Configure and Setup IntelliJ IDE for Spring

Estimated time needed: 15 minutes

Overview

In this lab, you will be setting up the IntelliJ Community Edition IDE for working on Spring projects.

Learning objectives

After completing this lab, you will be able to:

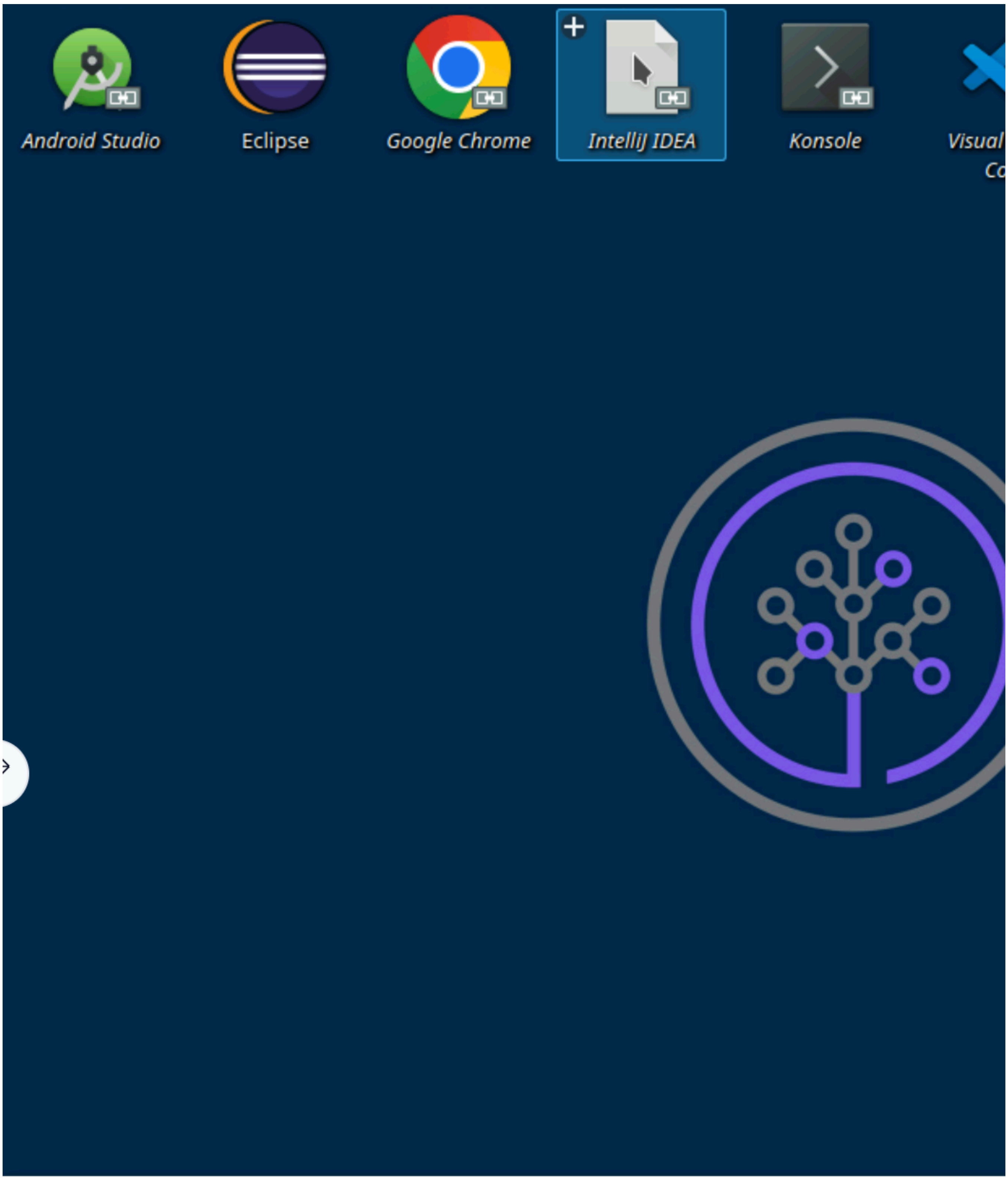
- Create a maven project in IntelliJ Community Edition
- Configure `project` settings
- Configure the `module` settings
- Create class in the source folder
- Build the project
- Run the project and view the output on the terminal

