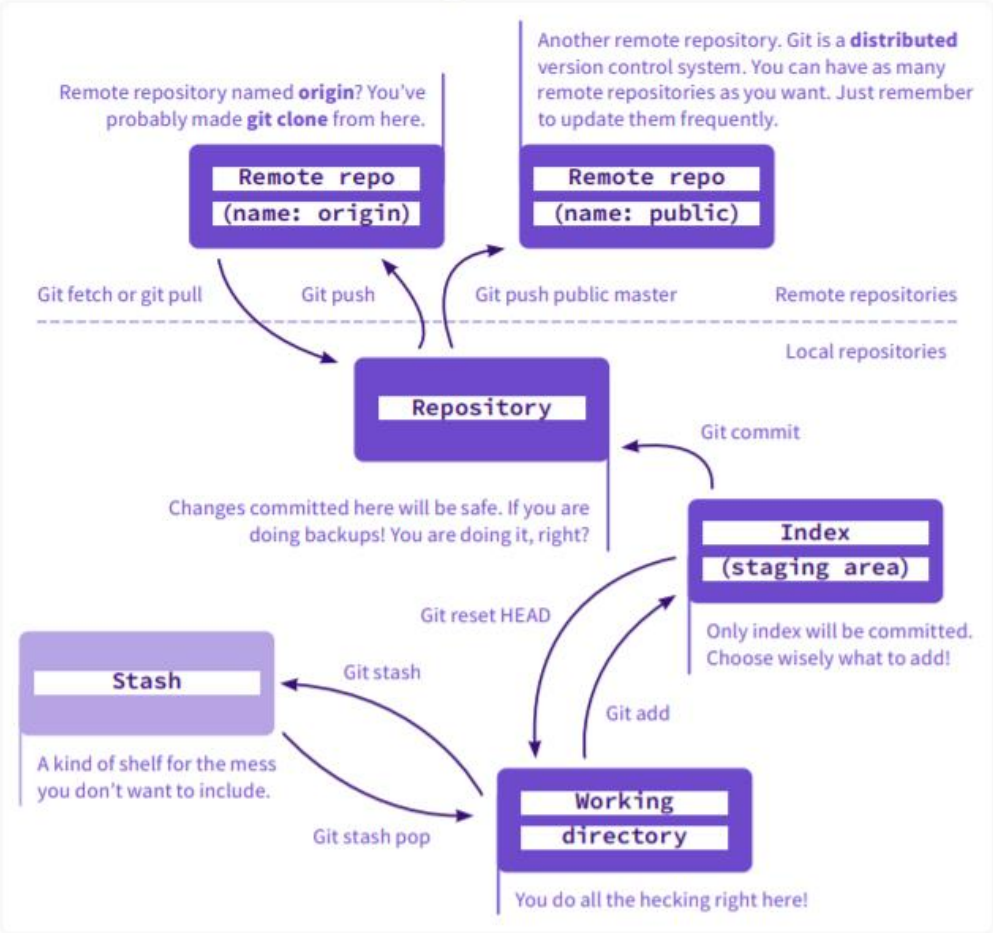
FUNDAMENTAL PROGRAMMING TECHNIQUES

GIT AND MAVEN

GIT - Overview

- Distributed version control system that tracks changes of computer files and helps coordinating several people who work on those files
- Advantages
  - Keeps a track of the changes made over time
  - Allows you to revert code to its initial stable version
  - Allows programmers to select which version of the file it wants to use at any point of time
  - Synchronizes code between multiple people
- Git-based repository hosting platforms: GitLab, GitHub

