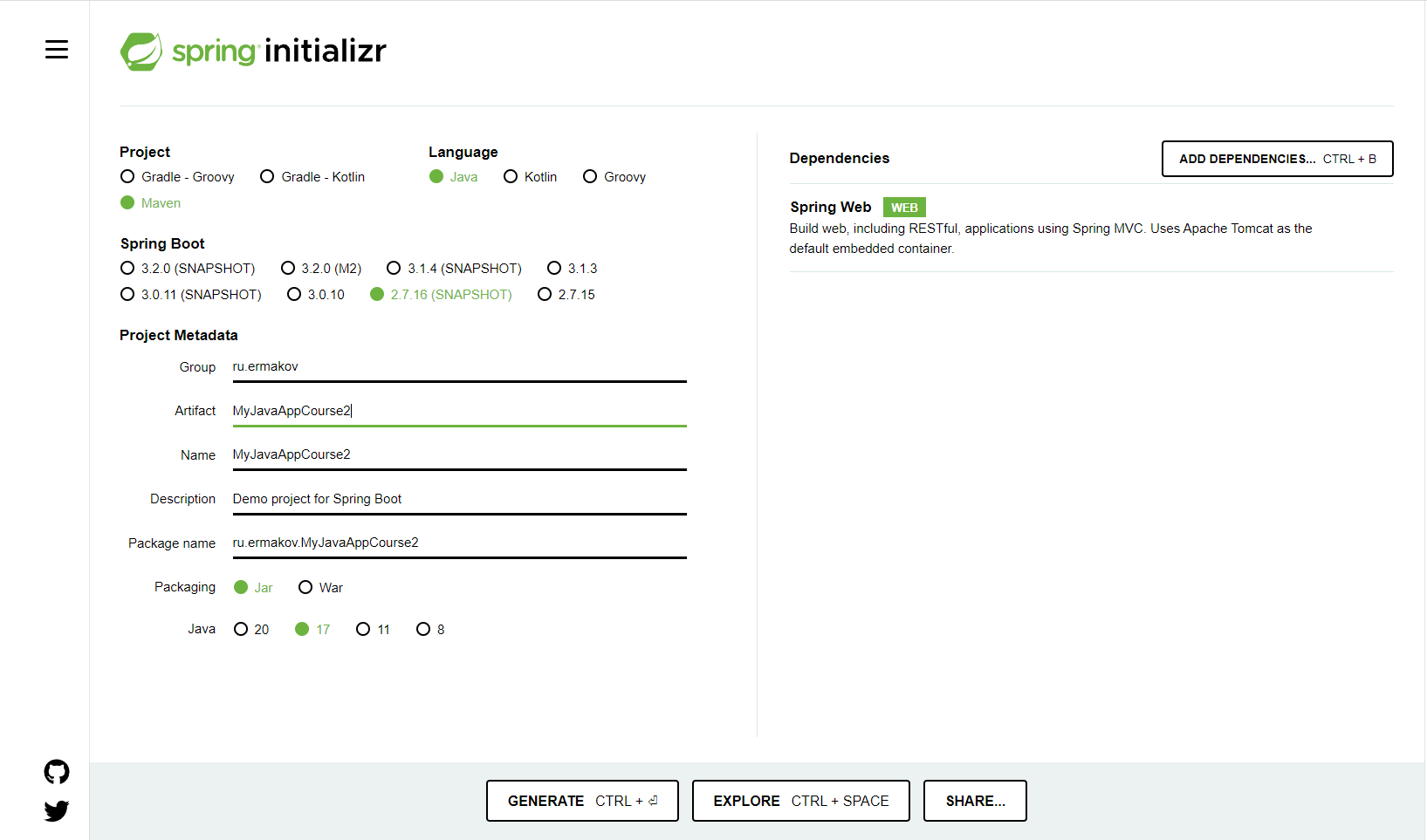


Рисунок 1 – Создание проекта в Spring initializr

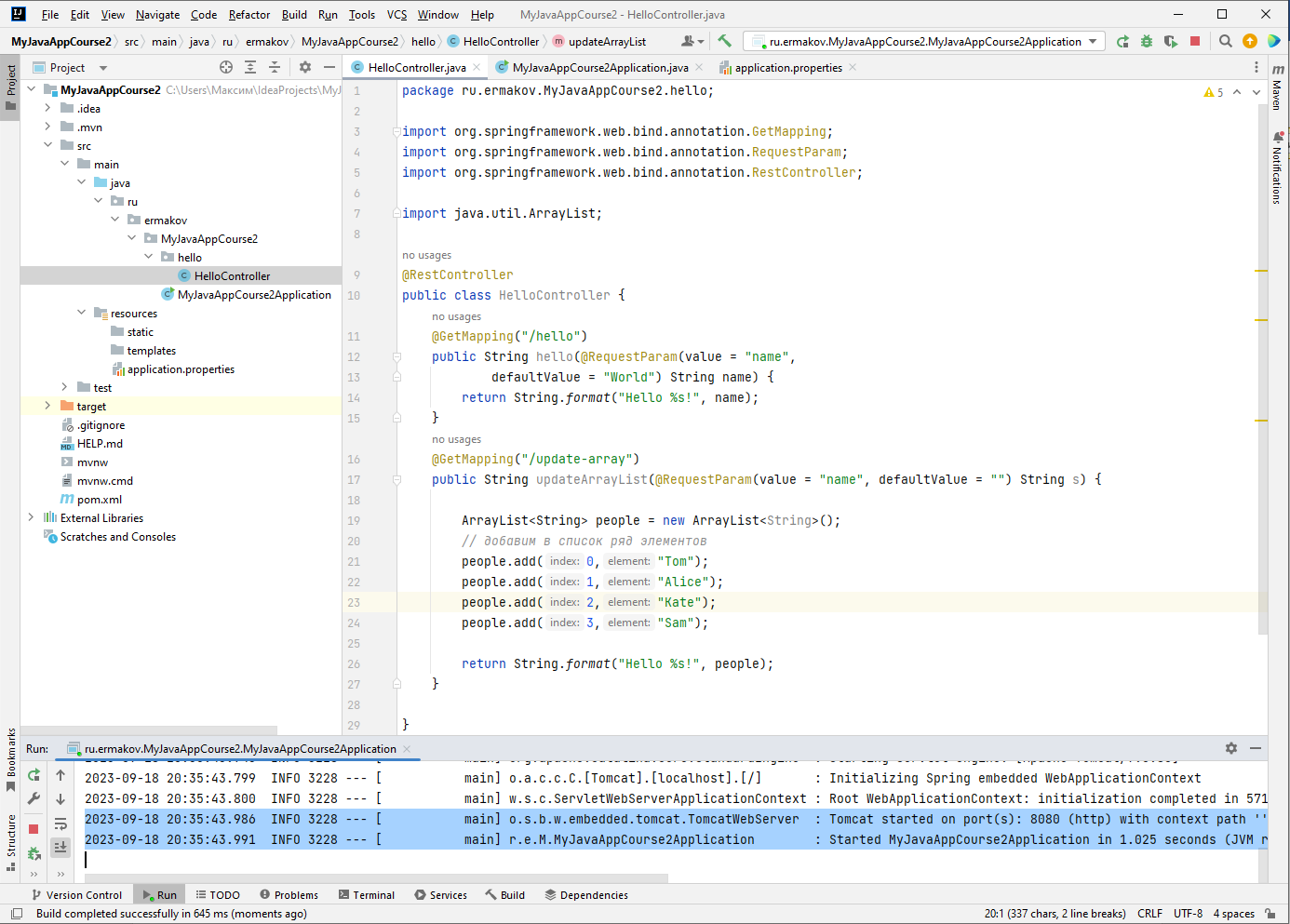


Рисунок 2 – Создание проекта в IJ и класса HelloController



Рисунок 3 – Тестирование приложения в браузере

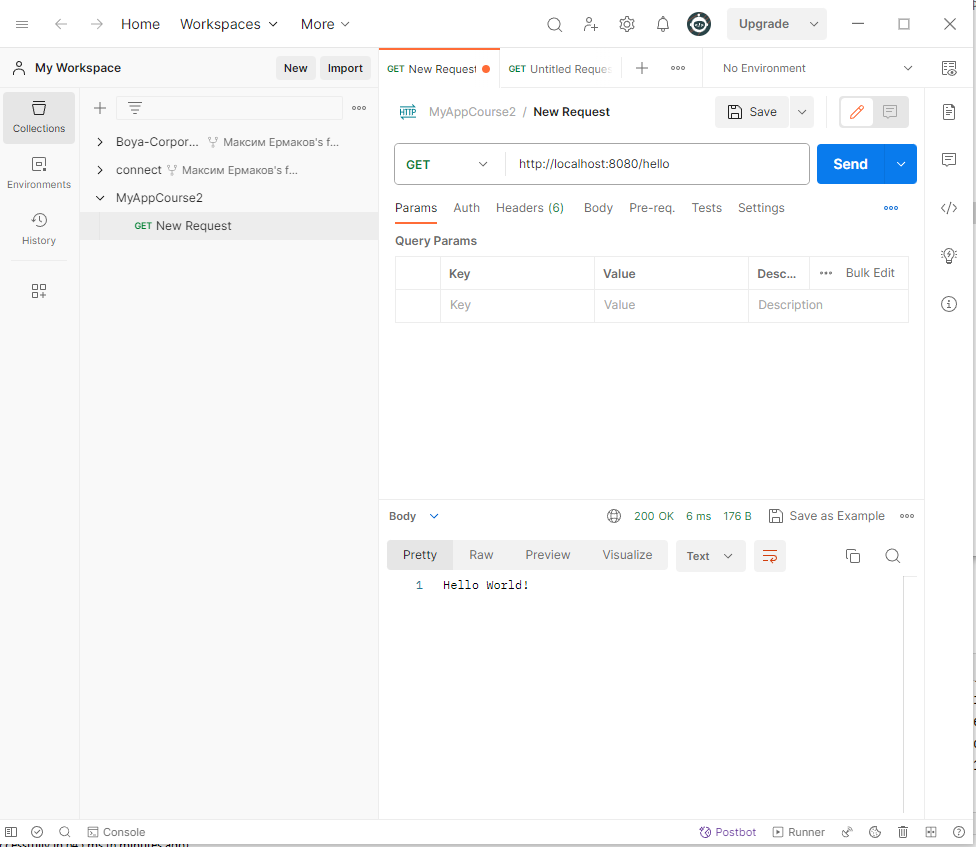


Рисунок 4 – Результат тестирования запроса с помощью Postman №1

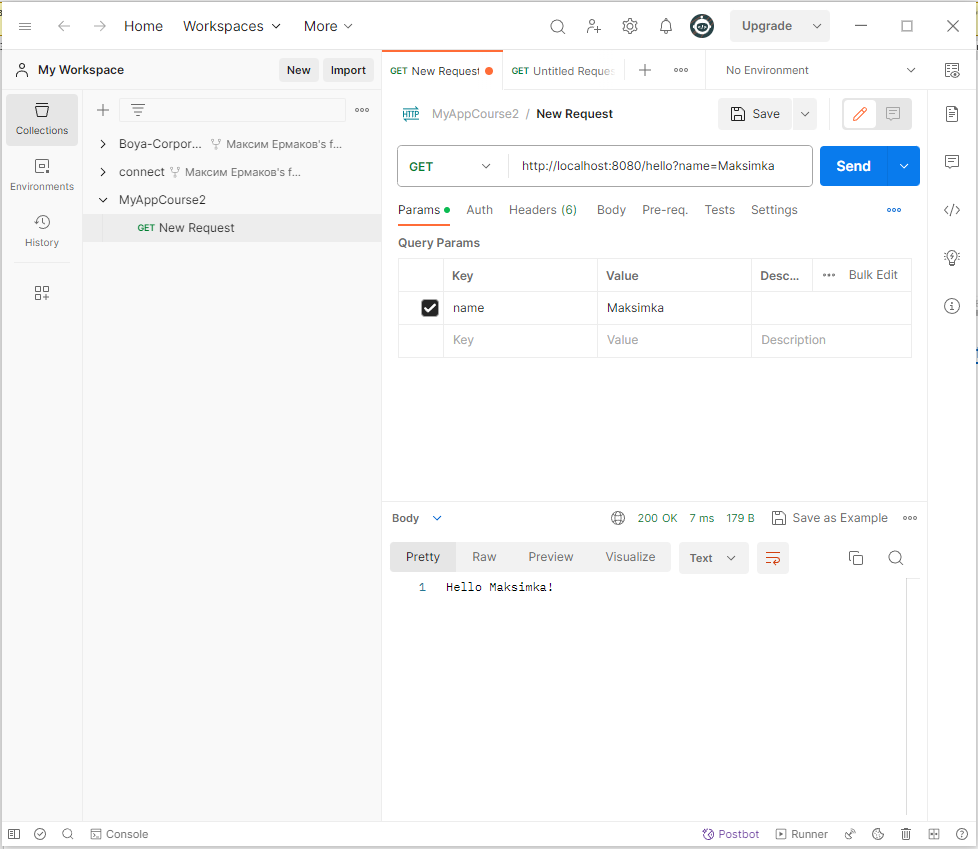


Рисунок 5 – Результат тестирования запроса с помощью Postman №2

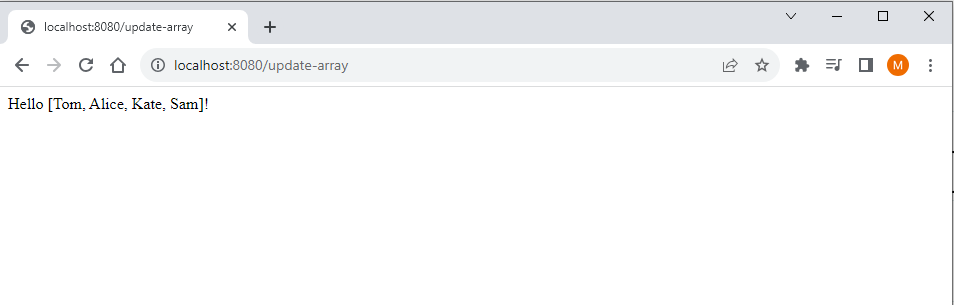


Рисунок 6 – Результат добавления Arraylist в браузере через команду update–array

**Ответы на вопросы:**

2) Некоторые проблемы Java, решённые в Kotlin

* Kotlin решает целый ряд проблем, от которых страдает Java:
* Ссылки на null контролируются системой типов.
* Нет сырых (raw) типов
* Массивы в Kotlin инвариантны
* Kotlin имеет правильные функциональные типы и поддерживает их использование вместо SAM-типов из Java