Prerequisites

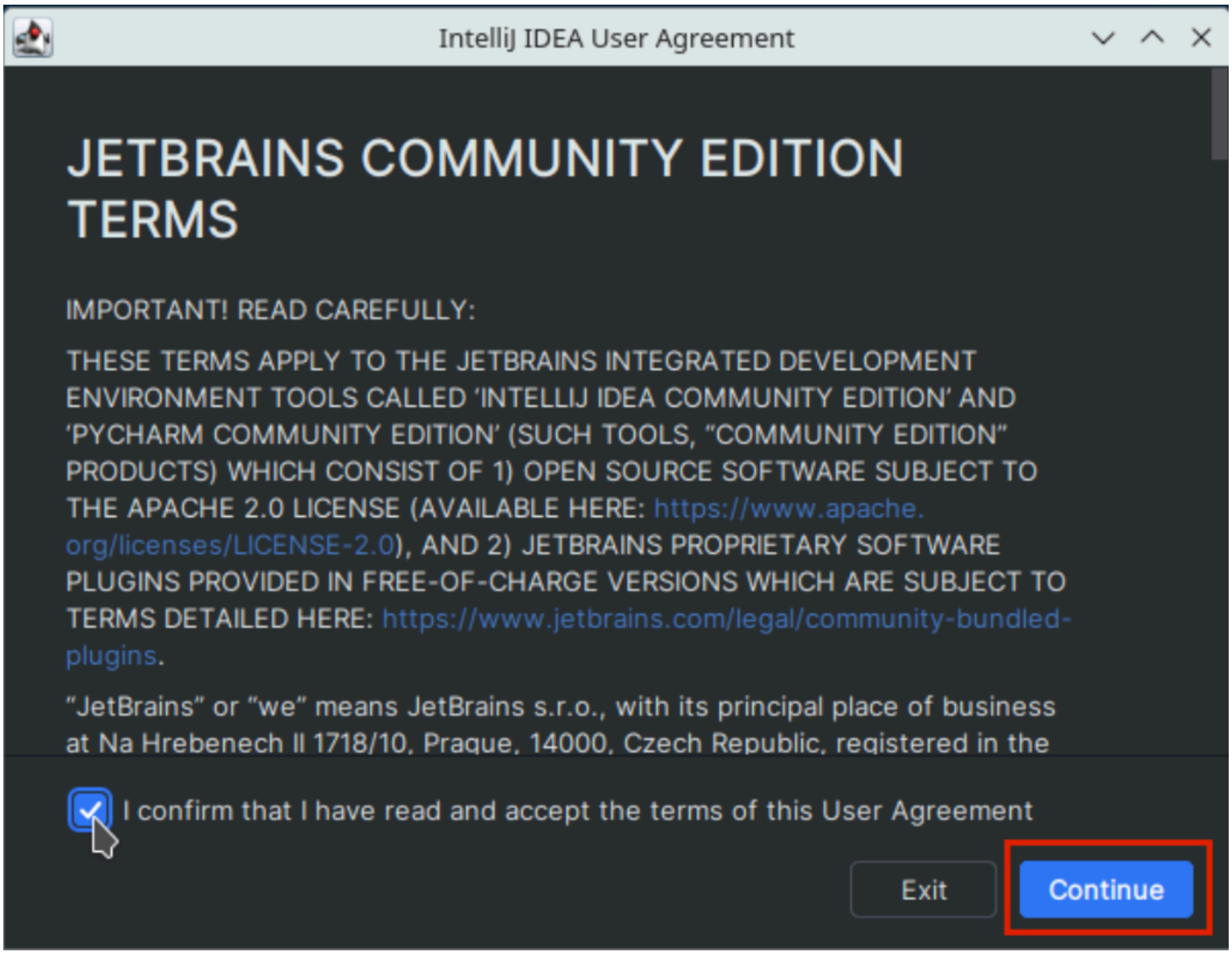
You should know basic Java programming before you get started with this lab.

Create a New Maven project in IntelliJ

1. Open the IntelliJ IDE in the lab environment provided.



2. A `User Agreement` containing the terms and conditions for using the IntelliJ community edition will be displayed. Agree by selecting the checkbox and clicking `Continue`.


IntelliJ IDEA User Agreement

JETBRAINS COMMUNITY EDITION TERMS

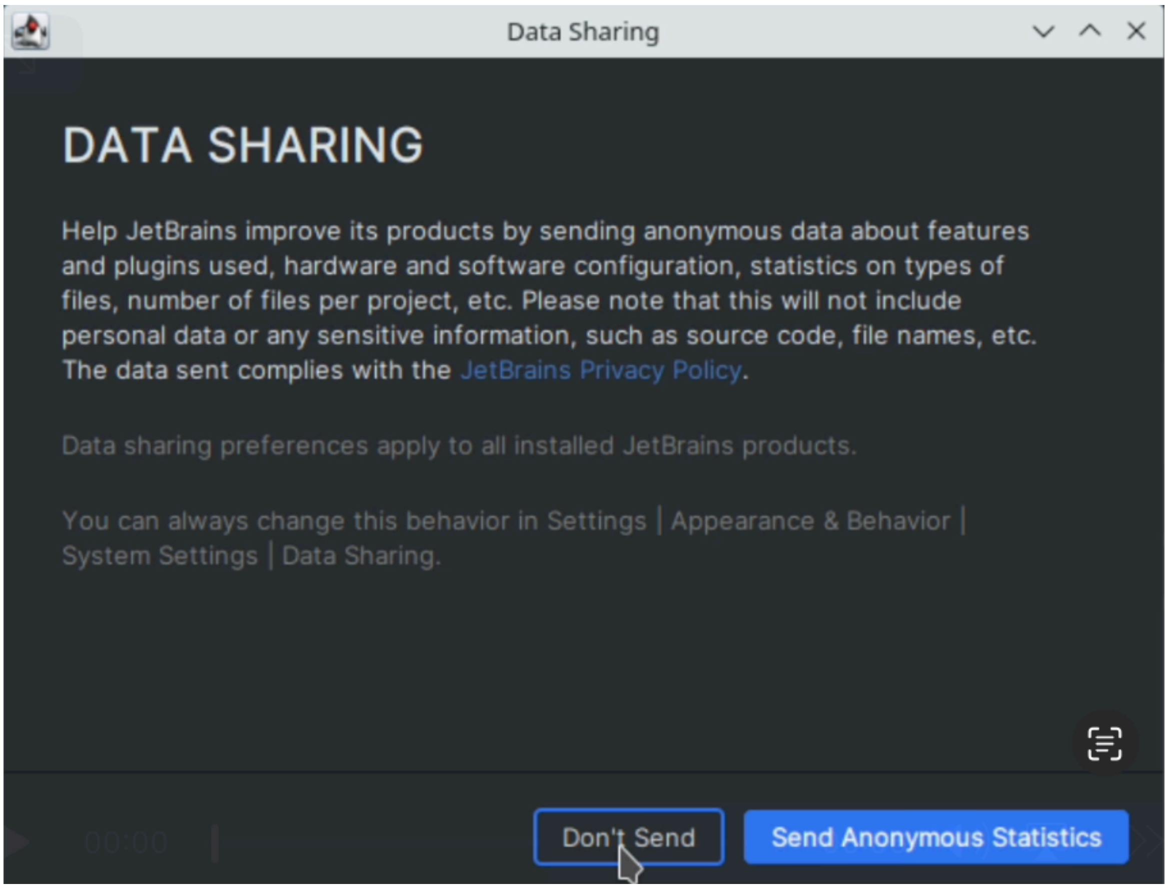
IMPORTANT! READ CAREFULLY:

THESE TERMS APPLY TO THE JETBRAINS INTEGRATED DEVELOPMENT ENVIRONMENT TOOLS CALLED 'INTELLIJ IDEA COMMUNITY EDITION' AND 'PYCHARM COMMUNITY EDITION' (SUCH TOOLS, "COMMUNITY EDITION" PRODUCTS) WHICH CONSIST OF 1) OPEN SOURCE SOFTWARE SUBJECT TO THE APACHE 2.0 LICENSE (AVAILABLE HERE: https://www.apache.org/licenses/LICENSE-2.0), AND 2) JETBRAINS PROPRIETARY SOFTWARE PLUGINS PROVIDED IN FREE-OF-CHARGE VERSIONS WHICH ARE SUBJECT TO TERMS DETAILED HERE: https://www.jetbrains.com/legal/community-bundled-plugins.

"JetBrains" or "we" means JetBrains s.r.o., with its principal place of business at Na Hrebenech II 1718/10, Prague, 14000, Czech Republic, registered in the

[✓] I confirm that I have read and accept the terms of this User Agreement

3. A window asking if you want to share your data on usage statistics will be displayed. Click `Don't Send` option.


Data Sharing

DATA SHARING

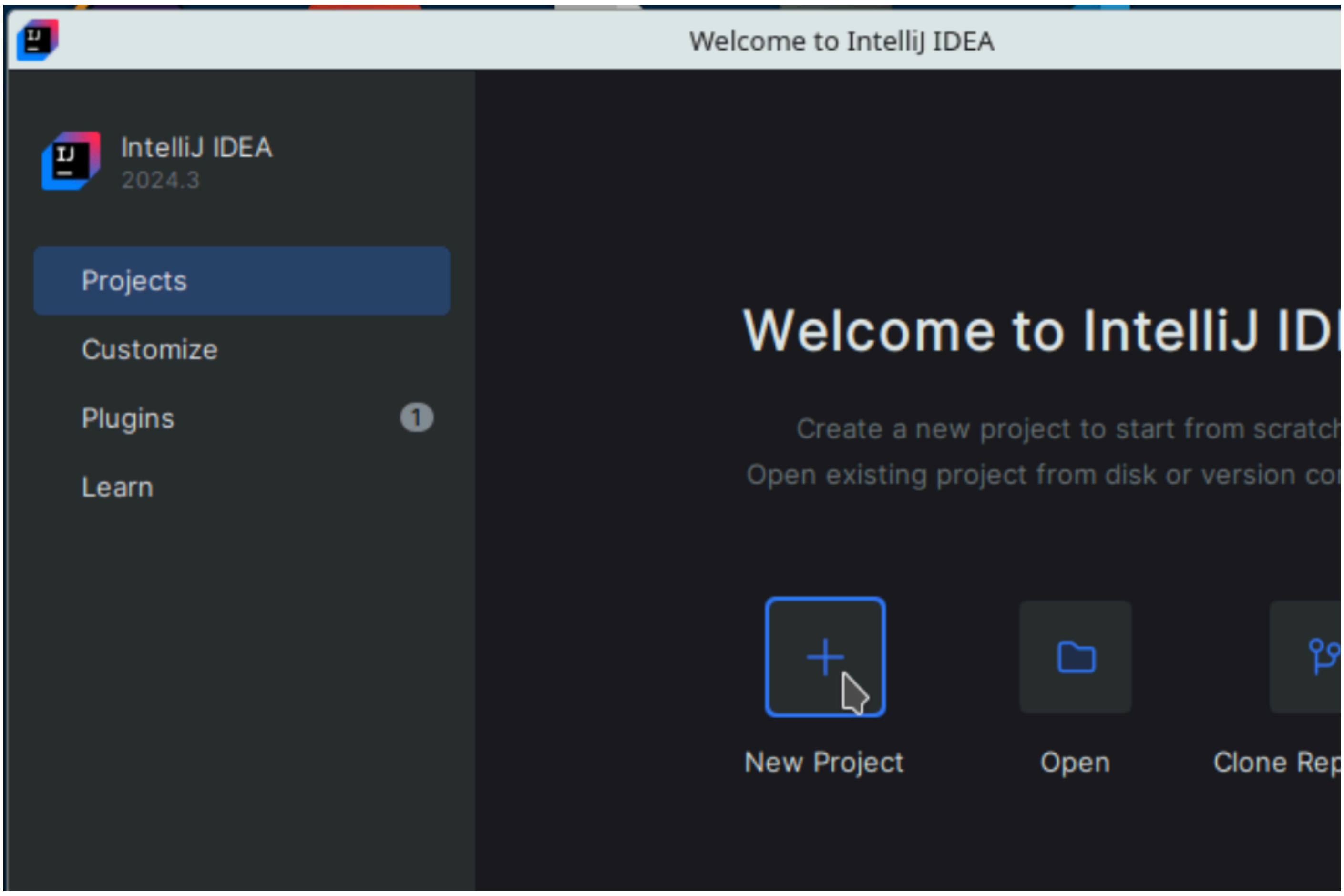
Help JetBrains improve its products by sending anonymous data about features and plugins used, hardware and software configuration, statistics on types of files, number of files per project, etc. Please note that this will not include personal data or any sensitive information, such as source code, file names, etc. The data sent complies with the JetBrains Privacy Policy.

