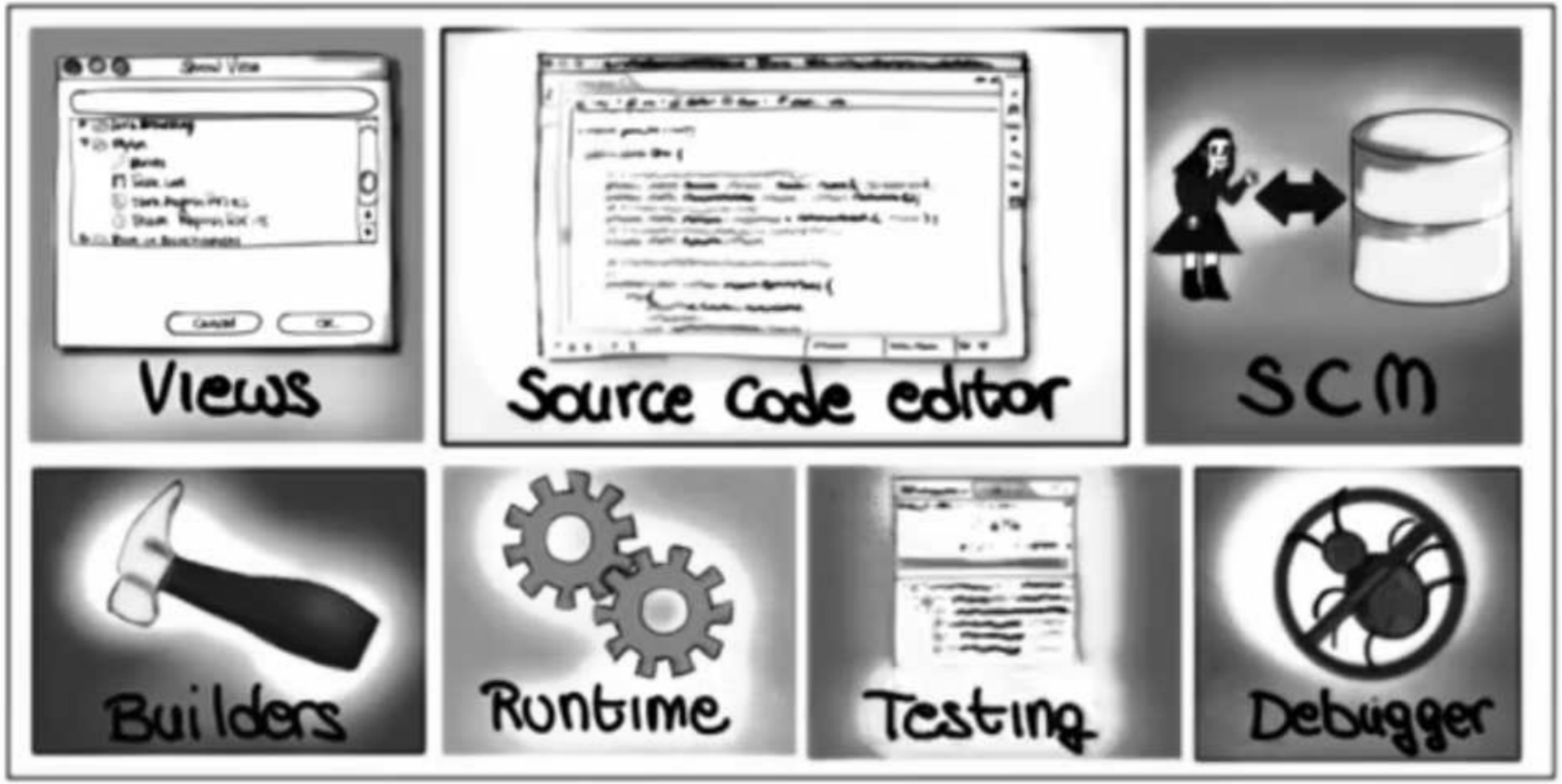
CS3300 Introduction to Software Engineering

Lecture 03: Tools of the Trade #2

IntelliJ IDEA, Junit Testing, Maven

Dr. Nimisha Roy ▸ nroy9@gatech.edu

What is an IDE?