GIT – BASIC CONCEPTS AND COMMANDS



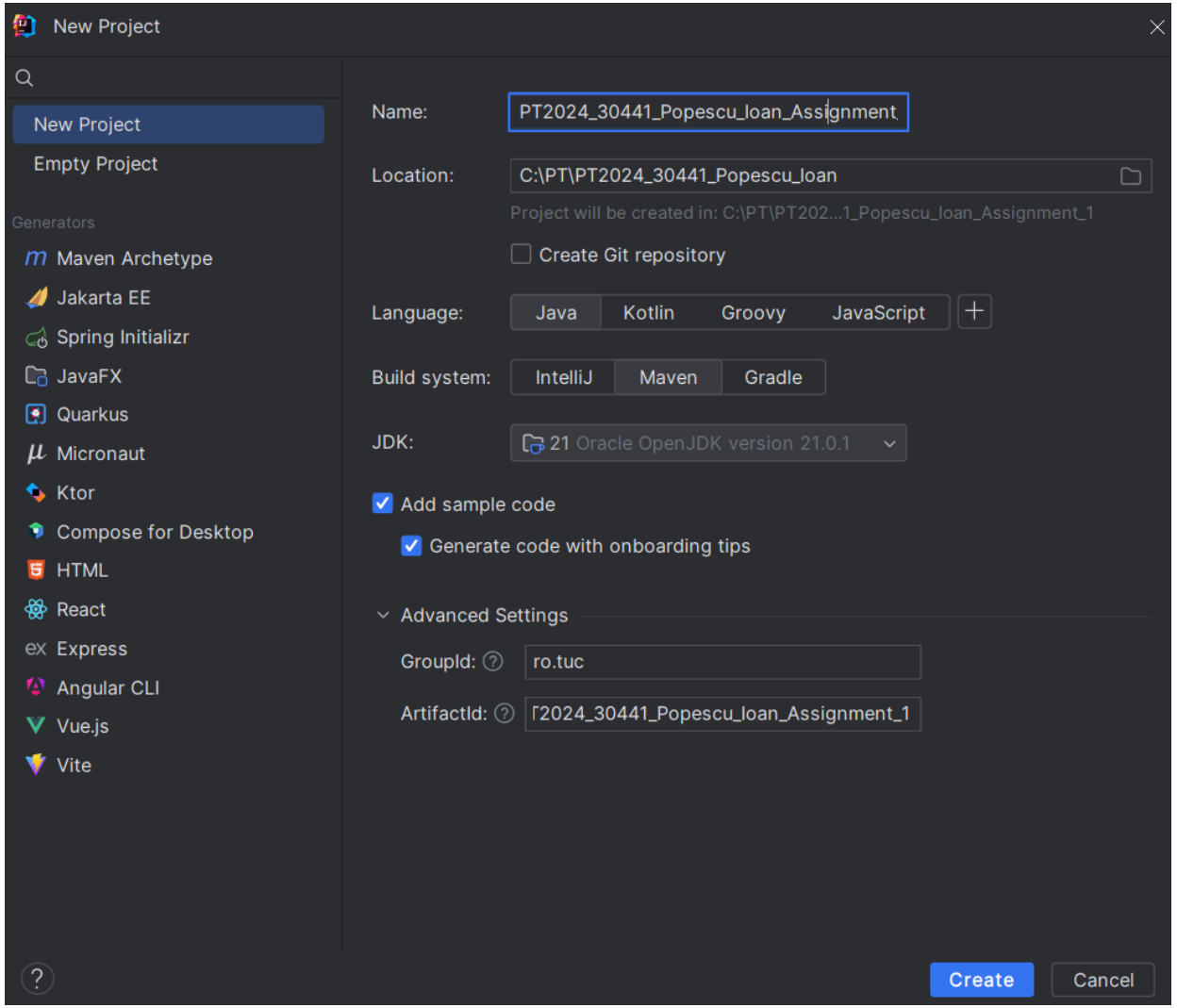
[Source]

Command	Description
git init	Initializes a repository in a project's directory => creates a subdirectory named .git containing the Git repository skeleton.
git clone remote_repository_URL	Gets a copy of an existing Git repository to which you want to contribute
git remote add origin remote_repository_URL	Sets a connection to another repository
git add .	Tracks new files, stages files
git status	Shows which files are in which state
git commit –m "message"	Commits the added files to your local git repository
git push –u origin main	Sends the committed changes to your remote repository
git pull	Fetches and merges changes on the remote server to your working directory

MAVEN - OVERVIEW

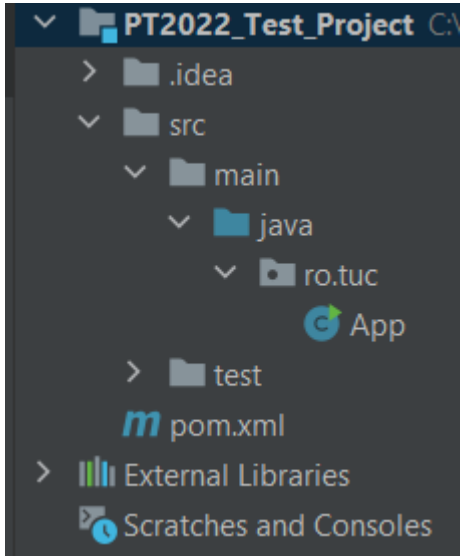
- Software project management tool that can be used for building and managing any Java-based project
- Builds a project using its **project object model** (i.e., POM) and a set of plugins
  - **Project object model** – XML file named "**pom.xml**" residing in the base directory of the project
    - Contains information about the project and various configuration details used by Maven to build the project (e.g., project dependencies)
- Maven Local Repository
  - Folder located on the local machine - created when you run any maven command for the first time
  - Keeps your project's dependencies (library jars, plugin jars etc.)
  - When you run a Maven build, then Maven automatically downloads all the dependency jars into the local repository
- Maven Central Repository
  - Provided by the Maven community
  - Contains a large number of commonly used libraries
  - When Maven does not find any dependency in local repository, it starts searching in the central repository

MAVEN – CREATE PROJECT



MAVEN – PROJECT STRUCTURE

Maven Project Structure



A Maven project must also adhere to a standard directory structure.