Data sharing preferences apply to all installed JetBrains products.

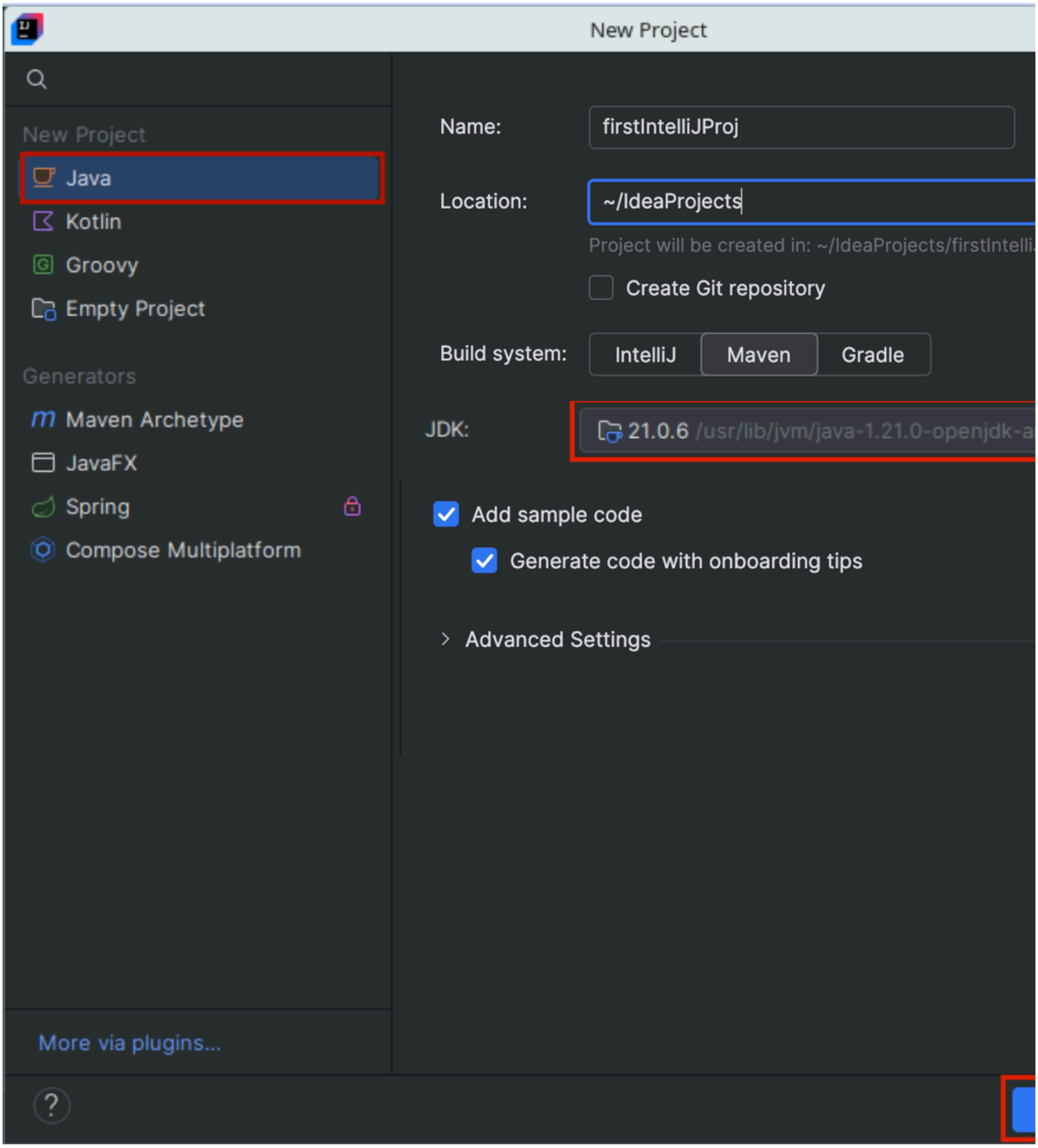
You can always change this behavior in Settings | Appearance & Behavior | System Settings | Data Sharing.

