Министерство науки и высшего образования Российской Федерации

**Федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

«Владимирский государственный университет

имени Александра Григорьевича и Николая Григорьевича Столетовых»

(ВлГУ)

Кафедра информационных систем и программной инженерии

**Лабораторная работа №2**

**по дисциплине**

**«Распределенные программные системы»**

**ТИПОВАЯ СТРУКТУРА JAVA ПРИЛОЖЕНИЯ. ИСПОЛЬЗОВАНИЕ СИСТЕМЫ УПРАВЛЕНИЯ ЖЦ ПРОЕКТА APACHE MAVEN.**

**Выполнил**:

ст. гр. ПРИ-120

Д. А. Грачев

**Принял**:

Проскурина Г. В.

Владимир, 2022

ЦЕЛЬ РАБОТЫ

Познакомиться с принципами разработки Java-приложений с использованием автоматизированной системы управления жизненным циклом проекта Apache Maven. Получить практические навыки решения типовых задач при сборке Java проекта.

ВЫПОЛНЕНИЕ РАБОТЫ

1. Создадим иерархию каталогов и файл Application.java

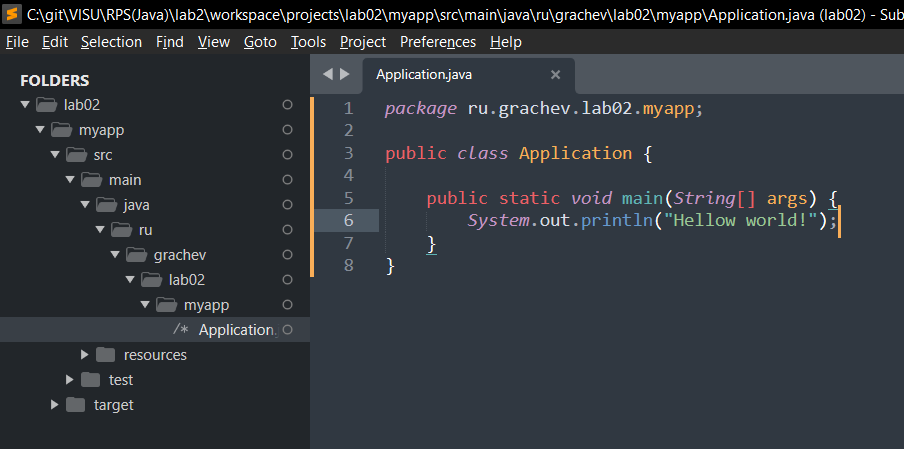


Рисунок . Application.java

1. Скомпилируем класс с помощью утилиты javac

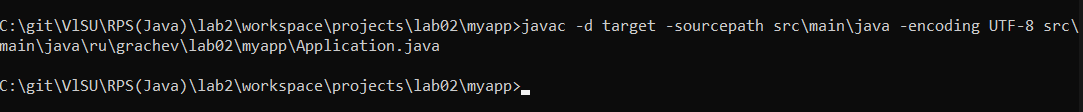


Рисунок . Компиляция класса

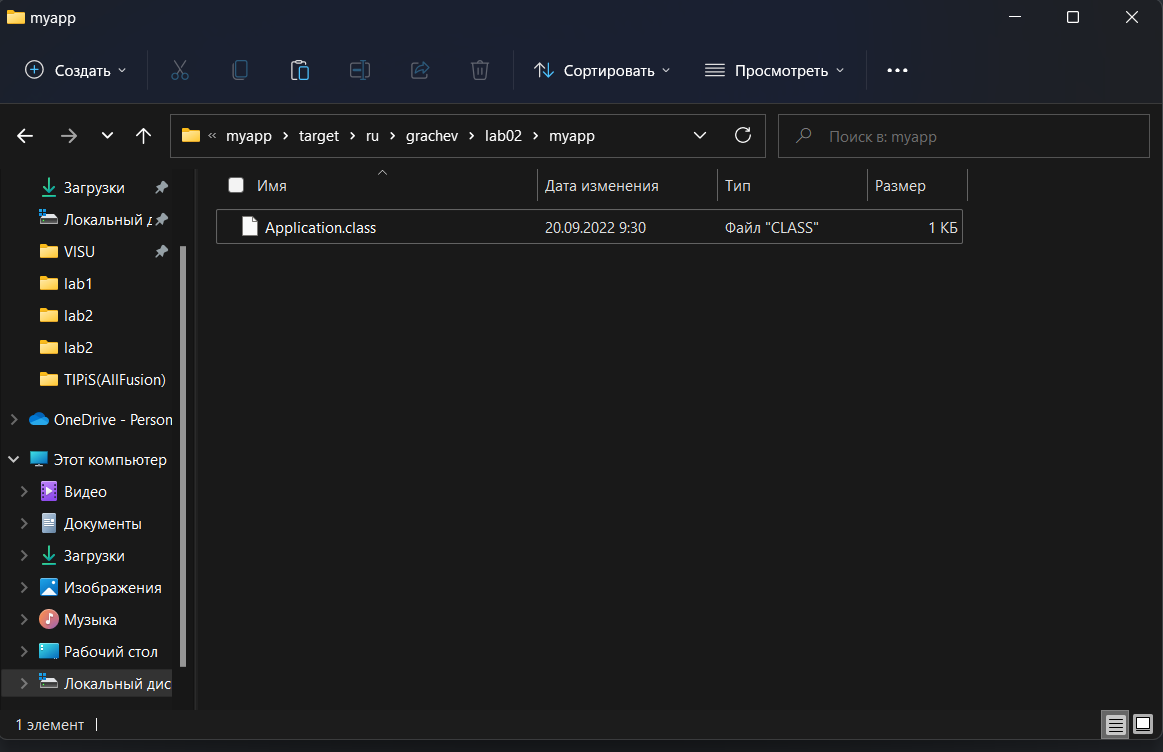


Рисунок . Скомпилированный класс

1. Выполним скомпилированный класс

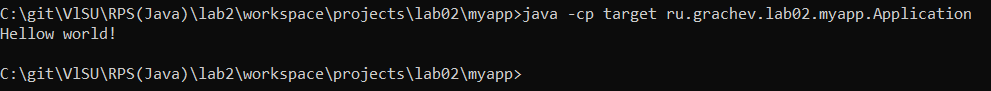
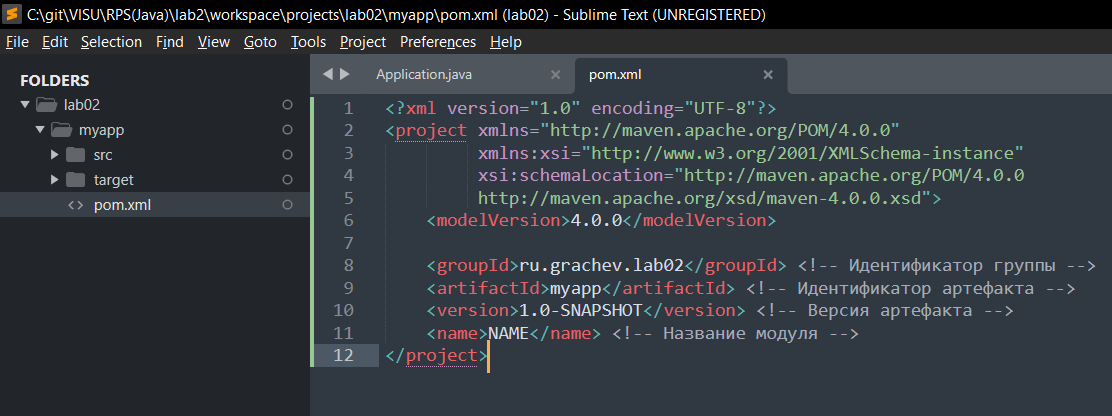


Рисунок . Выполнение скомпилированного класса

1. Создадим файл pom.xml



1. Выполним команду clean для инициализации maven в проекте, удалилась директория target