MAVEN STANDARD DIRECTORY LAYOUT (FRAGMENT)	
src/main/java	Contains Java code files
src/main/test	Contains test java code files
src/main/resources	Contains images/properties files

(For the complete directory layout check this link)

MAVEN – POM.XML

The ID of the project's group

The ID of the project

The version of the project

Specifies the type of artifact that the project produces (e.g., jar, war, etc.)

Junit dependency

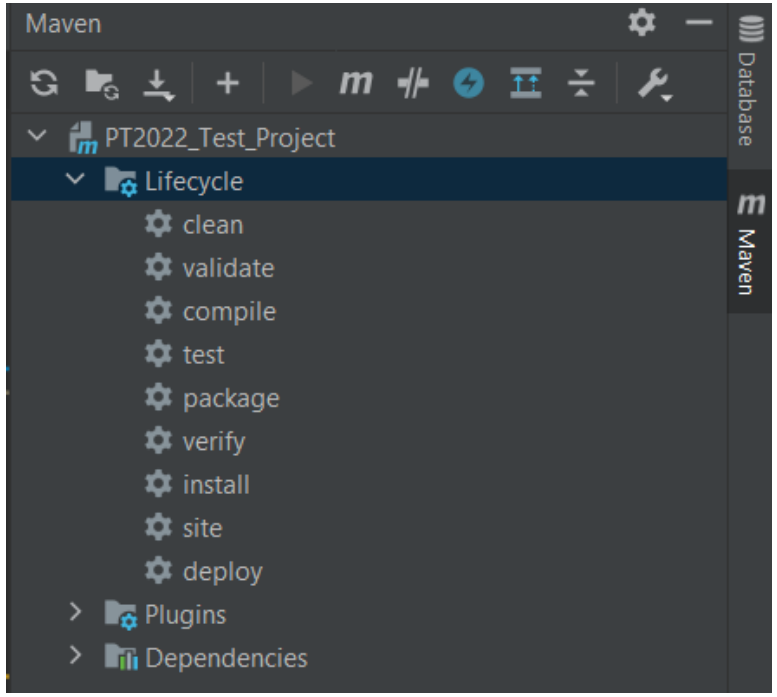
Adding the maven-jar-plugin

```xml
<?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>ro.tuc</groupId>
  <artifactId>PT2022_Test_Project</artifactId>
  <version>1.0-SNAPSHOT</version>
  <packaging>jar</packaging>
  <name>PT2022_Test_Project</name>
  <!-- FIXME change it to the project's website -->
  <url>http://www.example.com</url>
  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <maven.compiler.source>1.7</maven.compiler.source>
    <maven.compiler.target>1.7</maven.compiler.target>
  </properties>

  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.11</version>
      <scope>test</scope>
    </dependency>
  </dependencies>

  <build>
    <pluginManagement><!-- lock down plugins versions to avoid using Maven defaults (may be moved to pare
      <plugins>
        <plugin>
          <!-- Build an executable JAR -->
          <groupId>org.apache.maven.plugins</groupId>
          <artifactId>maven-jar-plugin</artifactId>
          <version>3.1.0</version>
          <configuration>
            <archive>
              <manifest>
                <mainClass>ro.tuc.App</mainClass>
              </manifest>
            </archive>
          </configuration>
        </plugin>
```