4. IntelliJ will open and provide you the options for creating a new project, opening an existing project from the limux environment, or cloning from a repository.

Choose the option to create a new project.



5. Ensure that the `Java` option is chosen in the left panel. Enter the name of the project in the Name text field, for example `firstIntelliJProj` as shown in the image, choose the `Maven` option in the Build system field, choose the JDK version (*21.0.6, that's installed in the lab environment*), and click `Create`.


New Project

Name: firstIntelliJProj

Location: ~/IdeaProjects

Project will be created in: ~/IdeaProjects/firstIntelli

Create Git repository

Build system: IntelliJ Maven Gradle

JDK: 21.0.6 /usr/lib/jvm/java-1.21.0-openjdk-a

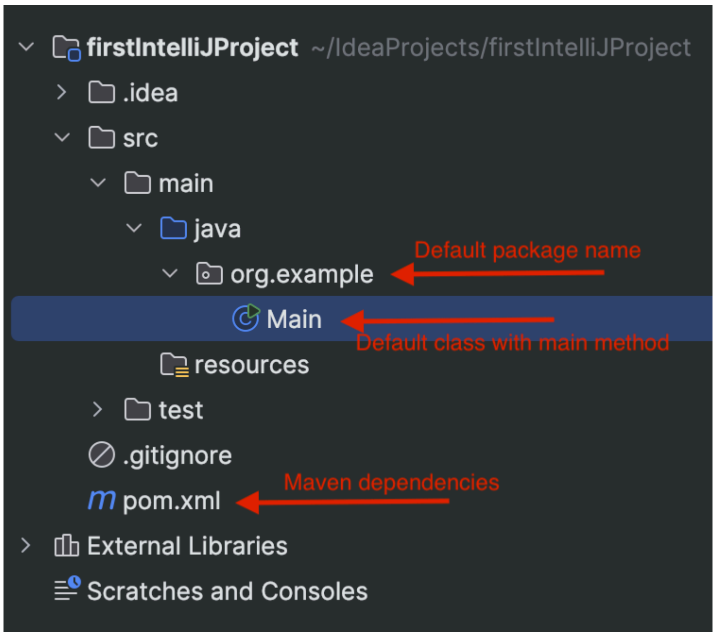
Add sample code

Generate code with onboarding tips

Your project named `firstIntelliJProj` would have now been created.

View the project structure

1. The project structure is generated by default.



2. The `package name` or the `artifact id` for the generated project is `org.example`. You can change these default values as per requirement.
3. The `org.example.Main` class, with the `main` method containing the code below, is also generated.

```
package org.example;
public class Main {
    public static void main(String[] args) {
        System.out.printf("Hello and welcome!");
        for (int i = 1; i <= 5; i++) {
            System.out.println("i = " + i);
        }
    }
}
```

4. The `pom.xml` that is generated is a fundamental part of Maven. It stands for Project Object Model and is an XML file that contains information about the project and configuration details used by Maven to build the project.

