**Second Program.**

1. **Command to build new maven project-**

mvn archetype:generate -DgroupId=com.example -DartifactId=myapp -DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false

1. cd install
2. mvn clean install
3. mvn compile
4. mvn test
5. java -cp target/myapp-1.0-SNAPSHOT.jar com.example.App

**Third Program.**

**(PHASE 1a)**

**BUILDING GROOVY PROJECT**

1. **Command to build new gradle project-**

gradle init --type java-application  
  
**It’ll ask user to fill these🡪**

**Enter target Java version (min: 7, default: 21):** 17

**Project name (default: Program3):** groovyProject

**Select application structure:**

**1: Single application project**

**2: Application and library project**

**Enter selection (default: Single application project) [1..2]** 1

**Select build script DSL:**

**1: Kotlin**

**2: Groovy**

**Enter selection (default: Kotlin) [1..2]** 2

**Select test framework:**

**1: JUnit 4**

**2: TestNG**

**3: Spock**

**4: JUnit Jupiter**

**Enter selection (default: JUnit Jupiter) [1..4]** 1

**Generate build using new APIs and behavior (some features may change in the next minor release)? (default: no) [yes, no]** no

1. gradle build
2. gradle test
3. gradle run

**(PHASE 1b)**

1. **app 🡪 build.gradle**

**there update these and write test**  
  
application {

mainClass = 'org.example.AdditionOperation'

}

dependencies {

testImplementation 'junit:junit:4.13.2'

}

test {

outputs.upToDateWhen { false }

testLogging {

events "passed", "failed", "skipped"

exceptionFormat = "full"

showStandardStreams = true

    }

}

1. **main 🡪 App.java**

**rename App.java to AdditionOperation.java**

**and code this**  
package org.example;

public class AdditionOperation {

    public static void main(String[] args) {

          double num1 = 5;

        double num2 = 10;

        double sum = num1 + num2;

        System.out.printf("The sum of %.2f and %.2f is %.2f%n", num1, num2, sum);

    }

}

1. **test 🡪AppTest.java**

**rename AppTest to AdditionOperationTest**

**and code this**  
package org.example;

import org.junit.Test;

import static org.junit.Assert.\*;

public class AdditionOperationTest{

    @Test

    public void testAddition() {

        double num1 = 5;

        double num2 = 10;

        double expectedSum = num1 + num2;

        double actualSum = num1 + num2;

        assertEquals(expectedSum, actualSum, 0.01);

    }

}

1. gradle build
2. gradle run
3. gradle test

**(PHASE 2a)**

**BUILDING KOTLOIN PROJECT**

1. **Command to build new gradle project**

gradle init --type java-application

1. **app 🡪 build.gradle.kts**

**make these code changes**

plugins {

    kotlin("jvm") version "1.8.21"

    application

}

dependencies {

implementation(kotlin("stdlib"))

    testImplementation("junit:junit:4.13.2")

}

application {

    mainClass.set("org.example.MainKt")

}

tasks.test {

    useJUnit()

    testLogging {

        events("passed", "failed", "skipped")

        exceptionFormat = org.gradle.api.tasks.testing.logging.TestExceptionFormat.FULL

        showStandardStreams = true

    }

    outputs.upToDateWhen { false }

}

java {

    toolchain {

        languageVersion.set(JavaLanguageVersion.of(17))

    }

}

1. **main 🡪 App.java**

**resave App.java as Main.kt**

**make these code changes**

package org.example

 fun addNumbers(num1: Double, num2: Double): Double {

 return num1 + num2

 }

 fun main() {

    val num1 = 10.0

    val num2 = 5.0

    val result = addNumbers(num1, num2)

    println("The sum of $num1 and $num2 is: $result")

 }

1. **test 🡪 AppTest.java**

**resave AppTest.java as MainTest.kt**

**make these code changes**

package org.example

 import org.junit.Assert.\*

 import org.junit.Test

 class MainTest {

    @Test

    fun testAddNumbers() {

        val num1 = 10.0

        val num2 = 5.0

                val result = addNumbers(num1, num2)

        assertEquals("The sum of $num1 and $num2 should be 15.0", 15.0, result, 0.001)

    }

 }

1. gradle build
2. gradle run
3. gradle test

**Fourth Program.**

1. **Create a Maven Project**

mvn archetype:generate -DgroupId=com.example -DartifactId=maven-example -DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false

1. **Edit pom.xml file**

**Add this code**

<?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>com.example</groupId>

    <artifactId>maven-example</artifactId>

    <version>1.0-SNAPSHOT</version>

    <dependencies>

        <dependency>

            <groupId>junit</groupId>

            <artifactId>junit</artifactId>

            <version>4.12</version>

            <scope>test</scope>

        </dependency>

    </dependencies>

    <build>

        <plugins>

            <plugin>

                <groupId>org.apache.maven.plugins</groupId>

                <artifactId>maven-compiler-plugin</artifactId>

                <version>3.8.1</version>

                <configuration>

                    <source>1.8</source>

                    <target>1.8</target>

                </configuration>

            </plugin>

        </plugins>

    </build>

</project>

1. **Main 🡪 App.java**

**Copy paste this code**  
  
package com.example;

 public class App {

    public static void main(String[] args) {

        System.out.println("Hello, Maven");

        System.out.println("This is the simple realworld example....");

        int a = 5;

        int b = 10;

        System.out.println("Sum of " + a + " and " + b + " is " + sum(a, b));

    }

    public static int sum(int x, int y) {

        return x + y;

    }

 }

1. cd maven-example
2. mvn clean install
3. mvn exec:java -Dexec.mainClass="com.example.App"
4. **Next run this command in maven-project only to migrate to gradle**

gradle init

**Next user is prompted to enter this**

**Found a Maven build. Generate a Gradle build from this? (default: yes) [yes, no]** yes

**Select build script DSL:**

**1: Kotlin**

**2: Groovy**

**Enter selection (default: Kotlin) [1..2]** 2

**Generate build using new APIs and behavior (some features may change in the next minor release)? (default: no) [yes, no]** no

1. **Maven-example 🡪 build.gradle**

**Copy paste this**  
  
 plugins {

    id 'java'

 }

 group = 'com.example'

 version = '1.0-SNAPSHOT'

 repositories {

    mavenCentral()

 }

 dependencies {

    testImplementation 'junit:junit:4.12'

 }

task run(type: JavaExec) {

    main = 'com.example.App'

    classpath = sourceSets.main.runtimeClasspath

 }

1. gradle build
2. gradle run

**Sixth Program.**

1. **Load JENKINS**

java -jar jenkins.war --httpPort=9090

1. **Manage Jenkins 🡪 Plugins 🡪 Available Plugins**

maven integration plugin(install this)

**restart JENKINS**

1. **Manage Jenkins 🡪 Tools**

**Under Maven Installations**

**Name:** Apache Maven 3.9.9

**MAVEN\_HOME:** path to maven from system environment

**Under JDK installations**

**Name:** jdk-21

**JAVA\_HOME:** Path to jdk from system environment

1. **New Item 🡪 Maven Project**

**Name:** Maven Test

**Click Ok**

1. **Source Code Management**

None

1. **Build**

**Goals and Options:** install

1. **Maven project (program 2) 🡪 myapp**

**Copy** src,target and pom **files and paste in Users 🡪 username 🡪.jenkins 🡪 workspace 🡪 Jenkins Item Name (Maven Test in this case)**

1. Click on Build Now