* Вариативность на месте использования без подстановочных символов (или масок, ориг.: wildcards)
* В Kotlin нет проверяемых исключений

Что есть в Java, но нет в Kotlin

* Проверяемые исключения
* Примитивные типы, которые не являются классами
* Статичные члены
* Не-приватные поля
* Подстановочные символы (маски, wildcards)

Что есть в Kotlin, но нет в Java

* Лямбда-выражения + inline-функции = производительные и контролируемые пользовательские структуры
* Функции-расширения
* Null-безопасность
* Умные приведения
* Строковые шаблоны
* Свойства
* Первичный конструктор
* Делегирование на уровне языка
* Выведение типа для переменных и свойств
* Синглтоны на уровне языка
* Вариативность на уровне объявления и Проекции типов
* Интервалы
* Перегрузка операторов
* Вспомогательные объекты
* Классы данных
* Раздельные интерфейсы для изменяемых и неизменяемых коллекций
* Сопрограммы (корутины)

4) Ключевые различия между Groovy и Kotlin

Оба Groovy против Kotlin производительность является одобренным выбором в отрасли. Давайте рассмотрим некоторые заметные различия между Groovy и Kotlin:

* Учитывая, что Groovy был создан как динамический язык, также @Compile Static, в то время как всеобъемлющая аннотация, была прикреплена ниже. Его характеристика кажется немного запутанной, более того, он не заставляет людей кодировать статическим образом. Это не применимо везде. Однако иногда даже в Groovy время от времени создается уникальная динамическая производительность. Котлин на 100% статичен, более того, динамический не вариант. Котлин спроектирован как язык со статической типизацией, включая великолепную систему типов плюс дополнительные преимущества языка со статической типизацией. Groovy - в начальной позиции это язык с динамической типизацией, также просто тогда - статически.
* Между тем, когда мы разрешаем компилировать static в groovy, мы получаем явно Java. С другой стороны - Kotlin, в их системе типов, имеет два типа ссылок: обнуляемый и ненулевой, так что вы можете составлять код ограниченным NPE.
* Следующая исключительная особенность Kotlin - он не делает никаких неявных преобразований, с другой стороны - groovy неявно преобразует double в big decimal и так далее.
* Хотя у Kotlin есть много других характеристик, таких как умные приведения, ADT (doc), безопасные для типов компоновщики, нулевые абстракции и, наконец, отличная поддержка IDE.
* Кроме того, работа по выводу типов Kotlin похожа на обаяние, без каких-либо аннотаций на каждом языке, кроме Groovy.
* Тернарный оператор: оба, Kotlin vs Groovy, предоставляют оператора Элвиса, однако, троичный оператор отсутствует в последнем.