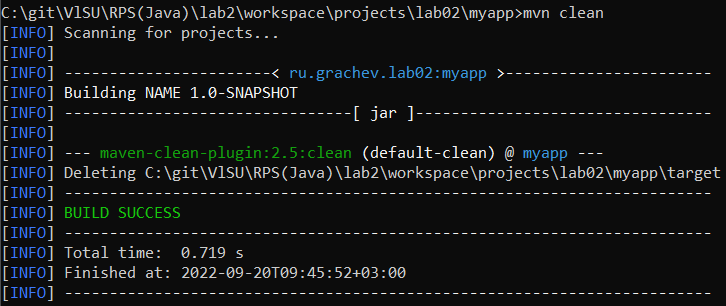


Рисунок 5. mvn clean

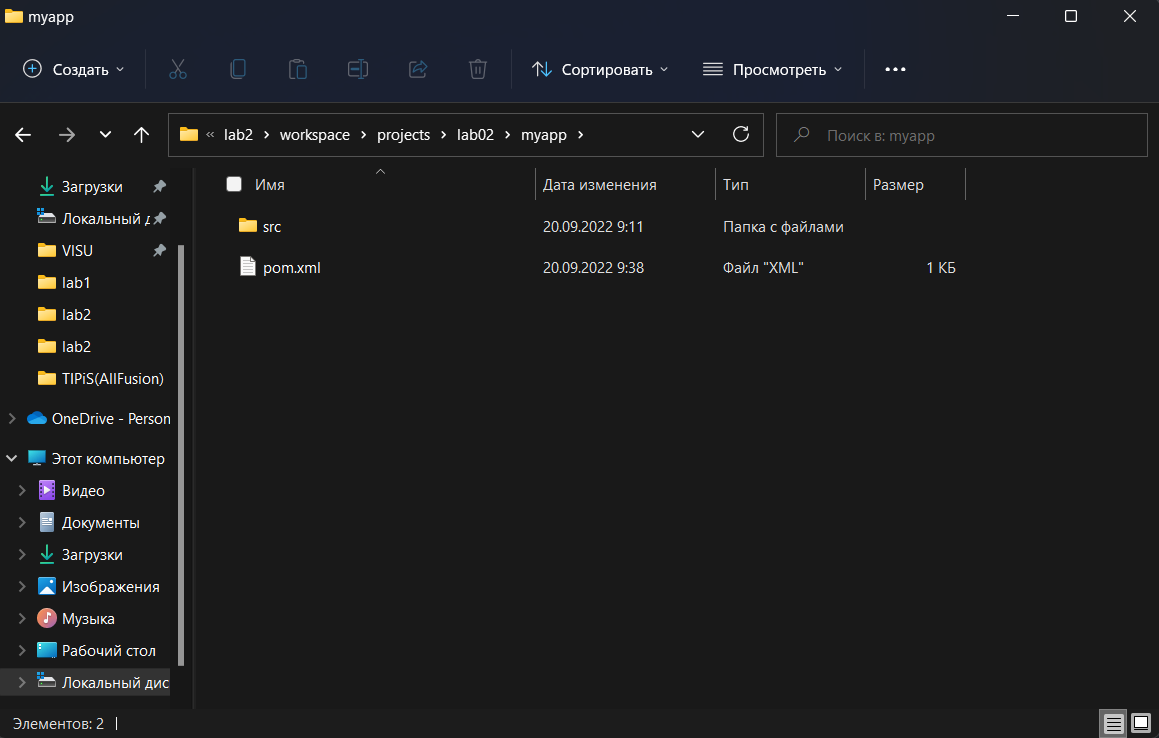


Рисунок 6. Директория без target

1. Сформируем папку приложения

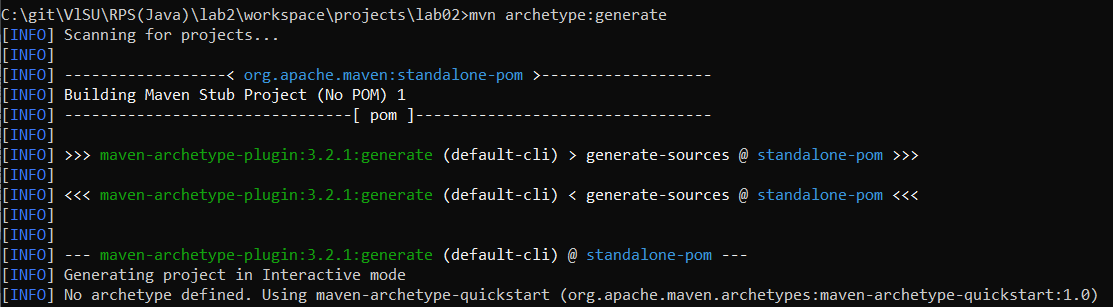
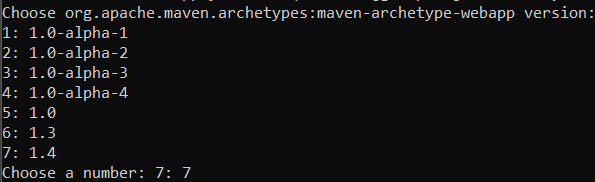


