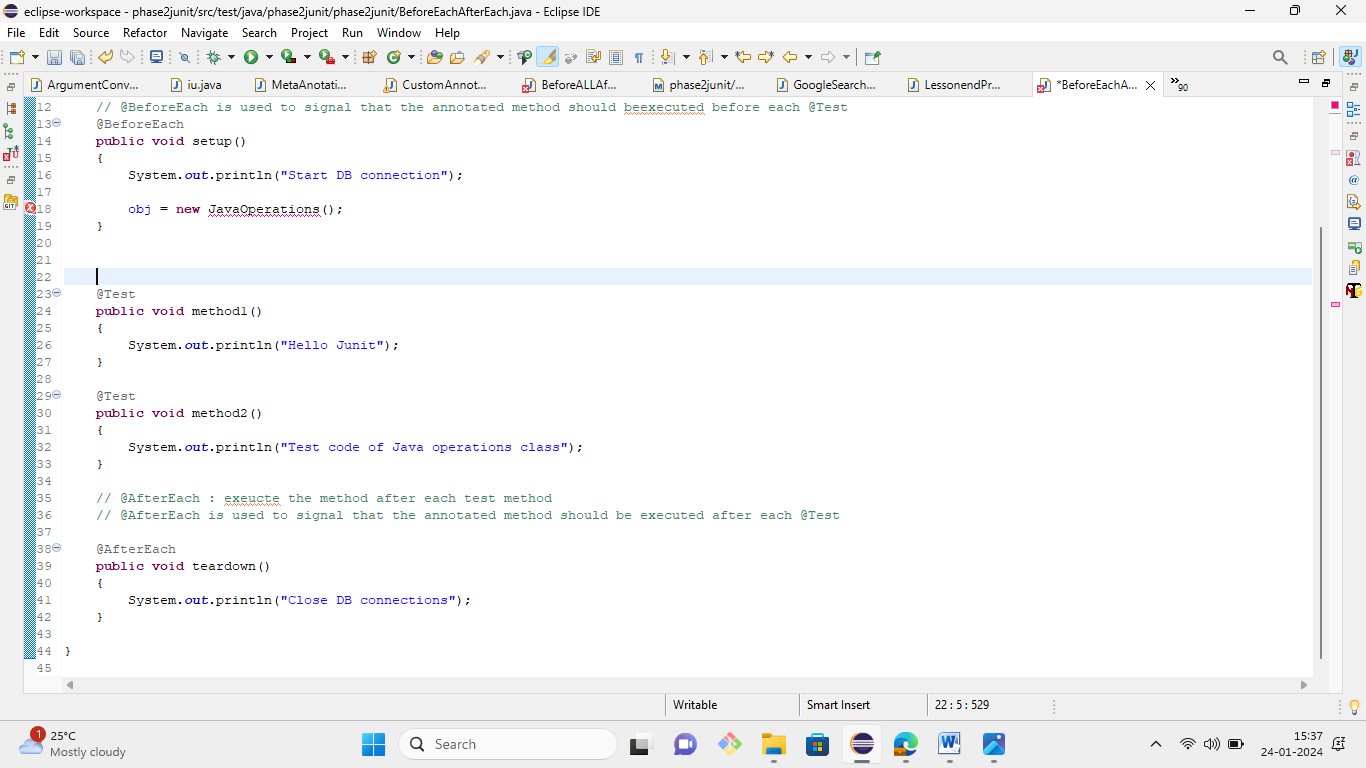
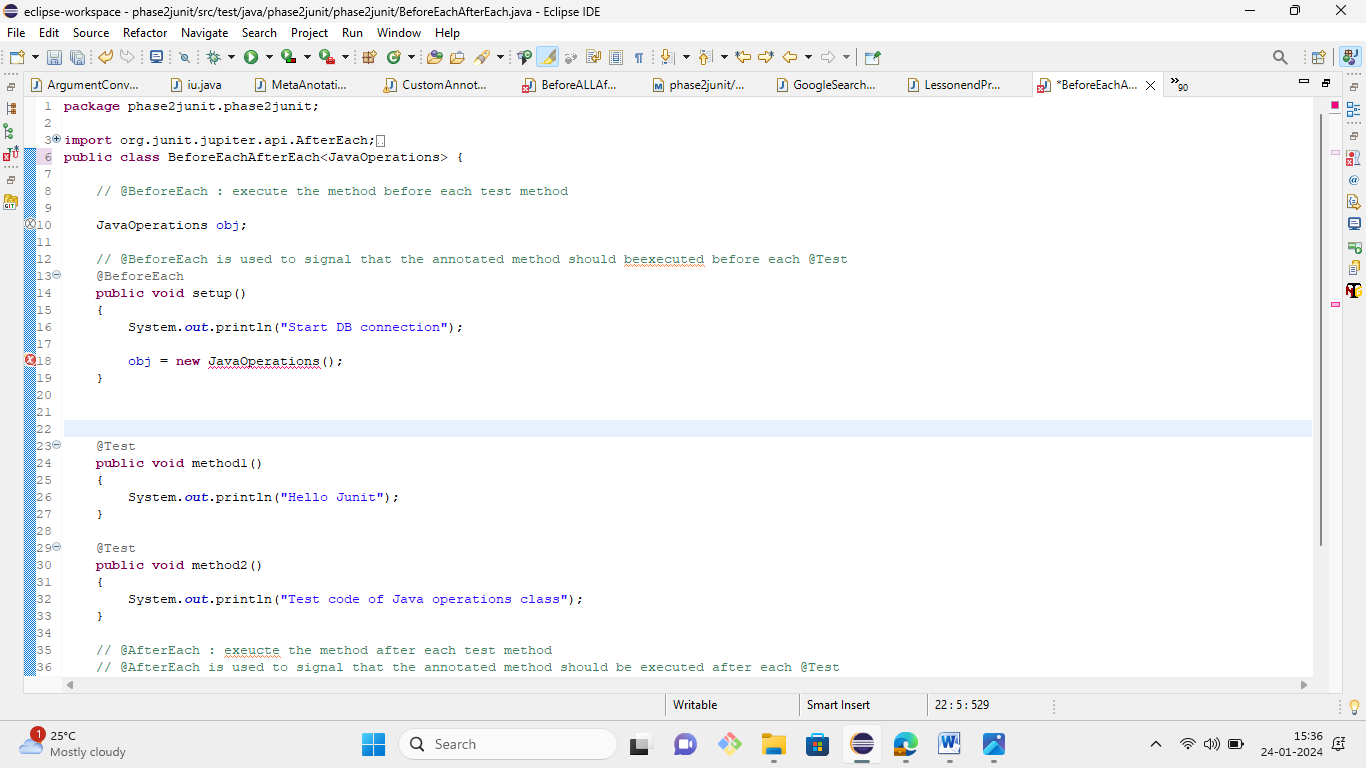