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>org.example</groupId>
    <artifactId>firstIntelliJProject</artifactId>
    <version>1.0-SNAPSHOT</version>
</project>
```

Key Uses of pom.xml:

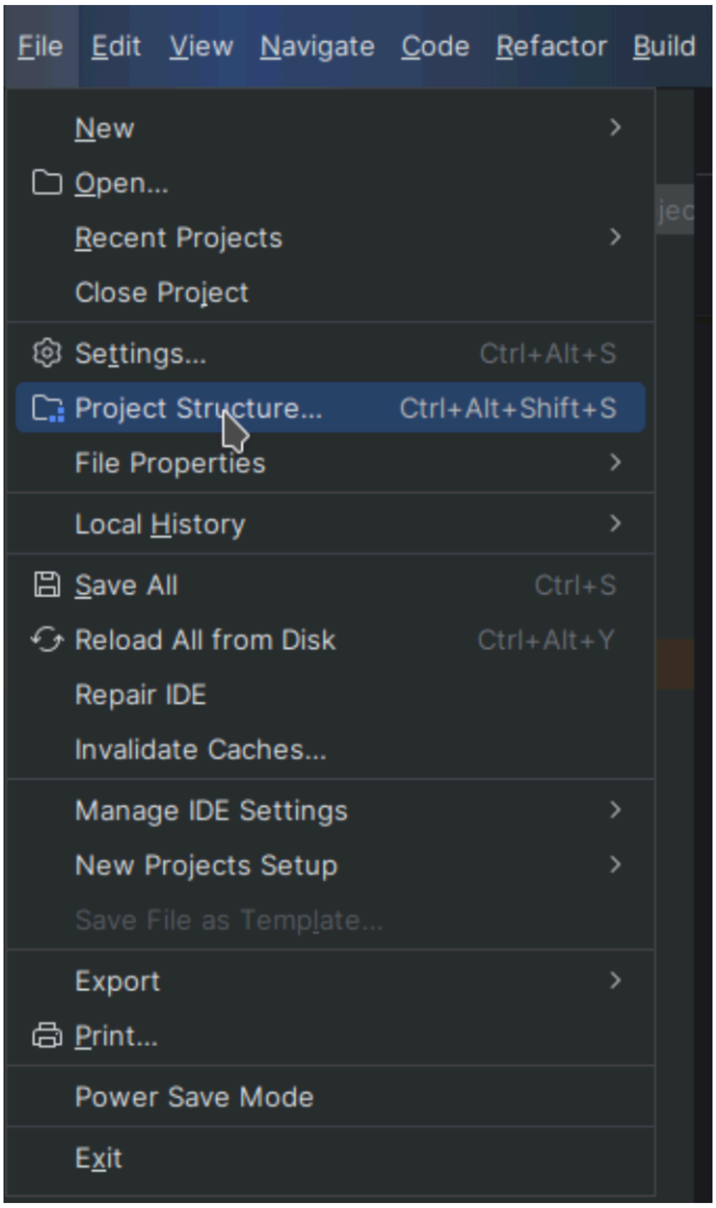
It defines the project's basic information, such as:

- `groupId` - A unique identifier for the project (for example, org.example)
- `artifactId` - The name of the project (for example, firstIntelliJProj)
- version - The version of the project (for example, 1.0.0).

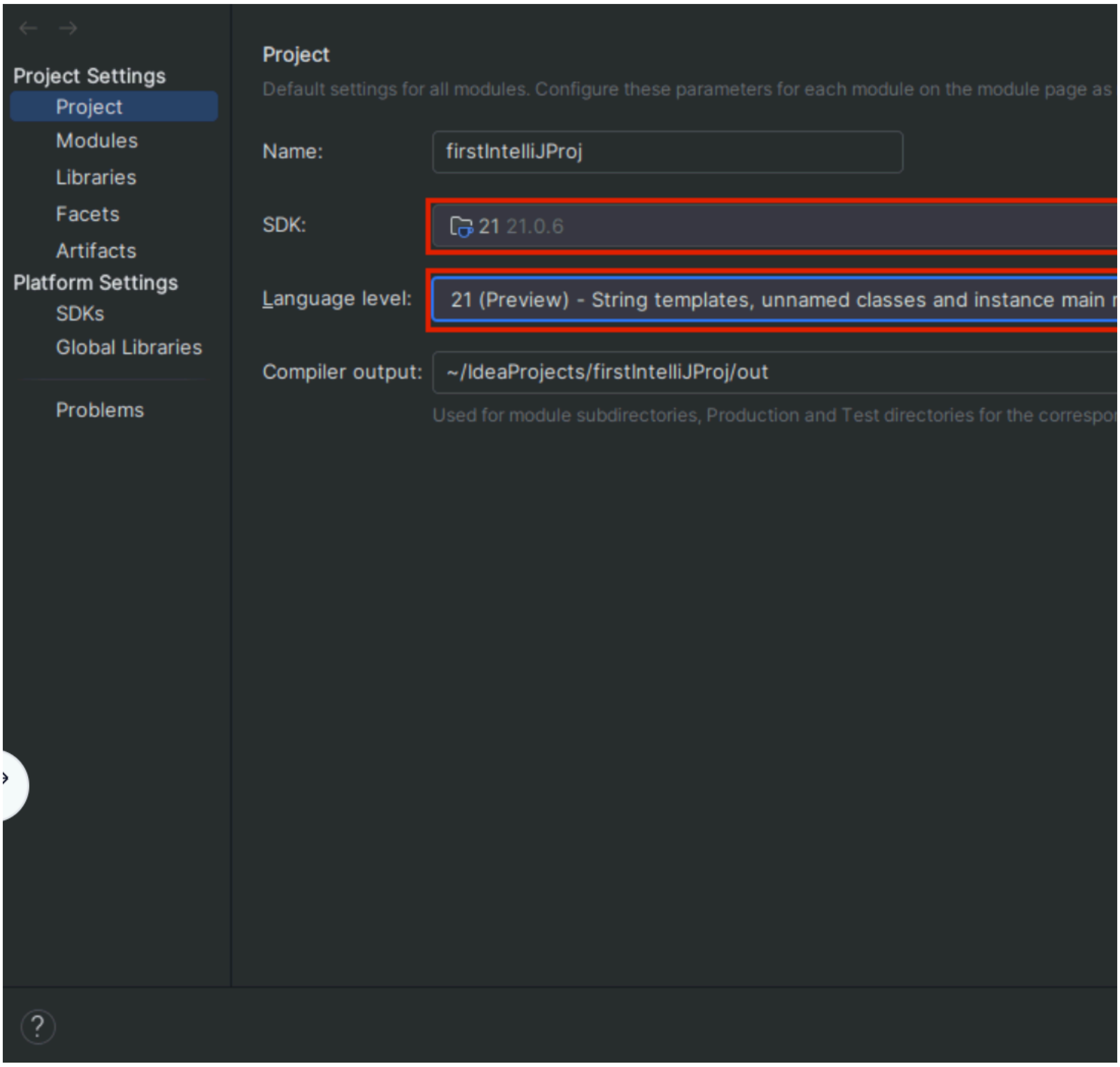
  Besides these, you have also define the repositories where the project can look for classes, the external dependencies for the classes in the project, and so on.

View and edit project settings

1. Click the `File` menu and select `Project Structure` to open the project settings.