Рисунок 7. Команда







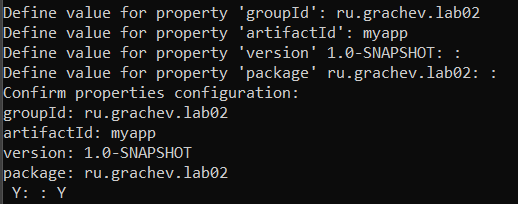


Рисунок 8. Выбор настроек

1. Скомпилируем и выполним проект с помощью mvn compile exec:java

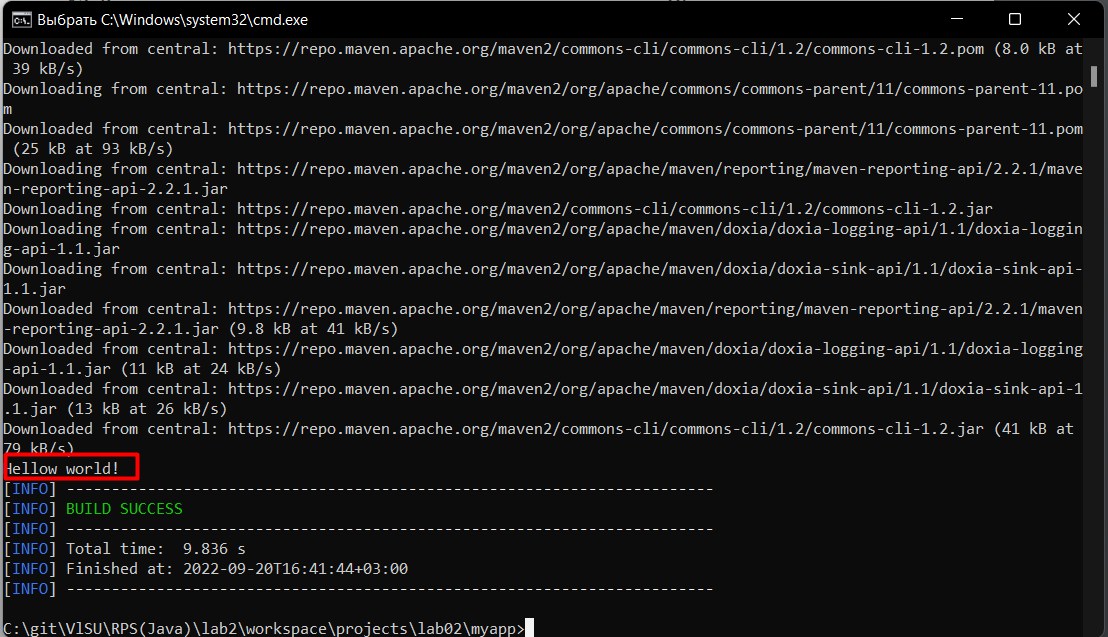


Рисунок 9. Выполнение проекта

1. Добавим класс GreetingsBuilder и используем его

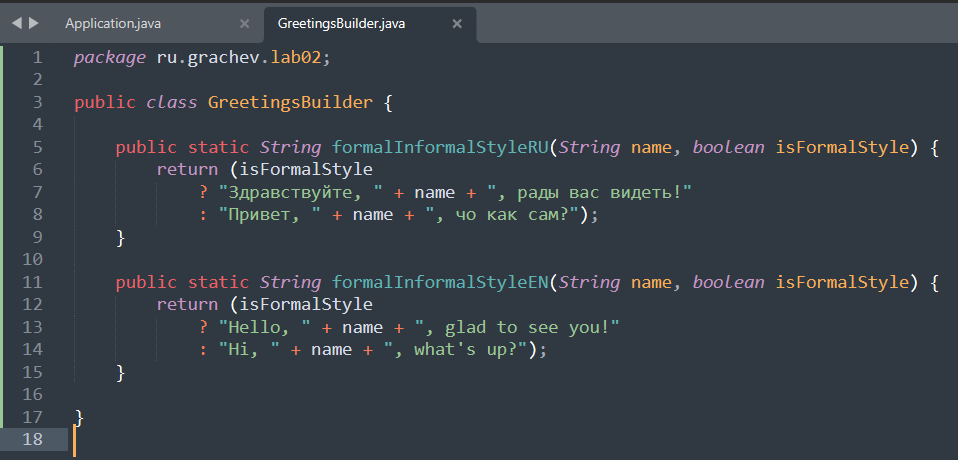


Рисунок 10. Класс GreetingsBuilder

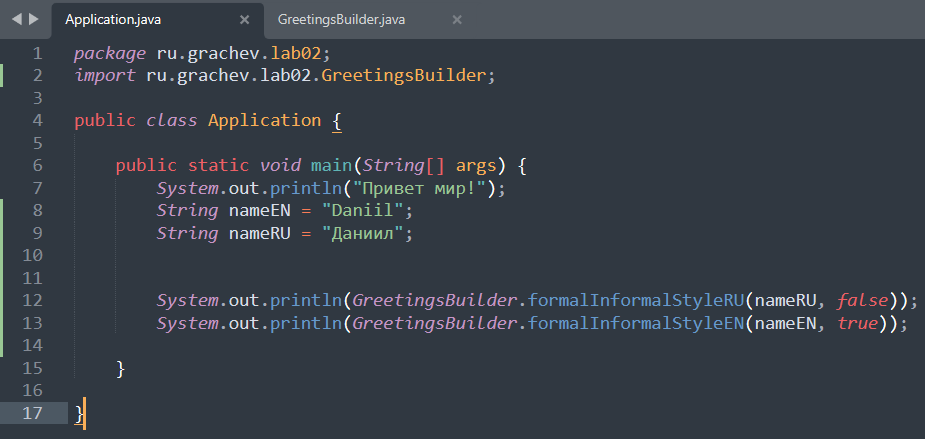


Рисунок 11. Добавление вызовов методов

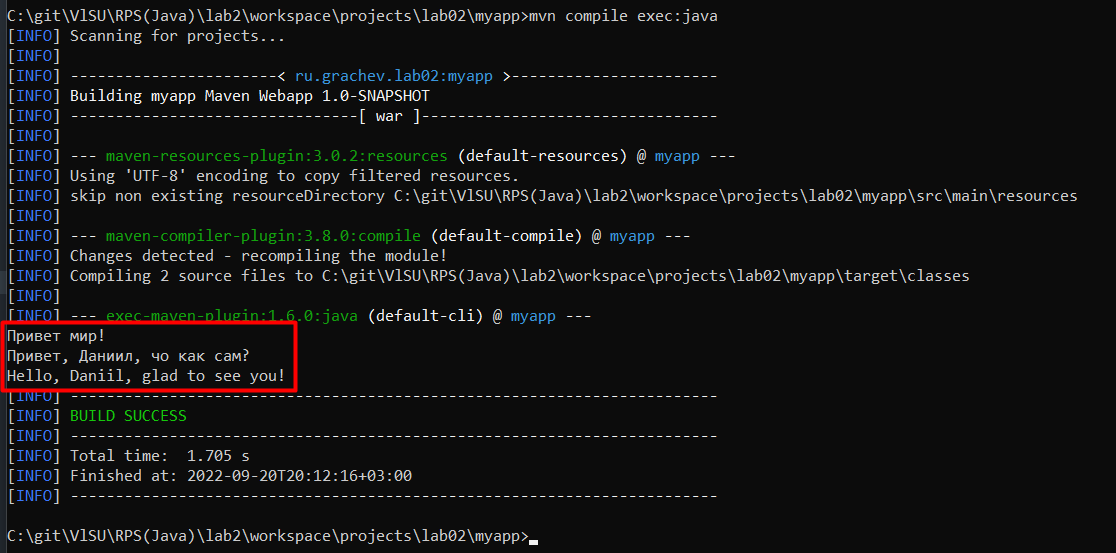


Рисунок 12. Выполнение программы

1. Создадим проект для библиотеки mvn archetype:generate -DarchetypeArtifactId=maven-archetype-quickstart