- Must be placed in the base directory of the project
- The dependencies' information can be retrieved from the Maven Central Repository:

https://mvnrepository.com/

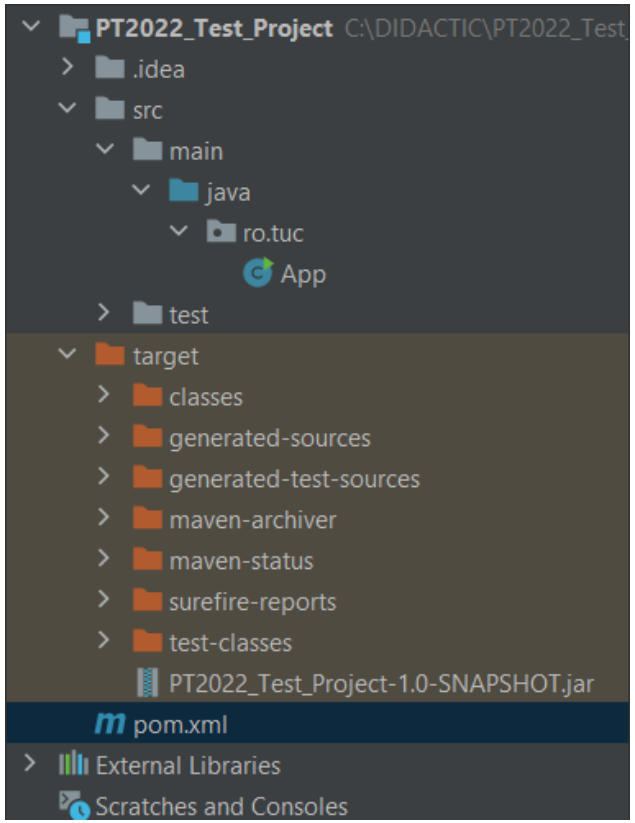
MAVEN – BUILD LIFECYCLE



Phase	Description
Clean	Deletes the build directory
Validate	Validates if the project is correct and if all necessary information is available
Compile	Compiles the source code
Test	Tests the compiled source code
Package	Creates the JAR/WAR package as mentioned in the packaging in POM.xml
Install	Installs the package in the local/remote maven repository

MAVEN – BUILD LIFECYCLE EXAMPLES

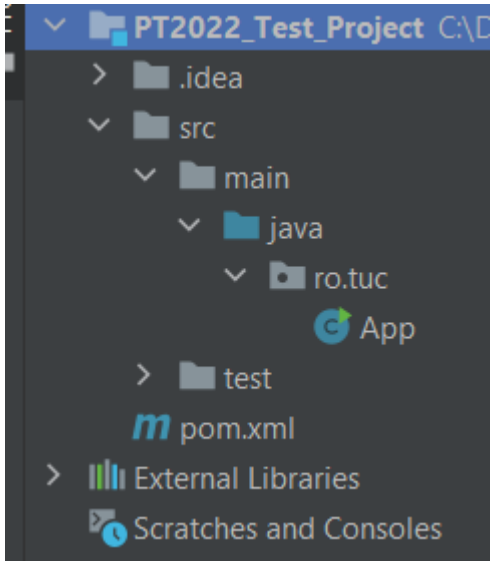
AFTER EXECUTING MAVEN PACKAGE



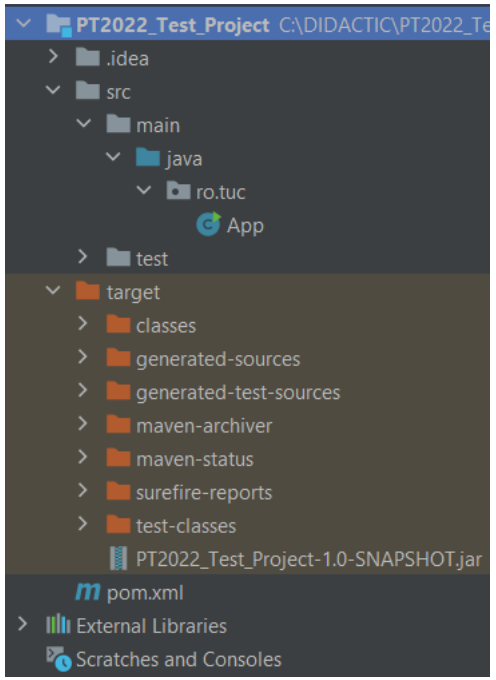
TO EXECUTE THE JAR RUN:

java -jar PT2022_Test_Project-1.0-SNAPSHOT.jar

AFTER EXECUTING MAVEN CLEAN

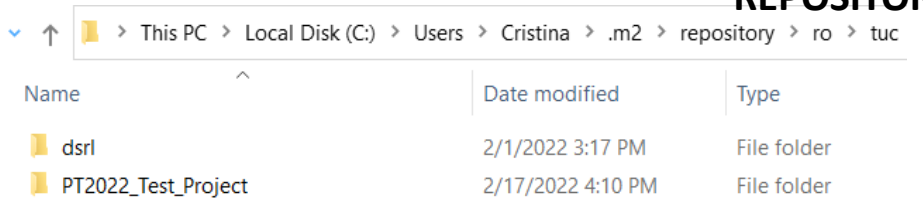


AFTER EXECUTING MAVEN INSTALL



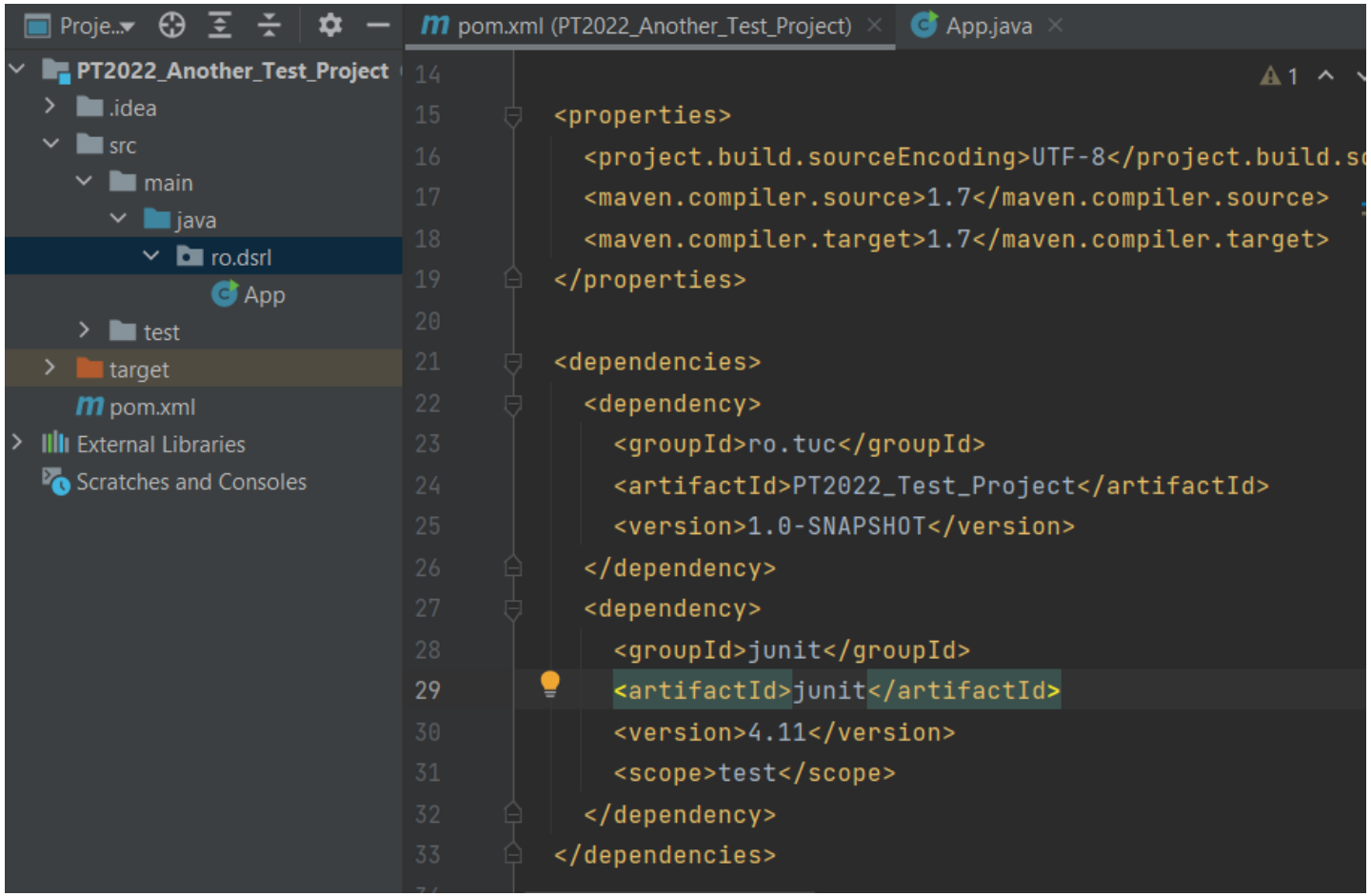
+

MAVEN LOCAL REPOSITORY



MAVEN – BUILD LIFECYCLE EXAMPLES

ADD DEPENDENCY TO THE PROJECT INSTALLED IN THE LOCAL DEPOSITORY IN ANOTHER PROJECT



Bibliography

[1] https://git-scm.com/book/en/v2

[2] https://maven.apache.org/index.html

[3] https://about.gitlab.com/images/press/git-cheat-sheet.pdf