2. Set the SDK and Language level to the version of Java installed (21 in this case), as shown in the image. Press `Apply` and then click `Ok`.


Project Settings
Project
Modules
Libraries
Facets
Artifacts
Platform Settings
SDKs
Global Libraries
Problems

Project
Default settings for all modules. Configure these parameters for each module on the module page as

Name: firstIntelliJProj

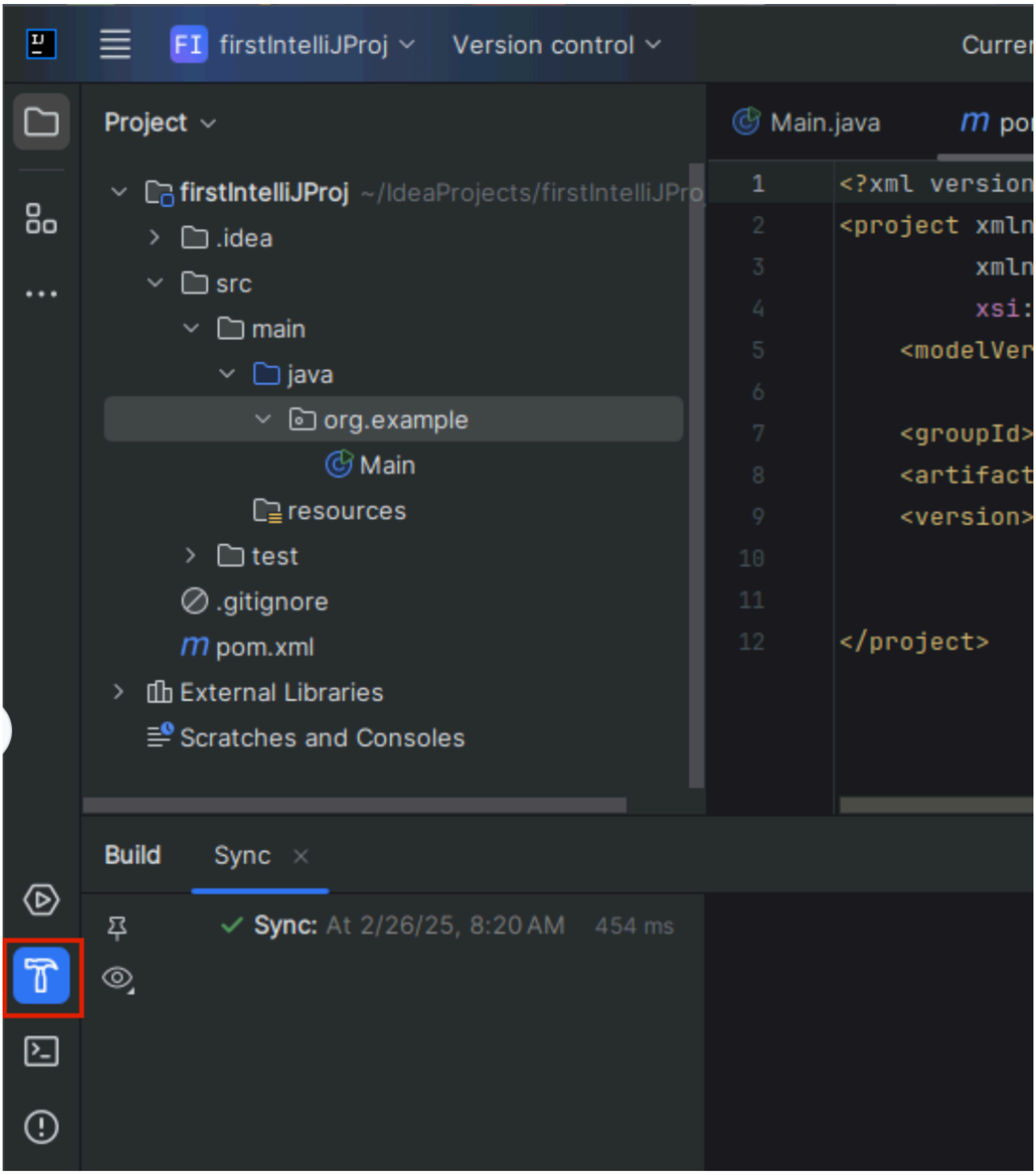
SDK: 21 21.0.6

Language level: 21 (Preview) - String templates, unnamed classes and instance main

Compiler output: ~/IdeaProjects/firstIntelliJProj/out
Used for module subdirectories, Production and Test directories for the correspo

Build and run the project

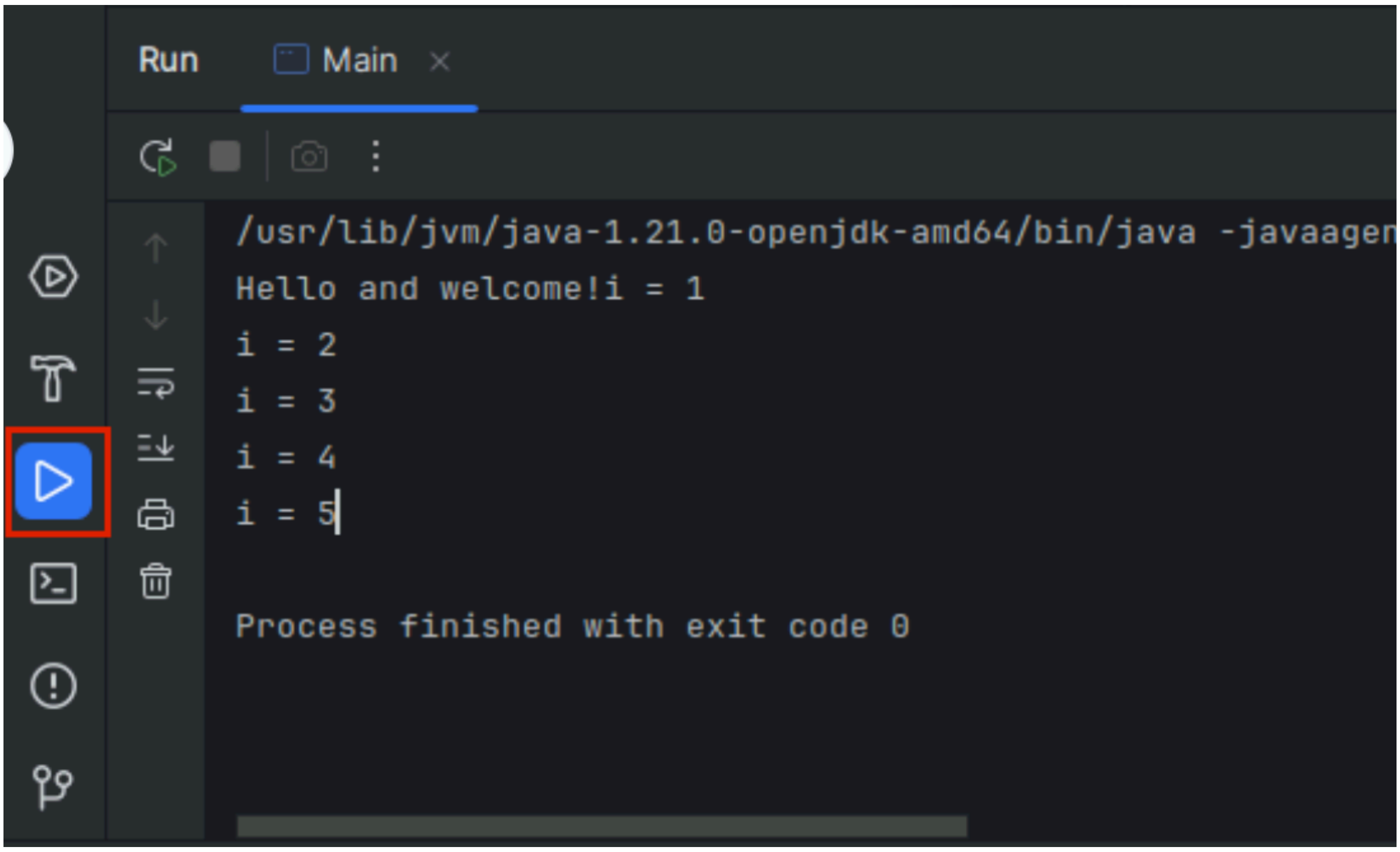
1. Build the project by clicking on the `build` hammer icon. If the build is successful you will see the green tick with no errors in the console.