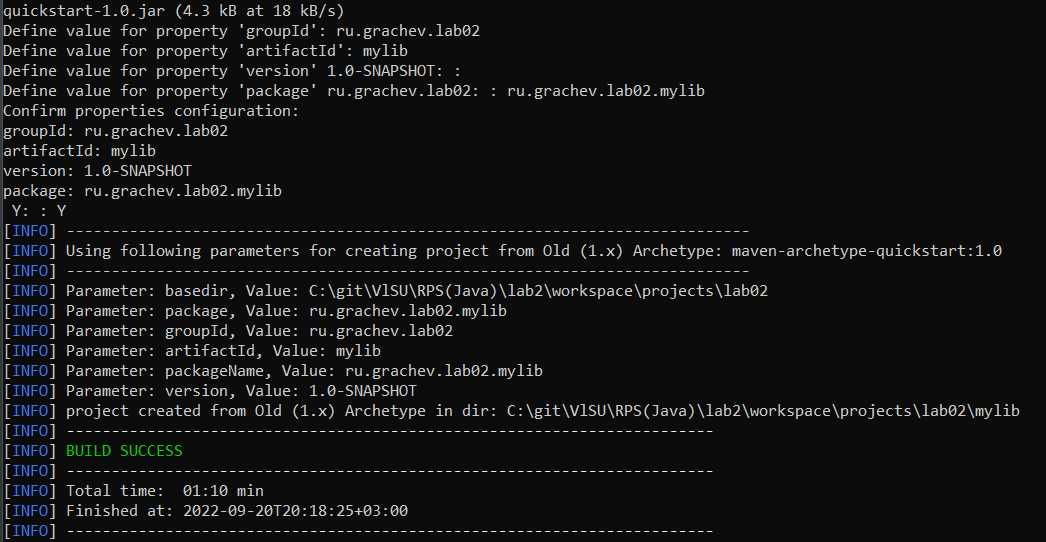


Рисунок 13. Создание проекта для библиотеки

1. Перенесем класс в библиотеку

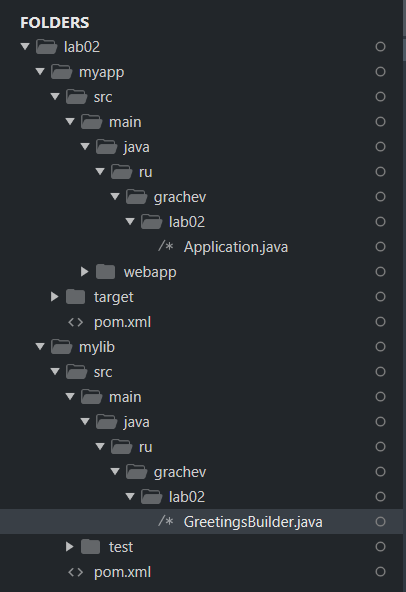


Рисунок 14. Перенос класса

1. Добавим конфигурацию для компилятора

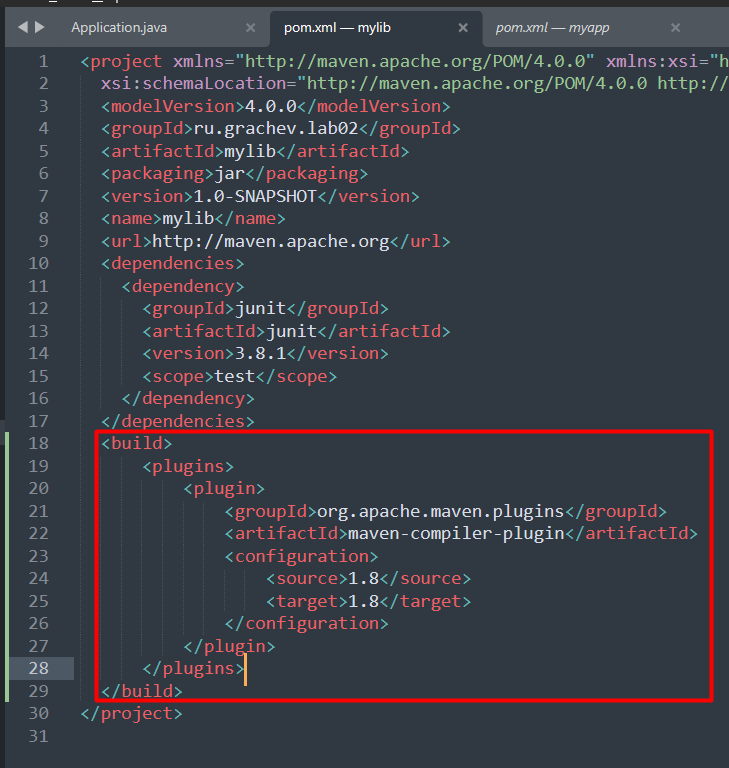


Рисунок 15. Плагин компиляции

1. Проверим, что mylib компилируется

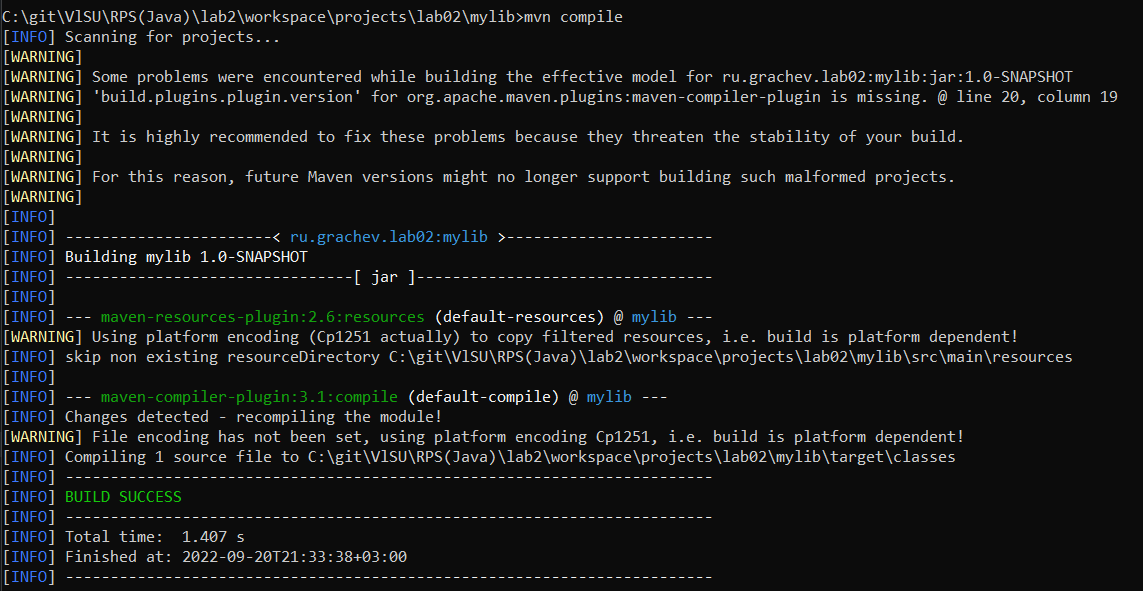


Рисунок 16. Компиляция mylib

1. Убедимся, что myapp больше не компилируется

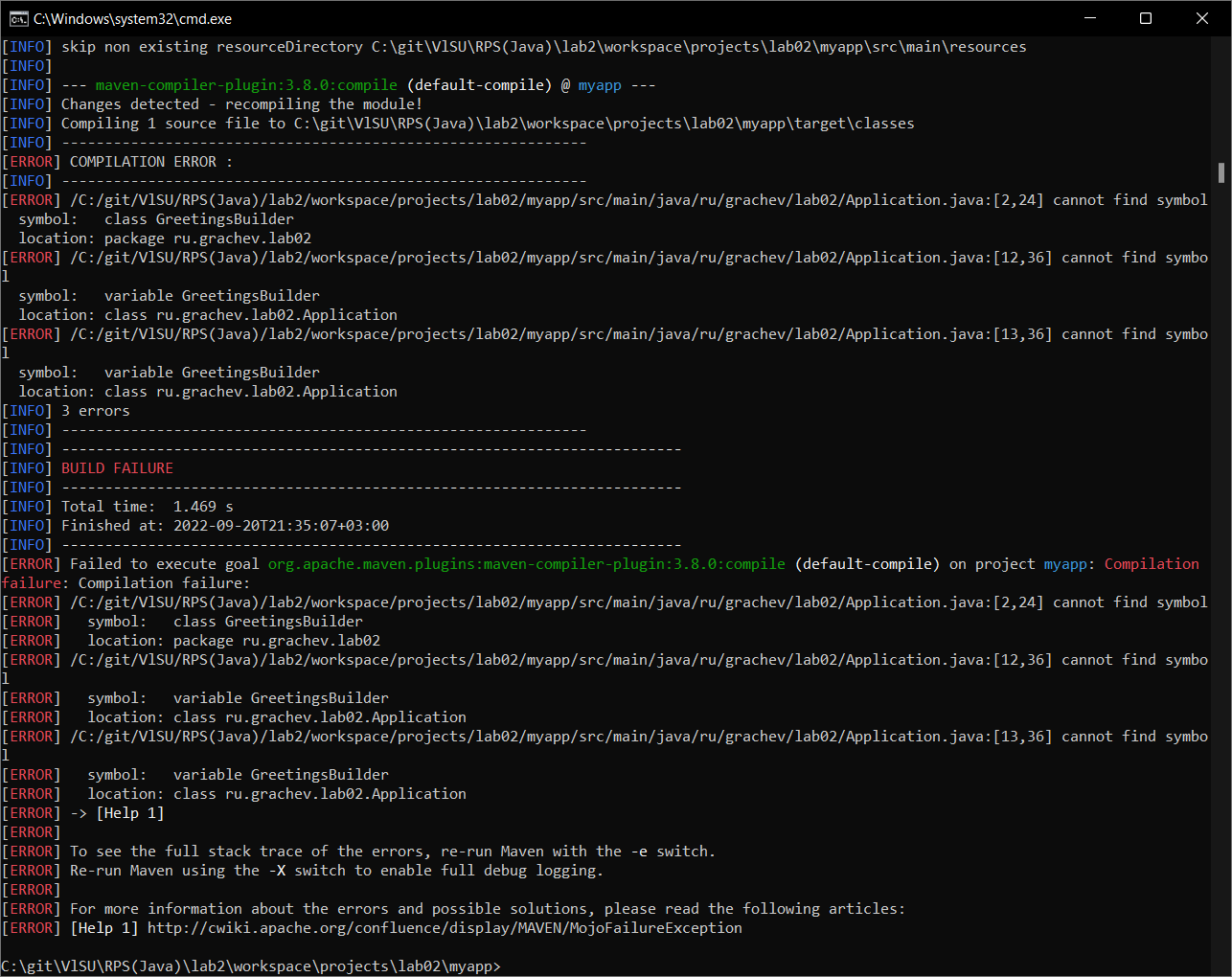


Рисунок 17. Попытка компиляции myapp

1. Установим зависимости в файле pom.xml в проекте myapp

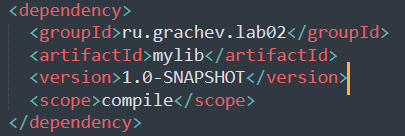


Рисунок 18. Установка зависимостей

1. Выполним команду mvn install для установки библиотеки в локальный репозиторий

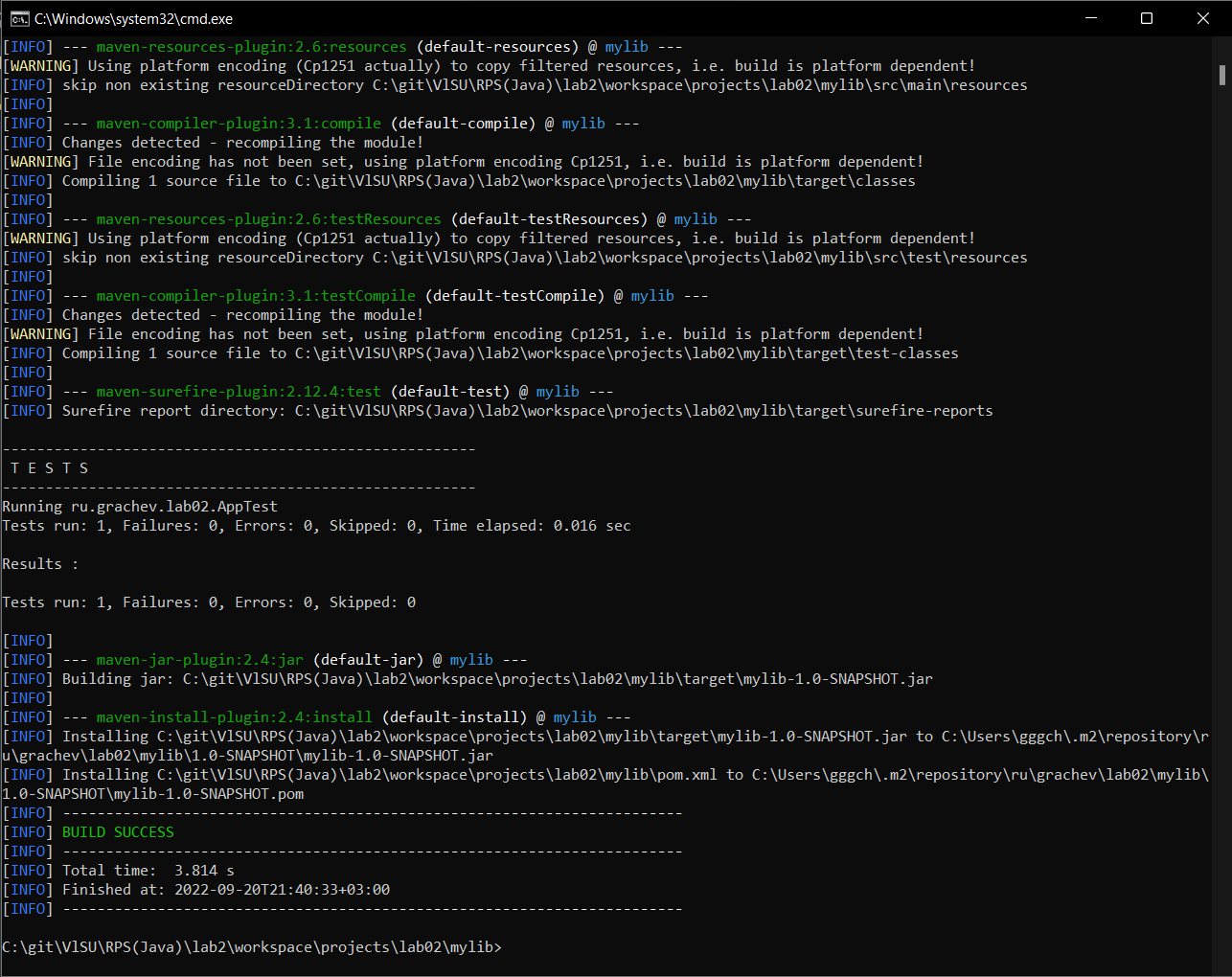


Рисунок 19. mvn install