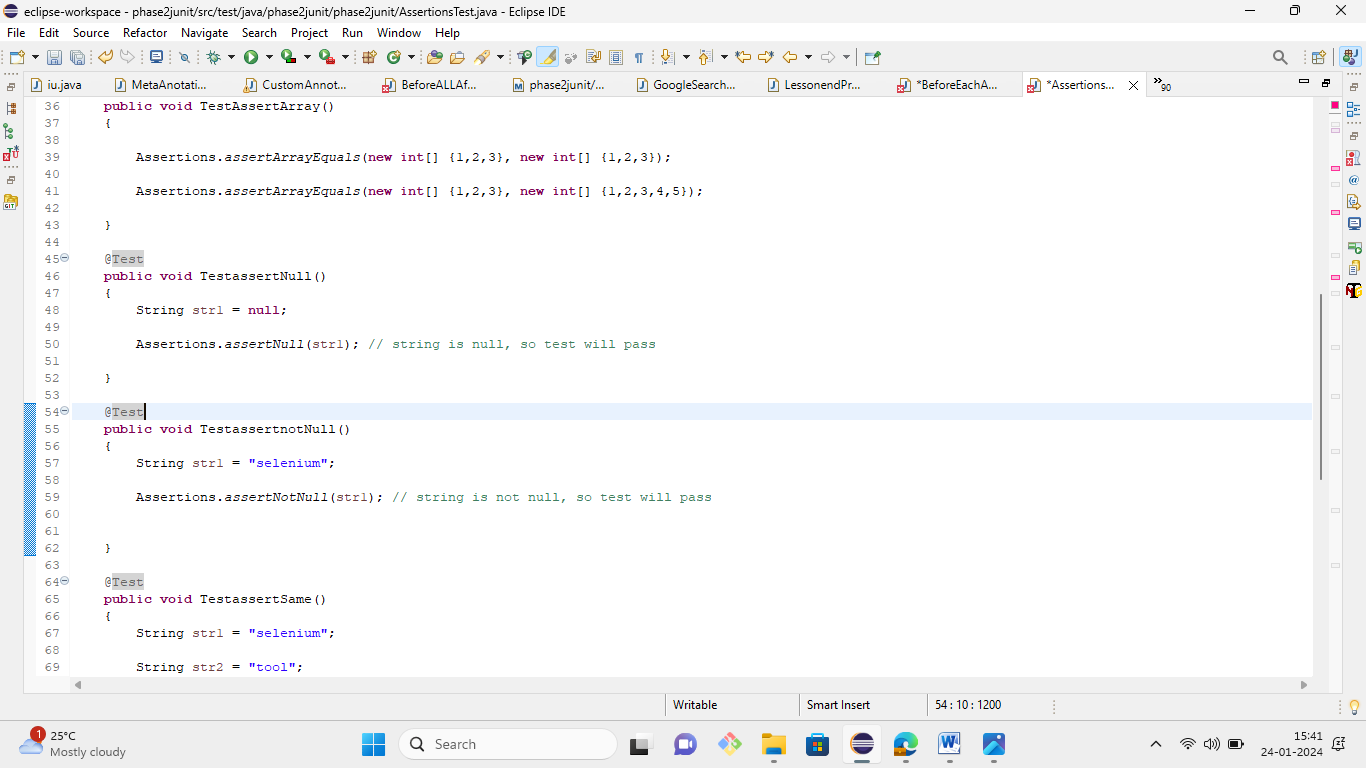