- Integrated Development Environments (IDEs) are software applications that support developers in many of their everyday tasks, such as writing, compiling, and debugging code.
- Some IDEs are designed to support only one programming language such as Java, while others can be used for various languages
- Most popular Java IDEs: IntelliJ IDEA, Eclipse, Netbeans

Slide adapted from Alessandro Orso

What is an IDE?- IntelliJ

- IntelliJ was an IDE developed by JetBrains and is powerful in assisting developers write Java code because of its smart code analysis, plugin ecosystem, and java support. It supports Java, Kotlin, Scala, Groovy languages.

- Plug-ins: additional functionality offering more features to IDEs, not available in core. Eg: Python Plugin in IntelliJ, Egit plug-in Eclipse which adds support for Git Version Control

- Almost all IDEs work on Mac, Windows and Linux. Careful about version of IDE based on OS

Maven

- Powerful build management tool that can be used for building and managing any Java based project.
  - Easy building of project
  - Generate source code, generate documentation from source code (log document, dependency list, unit test reports etc)
  - Getting right dependencies for project, Compiling source code
  - Packaging compiled codes into JAR, WAR files without scripting
  - Installing packaged code in local, server or central repositories
- Based on POM (Project Object Model)
  - POM files are XML files that contain information related to the project and configuration information such as dependencies, source directory, plugin, goals etc. used by Maven to build the project.

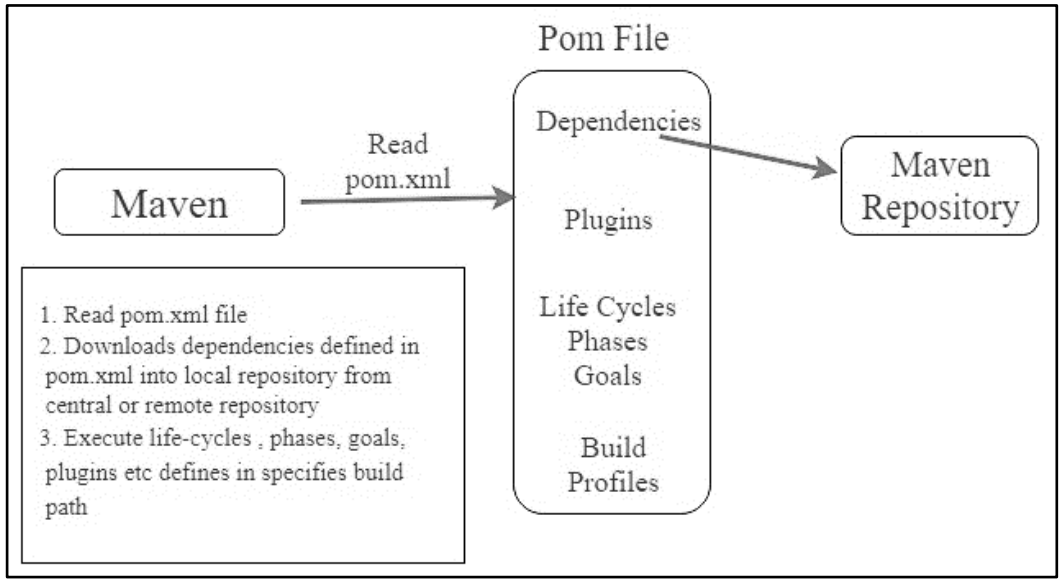


Image courtesy: https://www.geeksforgeeks.org/introduction-apache-maven-build-automation-tool-java-projects/

Junit Testing

- Lightweight framework for creating repeatable tests for your application
- Unit testing in Java
  - Imposes developers' discipline
  - Provides incremental specification
  - Avoids regression errors
  - Allows for changing with confidence
- Very helpful in Test driven development