1. Скомпилируем и запустим проект myapp, теперь все заработало и класс отработал

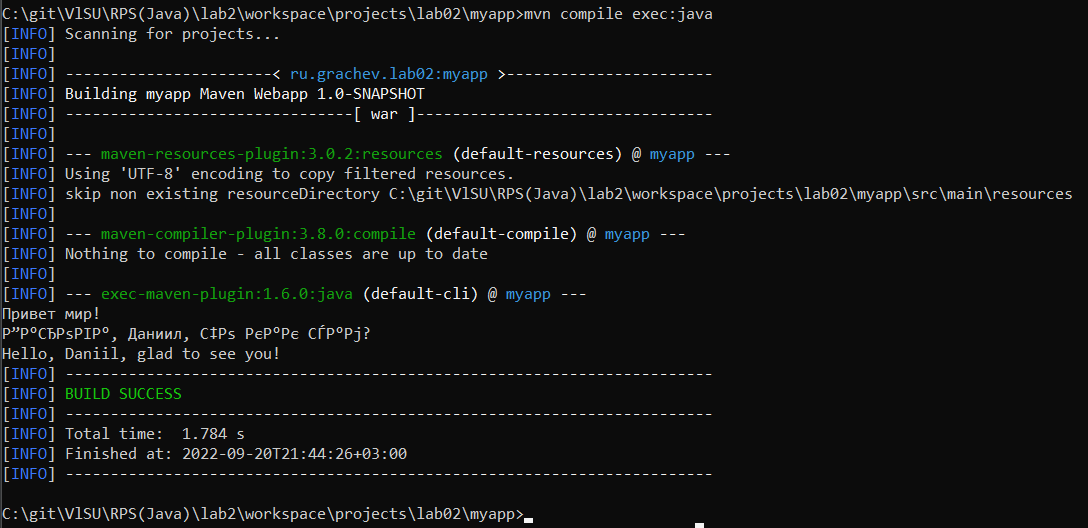


Рисунок 20. Запуск myapp

1. Создадим pom.xml в корневой папке lab02

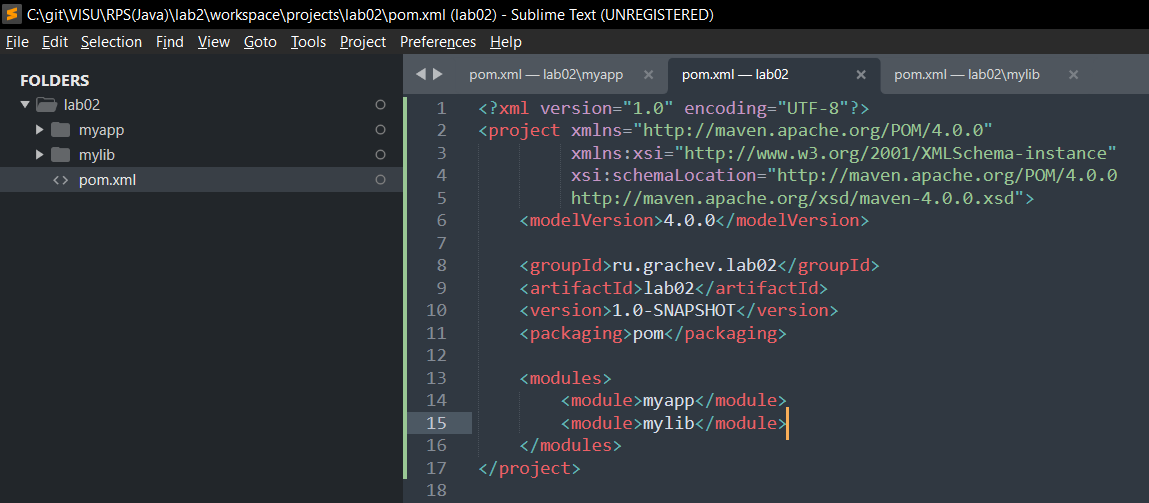


Рисунок 21. pom.xml в корневом каталоге

1. Определим в корневом проекте базовые настройки

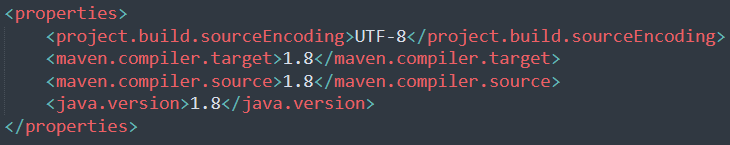
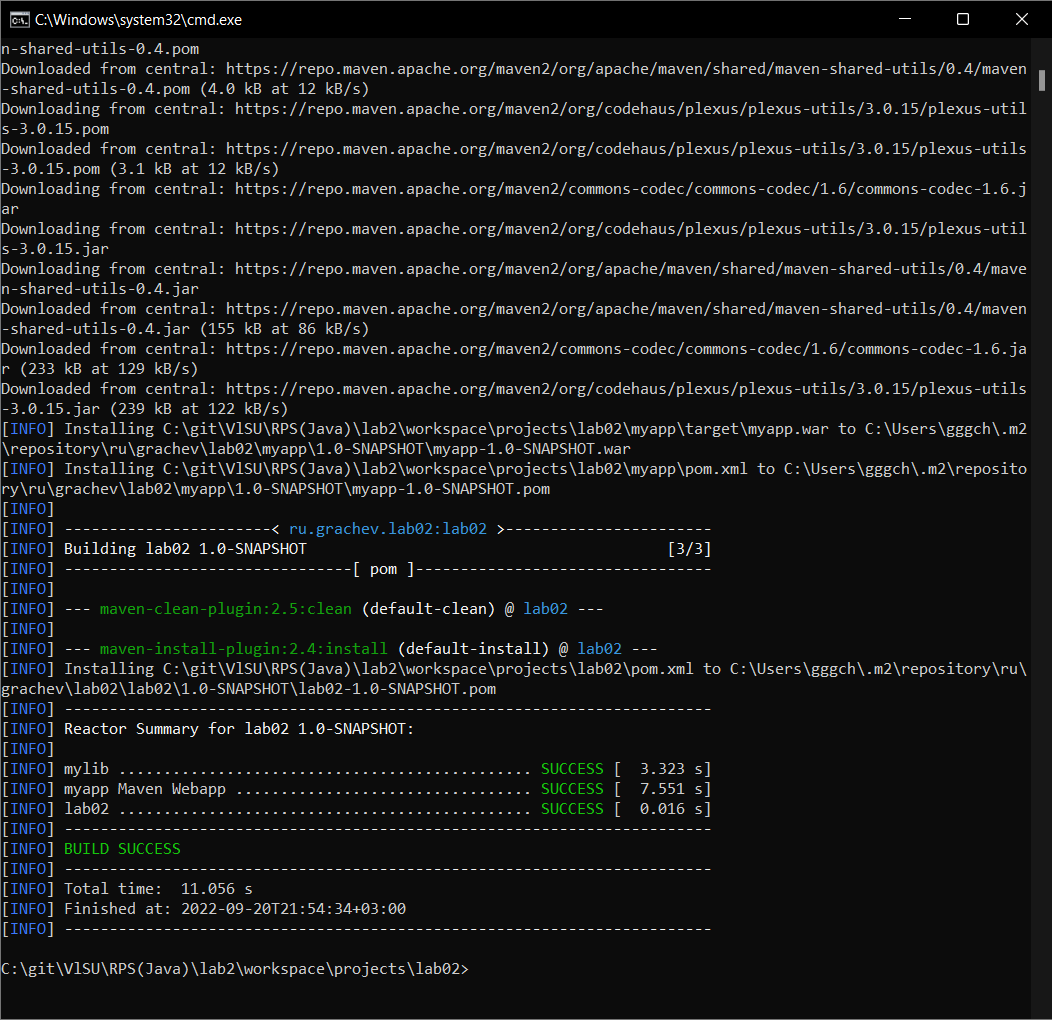


Рисунок 22. Настройки корневого каталога

1. Выполним команду mvn clear install для сборки корневого проекта



1. Убедимся, что проект myapp все еще запускается

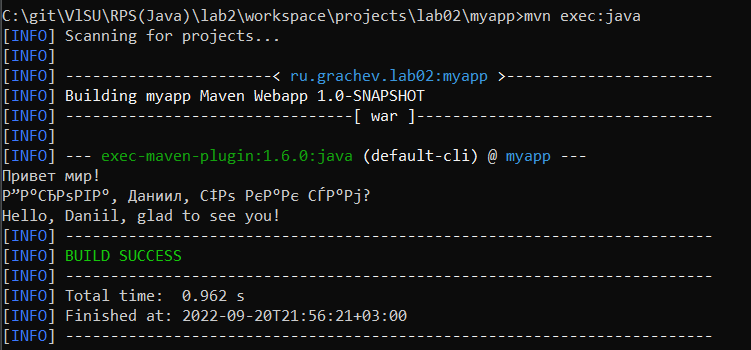


Рисунок 23. Проверка работы myapp

1. Изменим версию Junit на 4.12 в mylib/pom.xml

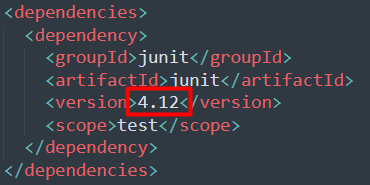


Рисунок 24. Смена версии JUnit

1. Напишем тест на функцию formalInformalStyleEN()