```
<?xml version
<project xmln
         xmln
         xsi:
    <modelVer

    <groupId>
    <artifact
    <version>



</project>
```

```
Sync: At 2/26/25, 8:20 AM   454 ms
```

2. Run the project by clicking on the `run` play button icon. You will see the output of the main method on the console.



```
/usr/lib/jvm/java-1.21.0-openjdk-amd64/bin/java -javaagen
Hello and welcome!i = 1
i = 2
i = 3
i = 4
i = 5

Process finished with exit code 0
```

Practice Exercise

1. In `Main.java`, change the implementation for `main` method to iterate through an ArrayList of String, as given below and print the values on the console.

```
ArrayList<String> cities = new ArrayList<String>();
        usCities.add("New York");
        usCities.add("Los Angeles");
        usCities.add("Chicago");
        usCities.add("Houston");
        usCities.add("Phoenix");
        usCities.add("Philadelphia");
        usCities.add("San Antonio");
        usCities.add("San Diego");
        usCities.add("Dallas");
        usCities.add("San Jose");
```

▶ Click here for the solution

2. Build the project and run again and see the output.

Conclusion

In this lab you have:

- Set up the IntelliJ Community Edition Community edition
- Created a new maven project.
- Configured the `project` settings
- Built the project
- Ran the project and viewed the output on the terminal

Author(s)

Lavanya