Junit Best Practices

- Keep test cases separate from source code (src/main/tests)
- Method names need to be specific
- Keeping tests simple and focused on one feature or expected result
- Making sure to test for edge cases and not only “perfect scenarios”
- Use appropriate assertions to verify what's expected and what the function returns (actual)
- Tests are repeatable and reliable

```java
@Test
 public void testCalculateCircleAreaPositiveNumber() {
     double actualArea = Circle.calculateArea(2d); //radius = 2
     double expectedArea =  3.141592653589793 * 2 * 2; // formula = πr^2
     Assert.assertEquals(actualArea, expectedArea);

 }

 @Test
 public void testCalculateCircleAreaNegativeNumber() {
     double actualArea = Circle.calculateArea(-2d); //radius = -2
     double expectedArea =  3.141592653589793 * 2 * 2; // formula = πr^2
     Assert.assertNotEquals(actualArea, expectedArea);
 }
```

Demo – Run unit tests with Maven in Intellij IDEA

- Install JDK
- Install and setup IntelliJ
- Create a Maven project
- Perspectives & Layout
- Edit pom.xml to add dependencies
- Write code and test cases within the project
- Edit pom.xml to add dependencies
- Run tests

Demo – Run unit tests with Maven in Intellij IDEA

How to Install JDK

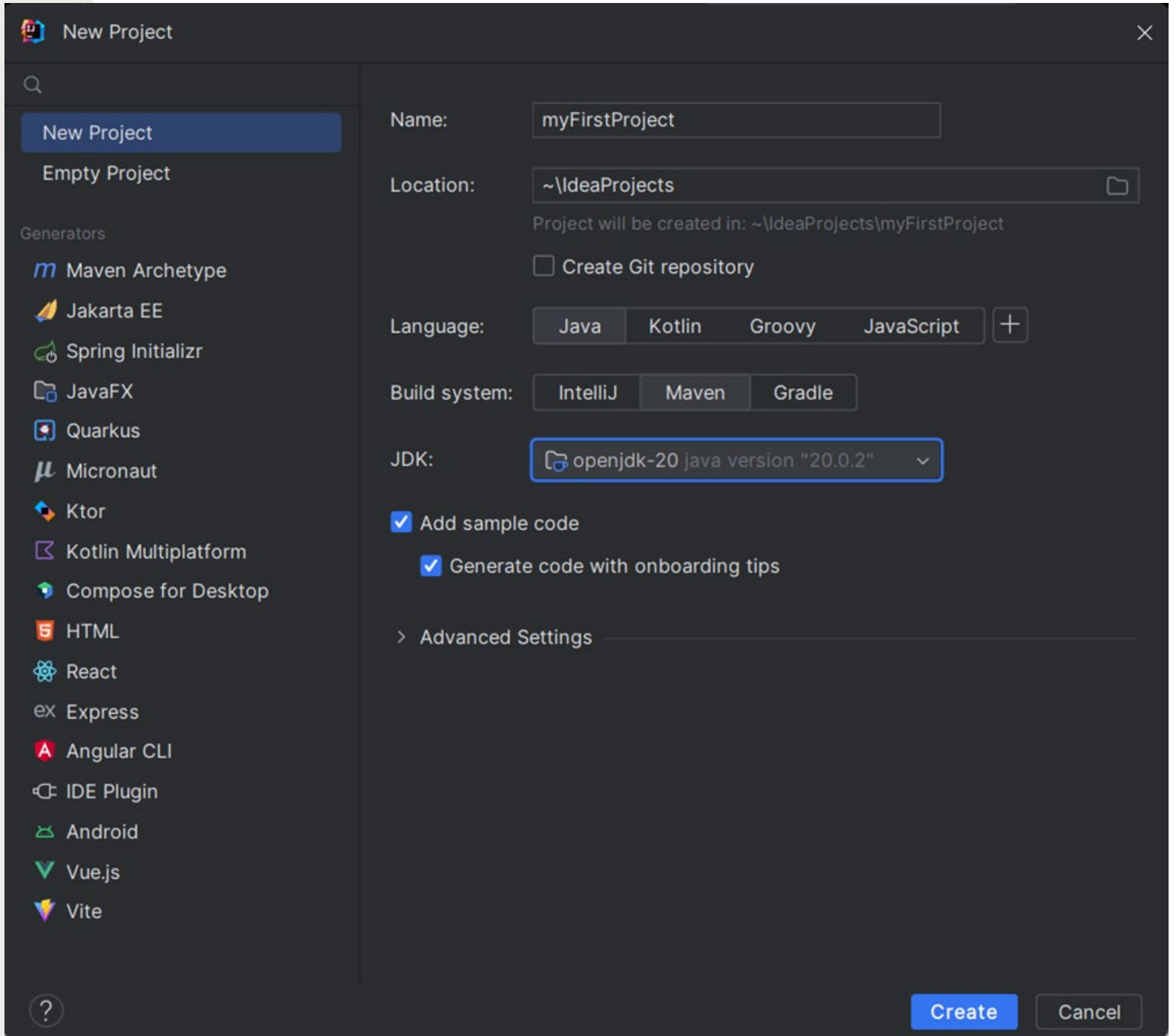
1. Browse to the AdoptOpenJDK website.
2. Choose your **Operating System**
3. Package type is **JDK**
4. Choose **OpenJDK 11 (LTS)** as the version
5. Click the **Latest release** download button to download the package.