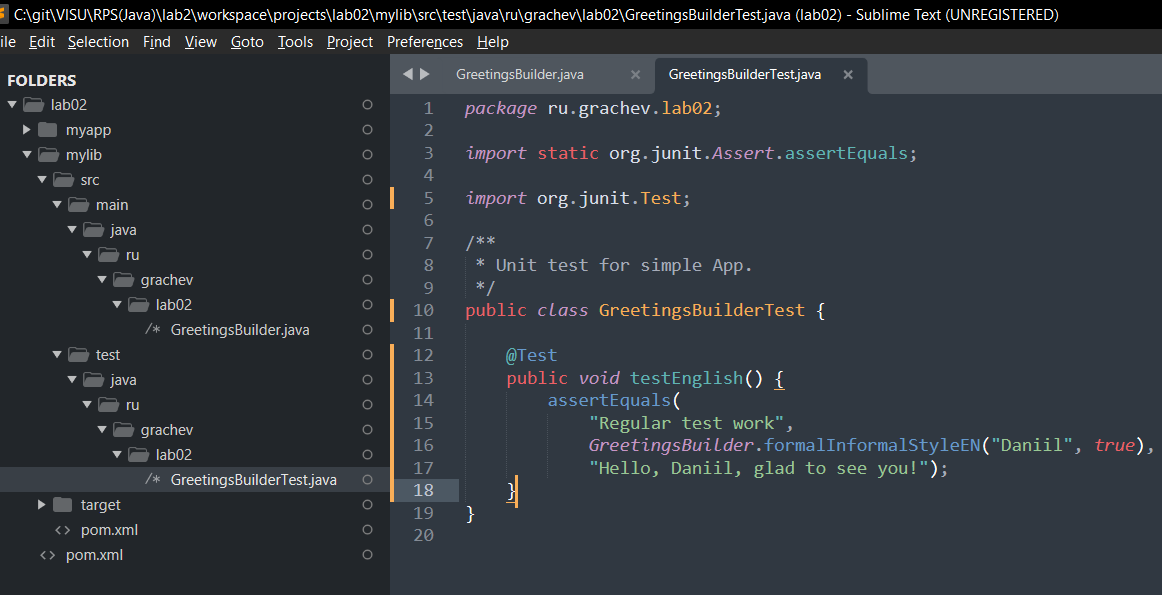


Рисунок 25. Листинг теста

1. Запустим команду mvn test для запуска тестов

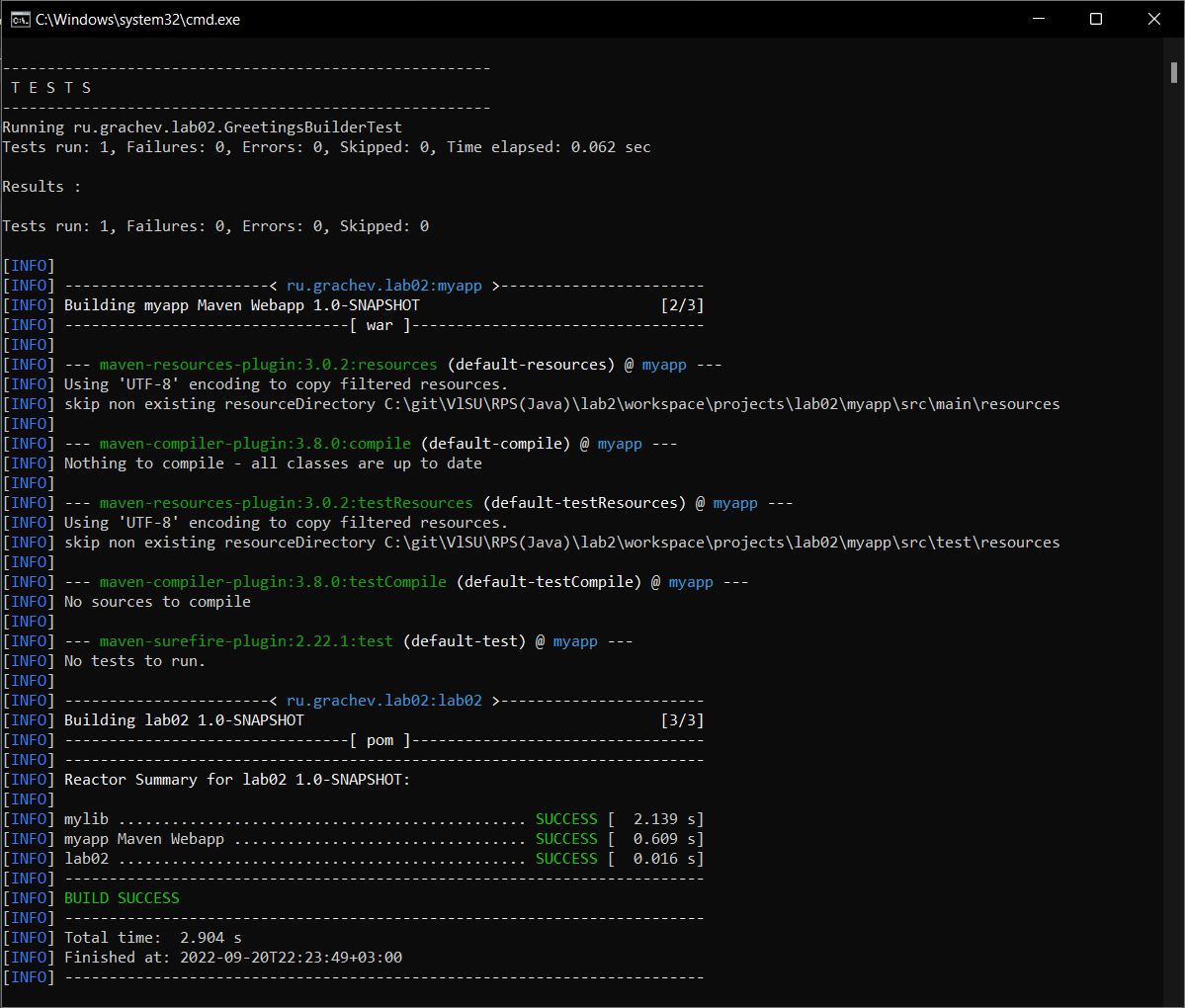


Рисунок 26. Тесты проекта

1. Подключим стороннюю библиотеку с помощью dependency в myapp/pom.xml

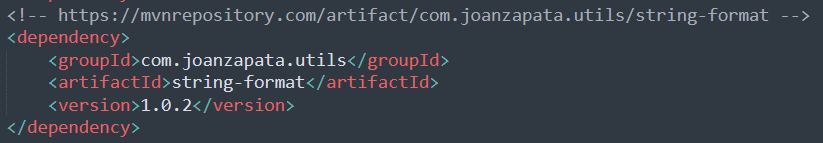


Рисунок 27. Подключенная библиотека

1. Используем подключенную библиотеку



Рисунок 28. Использование библиотеки

1. Работа библиотеки при выполнении

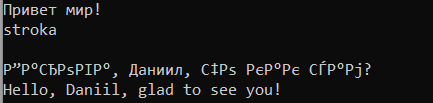


Рисунок 29. Выполнение программы

1. Добавим профили в myapp/pom.xml