6) Switch-выражения из Java 14

Чтобы получить значение из switch-выражения, раньше приходилось создавать отдельную переменную и постоянно использовать break;. Вот как это выглядело:

String season;

switch (month) {

case JANUARY:

case FEBRUARY:

season = "winter";

break;

case MARCH:

case APRIL:

case MAY:

season = "spring";

break;

В Java 14 появился новый формат записи, который помогает получать результат выбора и записывать выражение компактнее. Если перечислены все возможные варианты, ветка default теперь не нужна:

String season = switch (month) {

case JANUARY, FEBRUARY -> "winter"; //несколько вариантов

case MARCH, APRIL, MAY -> "spring";

case JUNE, JULY, AUGUST -> "summer";

case SEPTEMBER, OCTOBER, NOVEMBER -> "autumn";

case DECEMBER -> "winter"; //oдин вариант

}

Текстовые блоки из Java 15

Если раньше нужно было использовать литерал в несколько строк, его собирали через конкатенацию:

String query = "SELECT Students.name as \"Name\", SUM(Courses.duration) as \"Duration\"\n"

+ "FROM Students\n"

+ "JOIN Subscriptions ON Students.id = Subscriptions.student\_id\n"

+ "JOIN Courses ON Subscriptions.course\_id = Courses.id\n"

+ "GROUP BY Students.id\n"

+ "ORDER BY Students.name";

После того как появились текстовые блоки, это делают одним блоком. Текст с переносами и кавычками заключают в тройные кавычки:

String query = """

SELECT Students.name as "Name", SUM(Courses.duration) as "Duration"

FROM Students

JOIN Subscriptions ON Students.id = Subscriptions.student\_id

JOIN Courses ON Subscriptions.course\_id = Courses.id

GROUP BY Students.id

ORDER BY Students.name""";

System.out.println(query);

Новые варианты применения instanceof из Java 16

Чтобы проверить, к какому классу относится объект, используют оператор instanceof. Если нужно проверить объект и привести его к нужному виду, раньше объявляли переменную, присваивали ей тип, а затем проверяли объект:

Object string = "this is string!";

if(string instanceof String){

String realString = (String) string;

System.out.println(realString);

}

Начиная с Java 16 присвоение не требуется. Значение переменной можно задать прямо в выражении:

if(string instanceof String realString){

System.out.println(realString);

}

Запечатанные классы из Java 17

Sealed class дословно переводится как «запечатанный класс». В этом классе нужно сразу объявить список классов-наследников, потому что кроме них наследников быть не может. Это похоже на enum, только в разрезе наследования.

Посмотрим, как это выглядит в коде:

sealed class Person permits Student, Teacher, Curator {}

Классы Student, Teacher и Curator должны быть в том же пакете или модуле, что и Person. Кроме этого, у них обязательно должен быть один из модификаторов:

final, если класс запрещён к дальнейшему наследованию:

public final class Student extends Person{}

sealed, если наследование допустимо, но с заранее указанным списком наследников:

public sealed class Teacher extends Person permits MathTeacher, LanguageTeacher {}

non-sealed, когда для класса нужно снять любые ограничения по наследованию:

public non-sealed class Curator extends Person {}

У интерфейсов тоже может быть модификатор sealed:

sealed interface Person

permits Student, Teacher, Curator {

}

С учётом record мы можем имплементировать интерфейс и записать класс Student:

public record Student(String name) implements Person{}

Здесь record по умолчанию final, поэтому ограничения класса sealed соблюдены.

Для тех, кто будет переходить с 11, Java 17 будет достаточно большим апгрейдом. Если сложить все релизы с 12 по 17, то в сумме появилось 5 больших языковых конструкций: выражения switch, блоки текста, записи, паттерн-матчинг для instanceof и sealed классы. Появилось очень много интересного API, сделано много улучшений производительности, оптимизировано потребление памяти, уменьшено время старта JVM. Очень много чего было удалено, поэтому могут возникнуть некоторые трудности при миграции.

**Ссылка** на репозиторий **github.com** =

<https://github.com/themakson/MyJavaAppCourse2/tree/main/MyJavaAppCourse21>

**Вывод:**

В ходе лабораторной работы 1 мы научились создавать простейшие веб–приложения с помощью фреймворка Spring Boot в языке программирования Java, расширили функционал веб приложения, сравнили отличия между языками groovy, kotlin и java, а также рассмотрели версию java 17 с предыдущими его версиями.

|  |  |
| --- | --- |
| Сдал: | М.С. Ермаков |
| Проверил: | Н.А. Архипов |