How to Install and setup IntelliJ

- As a student or faculty member, you can get all of JetBrains's products for free. The easiest way is to visit JetBrains's student license page, click "APPLY NOW" and use your university email address. You'll get an email within a few minutes with instructions on downloading the products in their "Product Pack for Students."
- **Download IntelliJ IDEA Ultimate**

Demo – create project

- **How to create a new project with Maven**
- Select File -> New -> Project and add the following configurations.
- The JDK selected should be the JDK version you choose to use
- Choose Maven as the Build

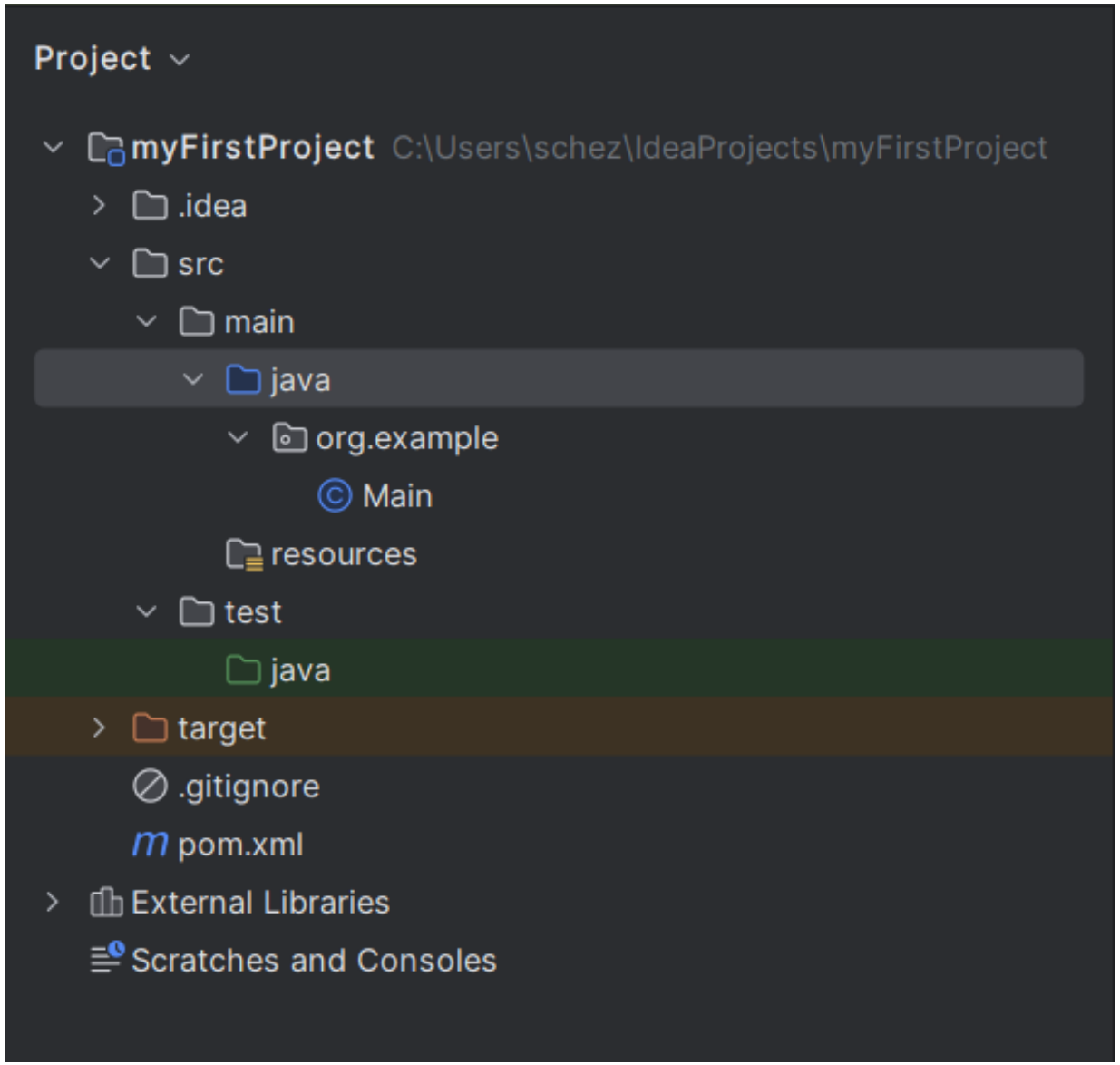


Demo – create project

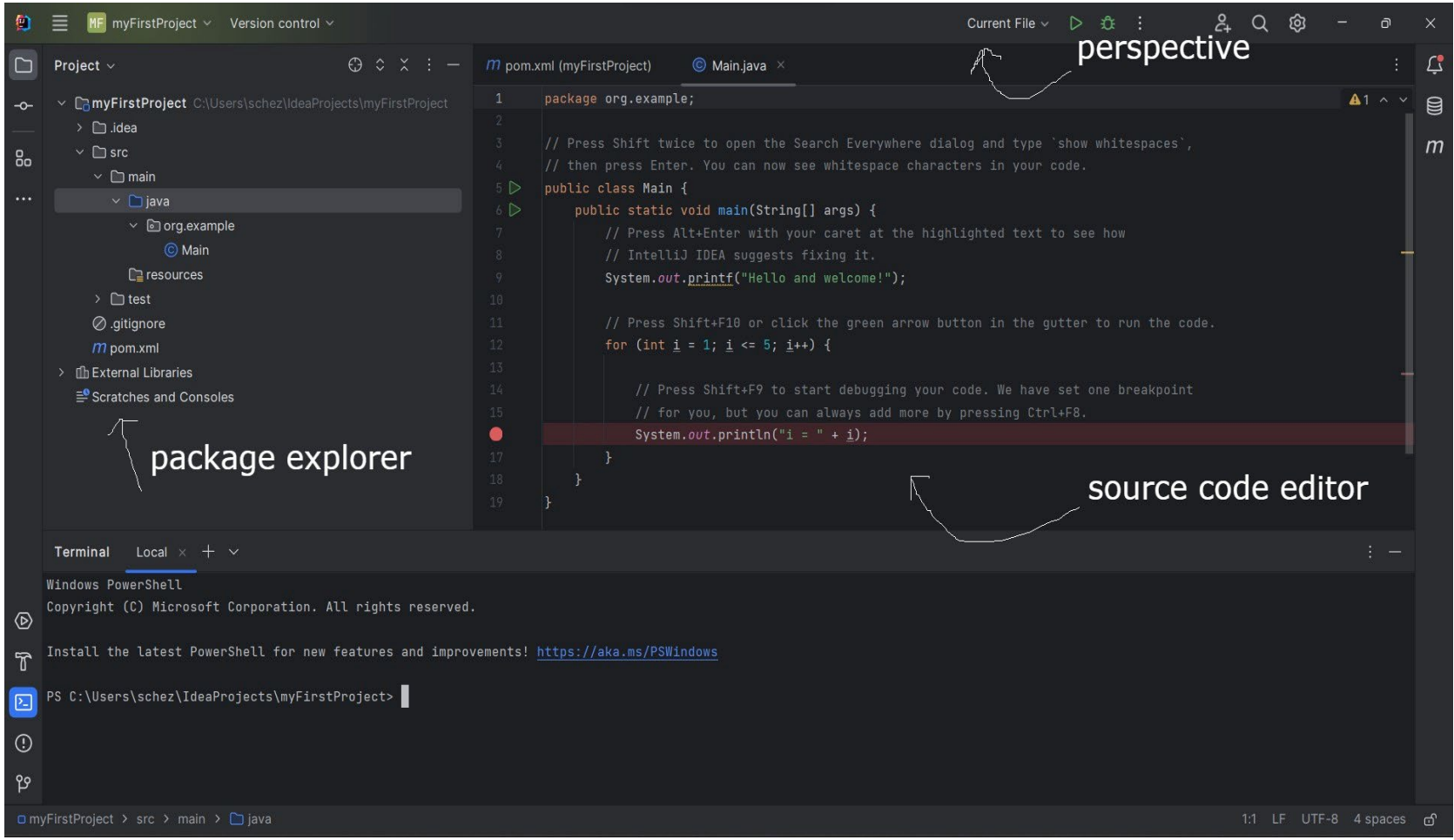
How to create a project with Maven

- Choose groupid and artifactid
  - Groupid: uniquely identifies your project across all projects. Kind of like package. *com.cs3300_Maven*
  - artifactId is the name of the project. Let's say *myMavenProject or myFirstProject*

- You will be able to see pom.xml file in the package explorer



Demo - Perspectives

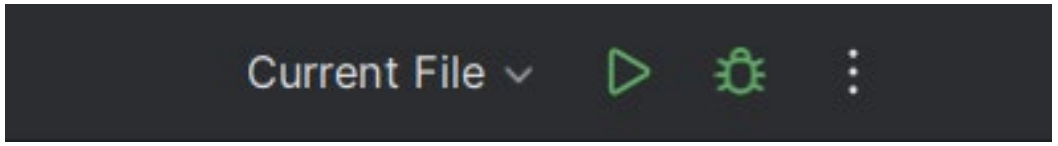


Perspective/Views

- The Project View on the left pane will show you your project hierarchy. This allows you to view how the classes are organized within your project.
- Source Code Editor in the middle. This is where most of your time will be spent within the IDE and where majority of your work will take place.