Рисунок 30. Созданные профили

1. Создадим 2 разных класса для русского и английского вывода

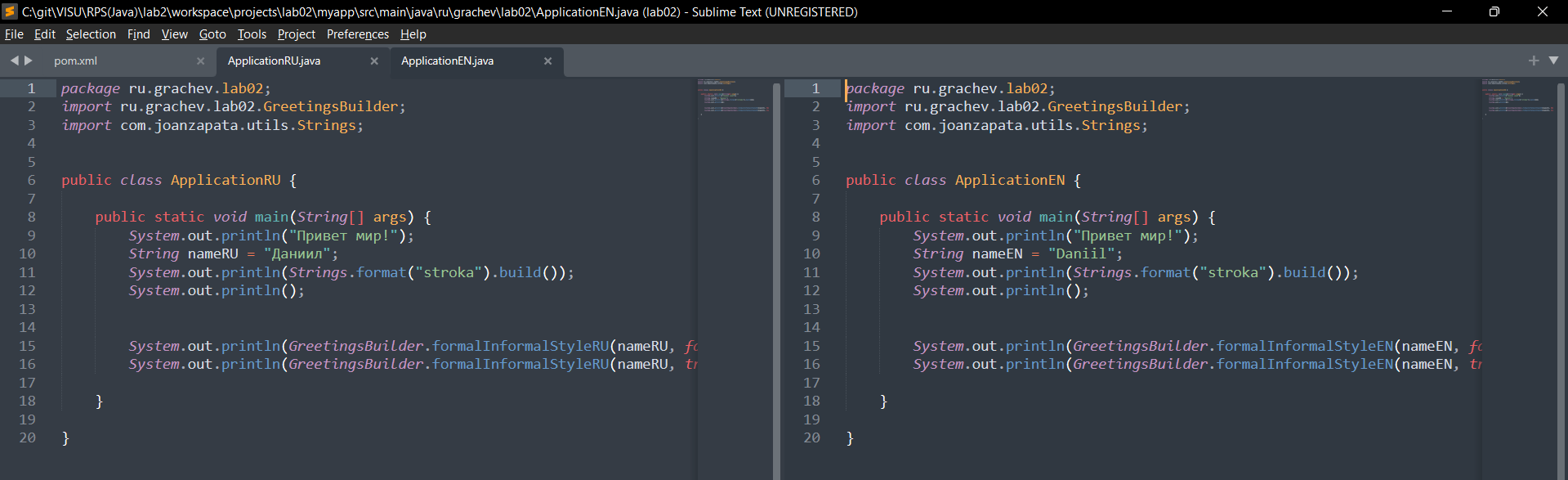


Рисунок 31. Разделенные классы

1. Добавим зависимость checkstyle



Рисунок 32. Зависимость checkstyle

1. Добавим плагин для активации проверки стиля кода

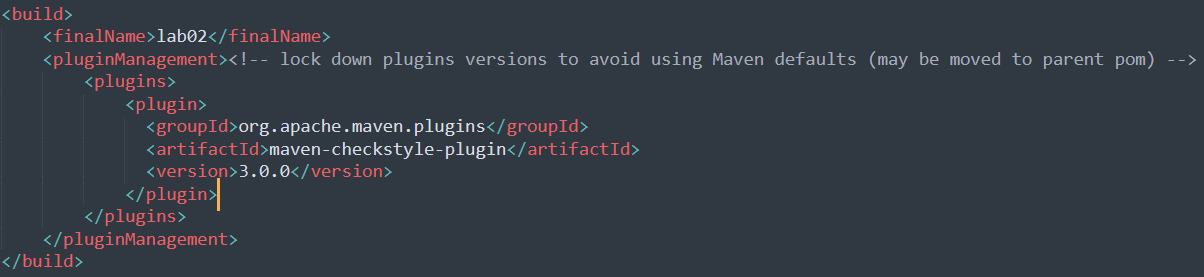


Рисунок 33. Плагин проверки стиля кода

1. Выполним команды mvn checkstyle:check, чтобы увидеть ошибки в коде

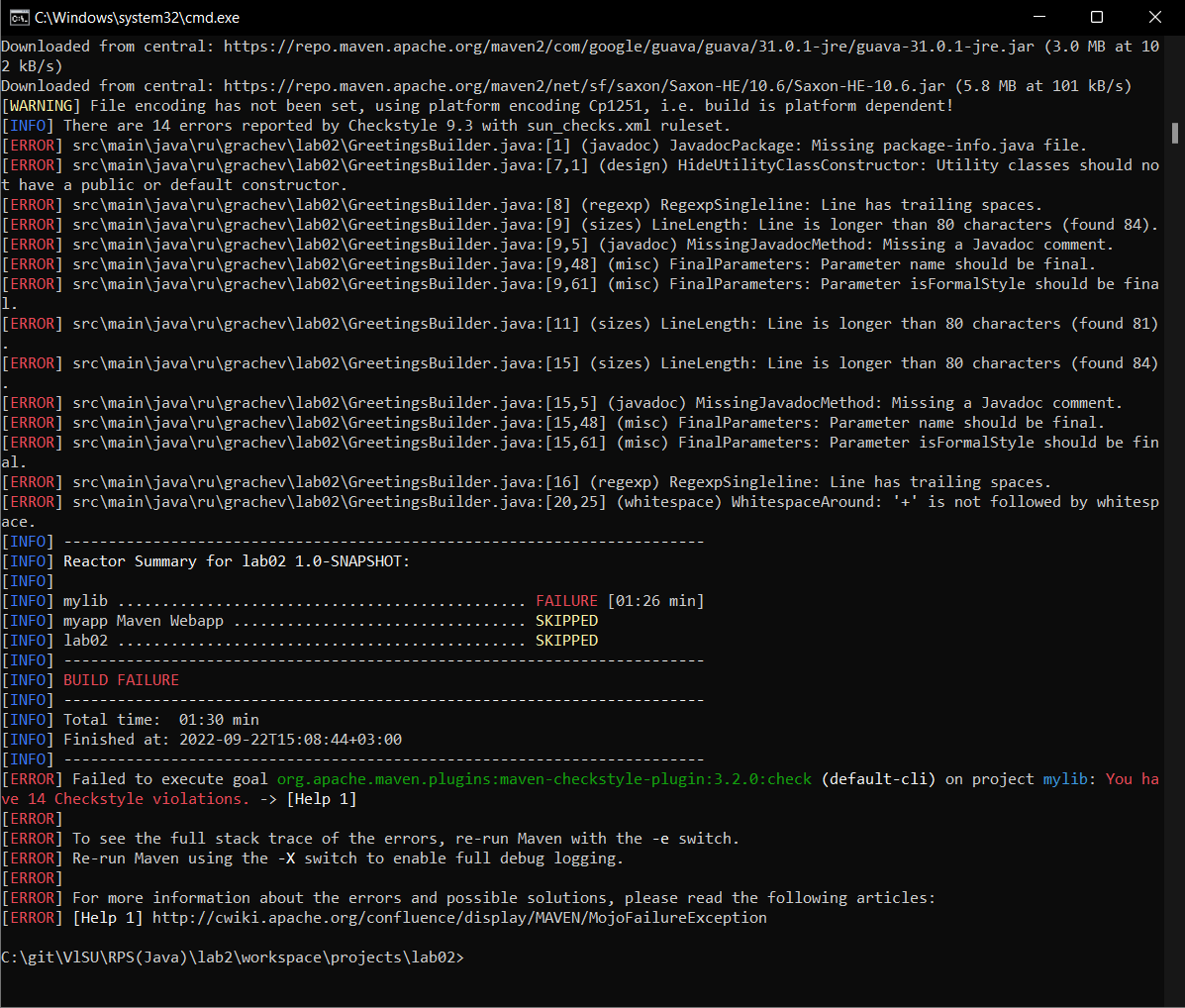


Рисунок 34. nvm checkstyle:check

1. Создадим конфигурацию запуска в IDE

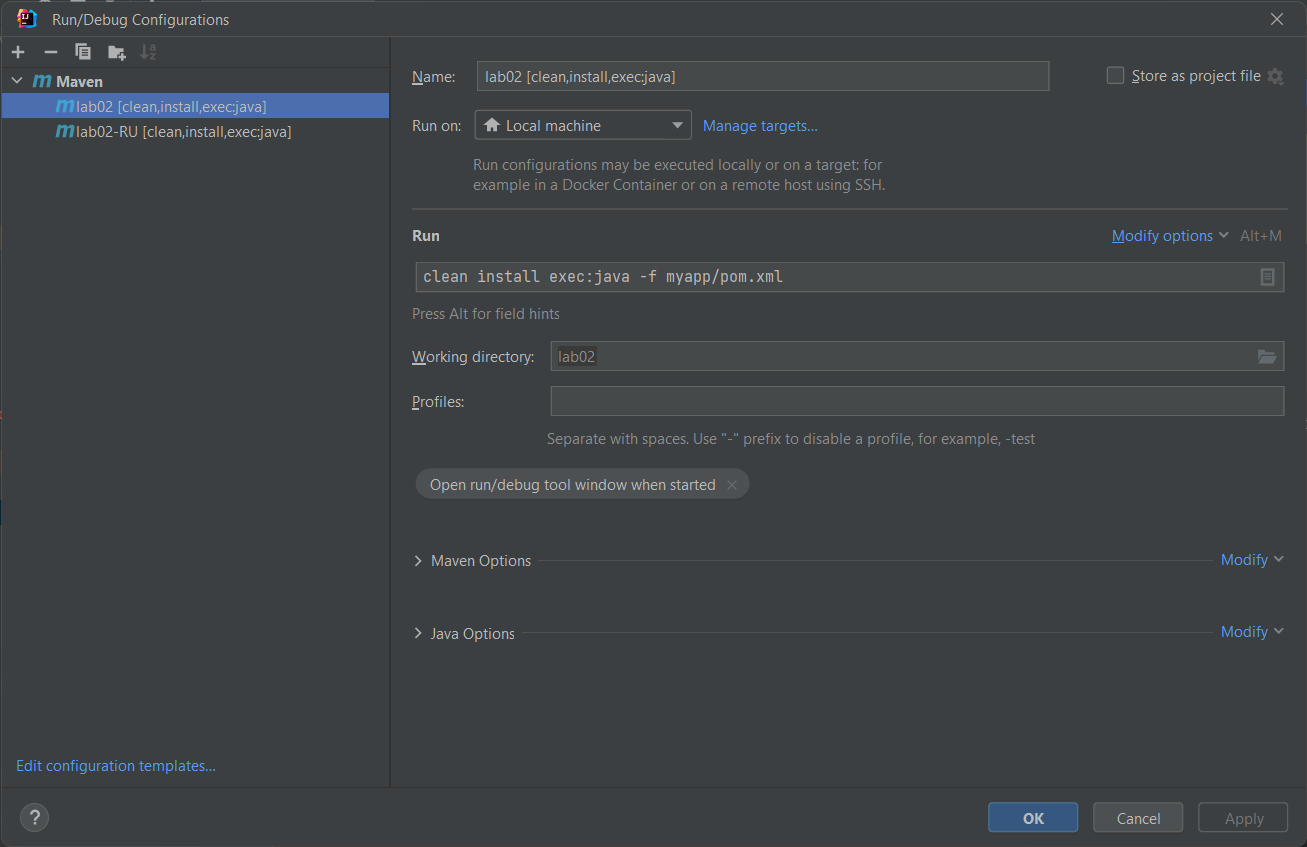


Рисунок 35. Конфигурация запуска в IDE

1. Добавим конфигурацию запуска с русским языком

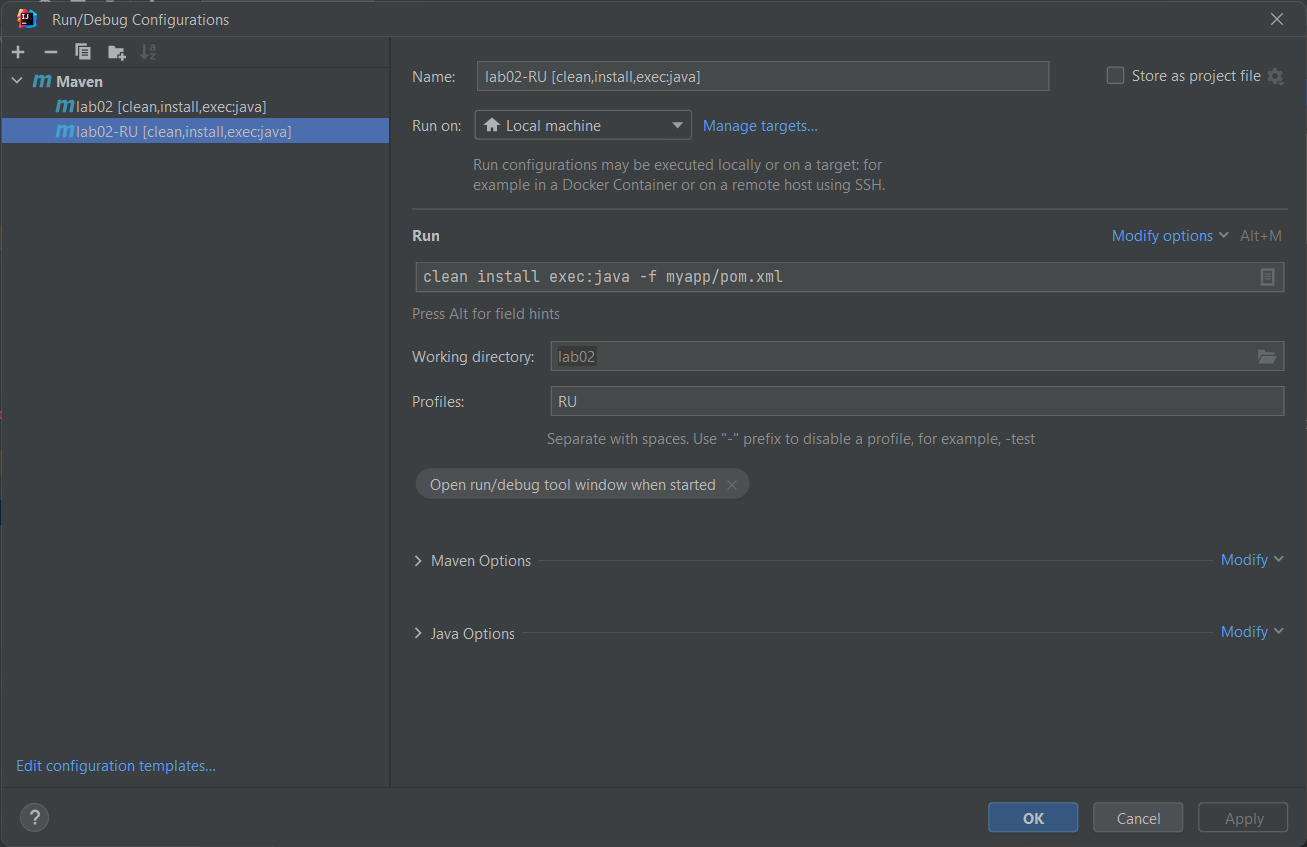


Рисунок 36. Конфигурация запуска с русским языком

1. Запустим проект в IDE

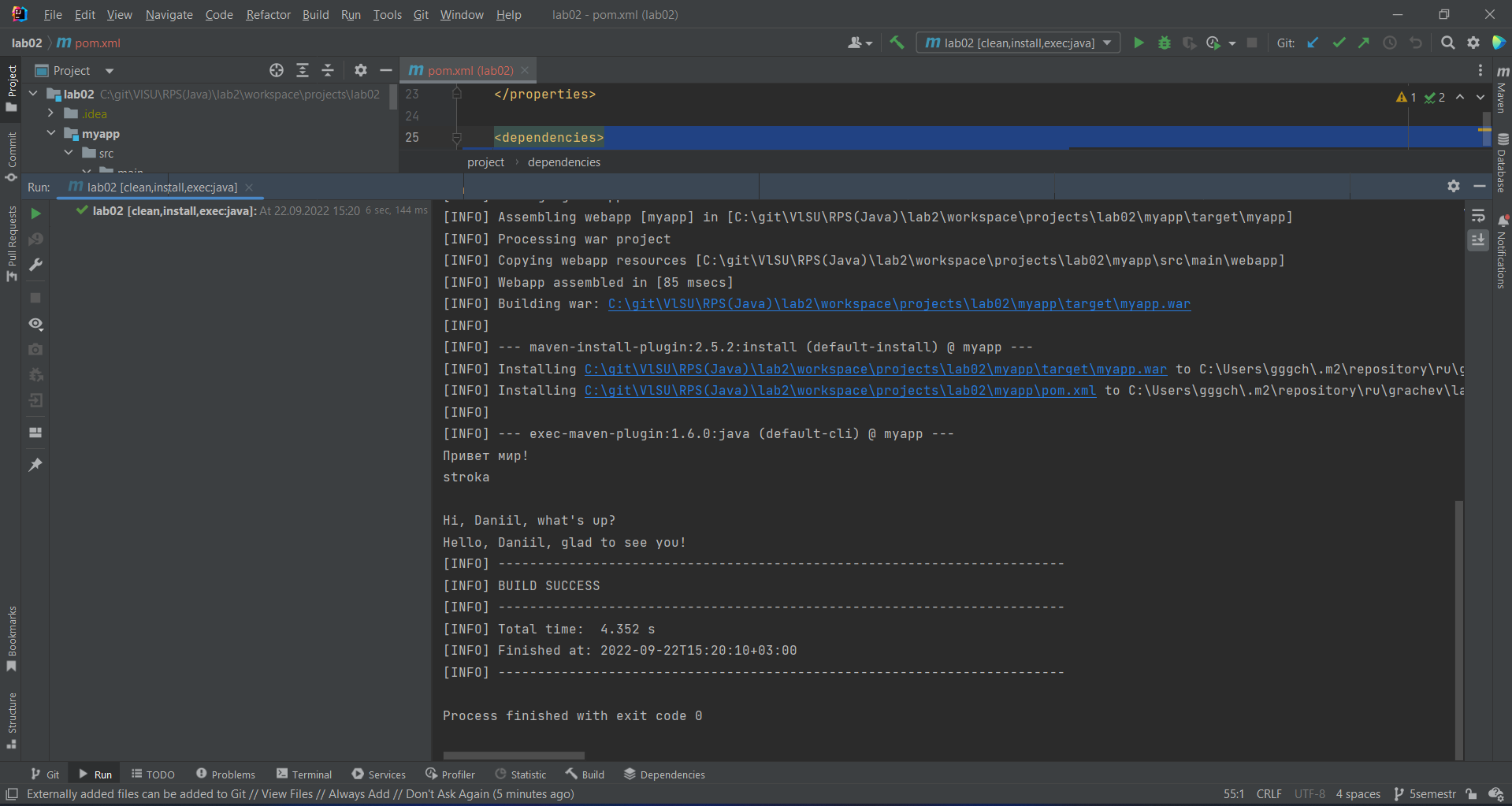


Рисунок 37. Работа проекта в IDE

ВЫВОД

В ходе выполнения лабораторной работы познакомился с принципами разработки Java-приложений с использованием автоматизированной системы управления жизненным циклом проекта Apache Maven, получил практические навыки решения типовых задач при сборке Java проекта.