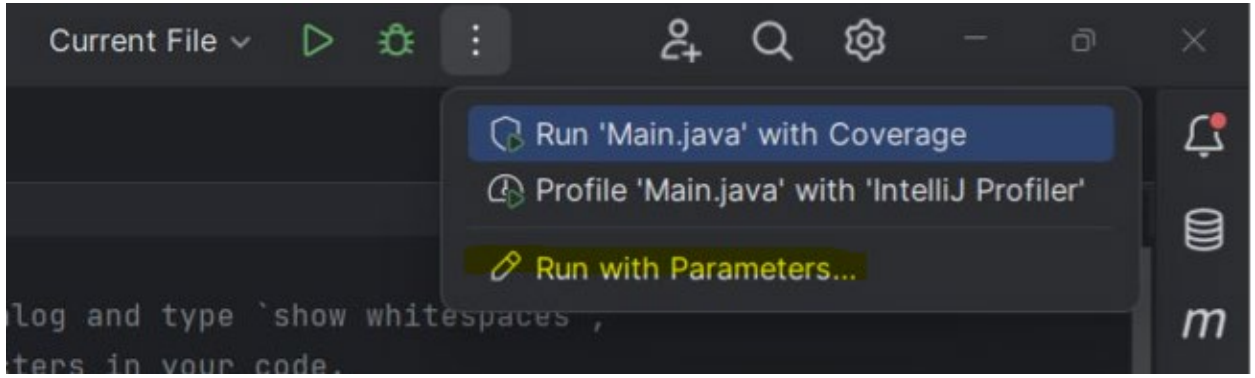
Demo – Run Configurations

Run Configurations



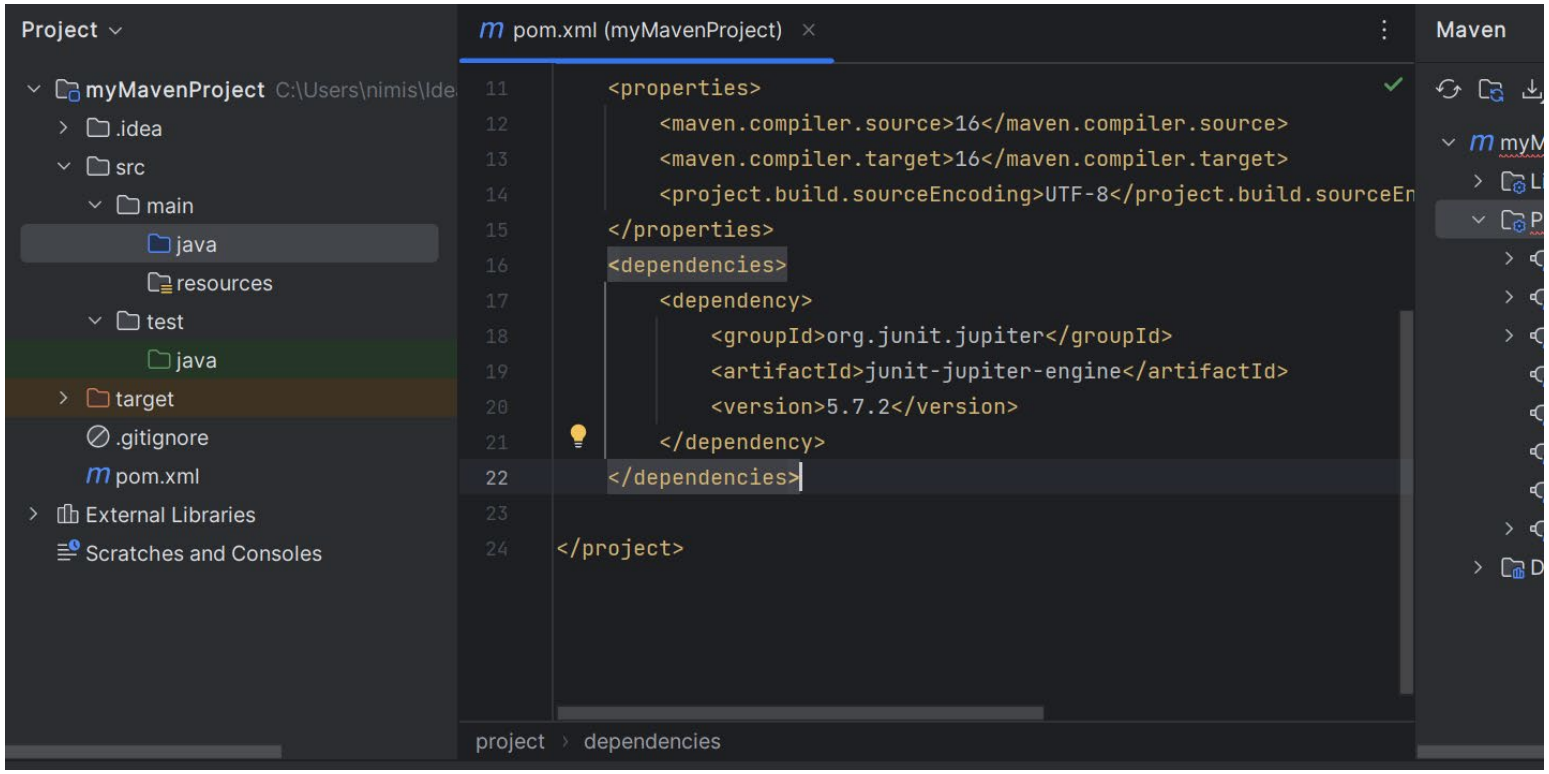
To run your program, click on the green arrow button with the target file selected from the dropdown.

To run with parameters, select the corresponding option from the menu given



Debug:
Add breakpoints, monitor value of variables. Select the 'green bug' icon to run your program in debug mode.

Demo- Editing pom.xml to add dependency



Edit pom.xml to add Junit as a dependency

1. In **pom.xml**, press Alt + Insert and select **Dependency**.
2. In the dialog that opens, type org.junit.jupiter:junit-jupiter in the search field. Locate the necessary dependency in the search results and click **Add**.

Demo- Adding code and test cases

- Create class in src/main/java. *AddConcatenate.java*
- Add 2 methods, 1 to add 2 numbers and 1 to concatenate 2 strings.
- Create corresponding 2 junit tests to test the 2 functions
- Run tests

Participation Assignment