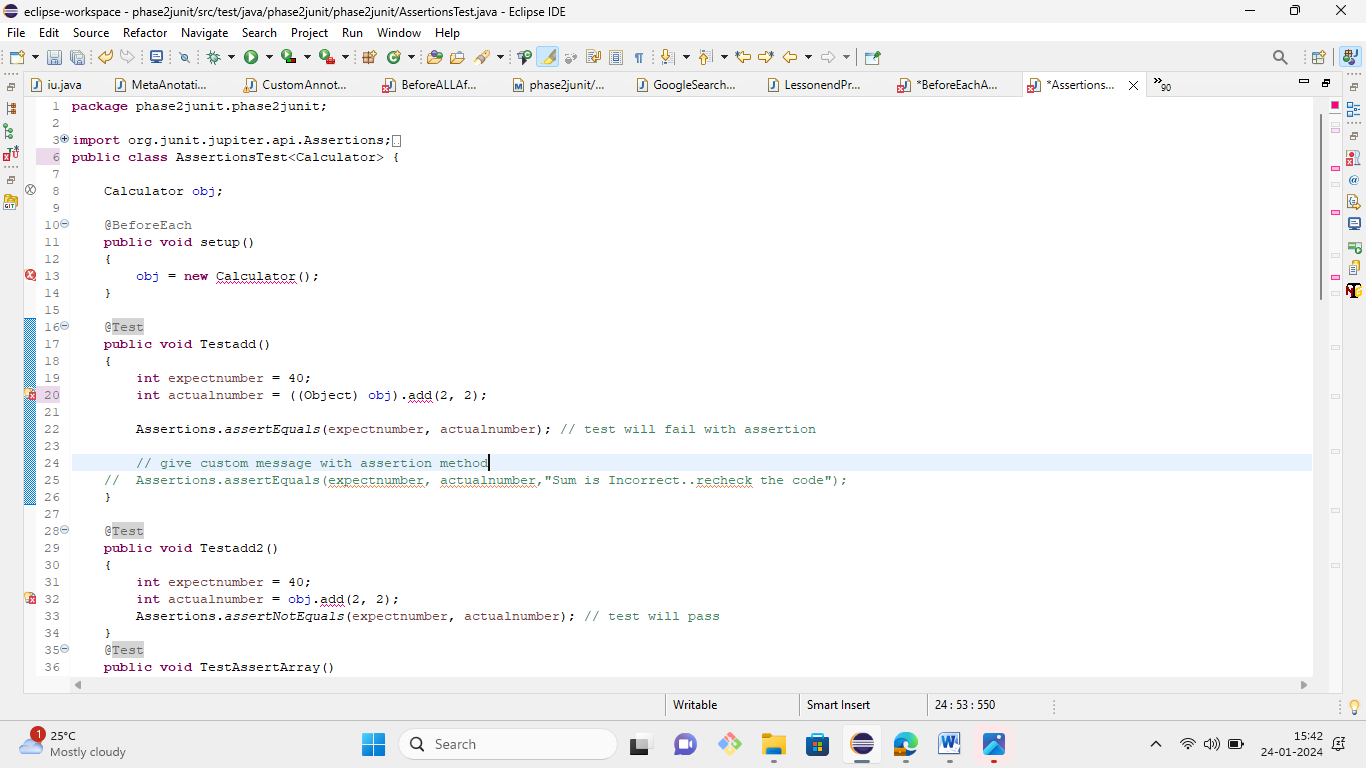
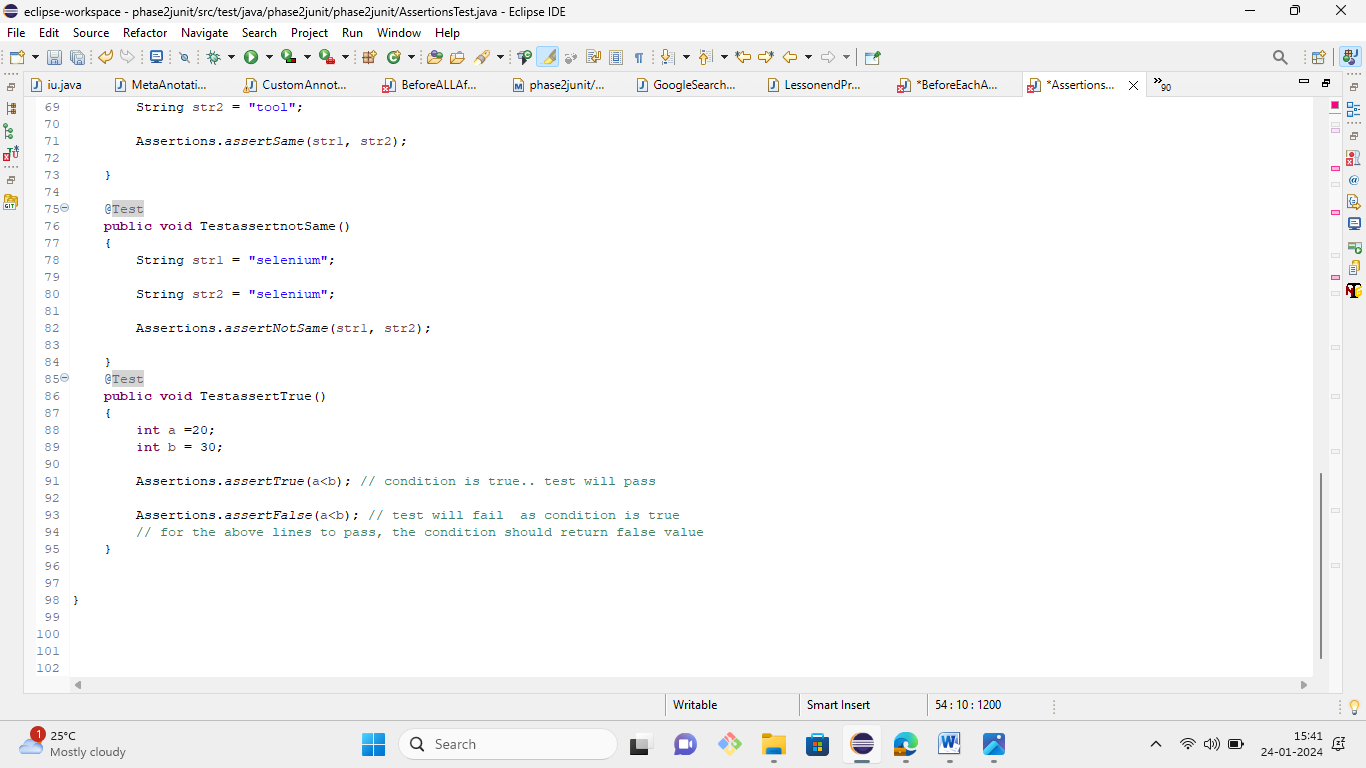
ASSISTED PROJECT EMP ID – 2587303

NAME:R.HIMABINDU

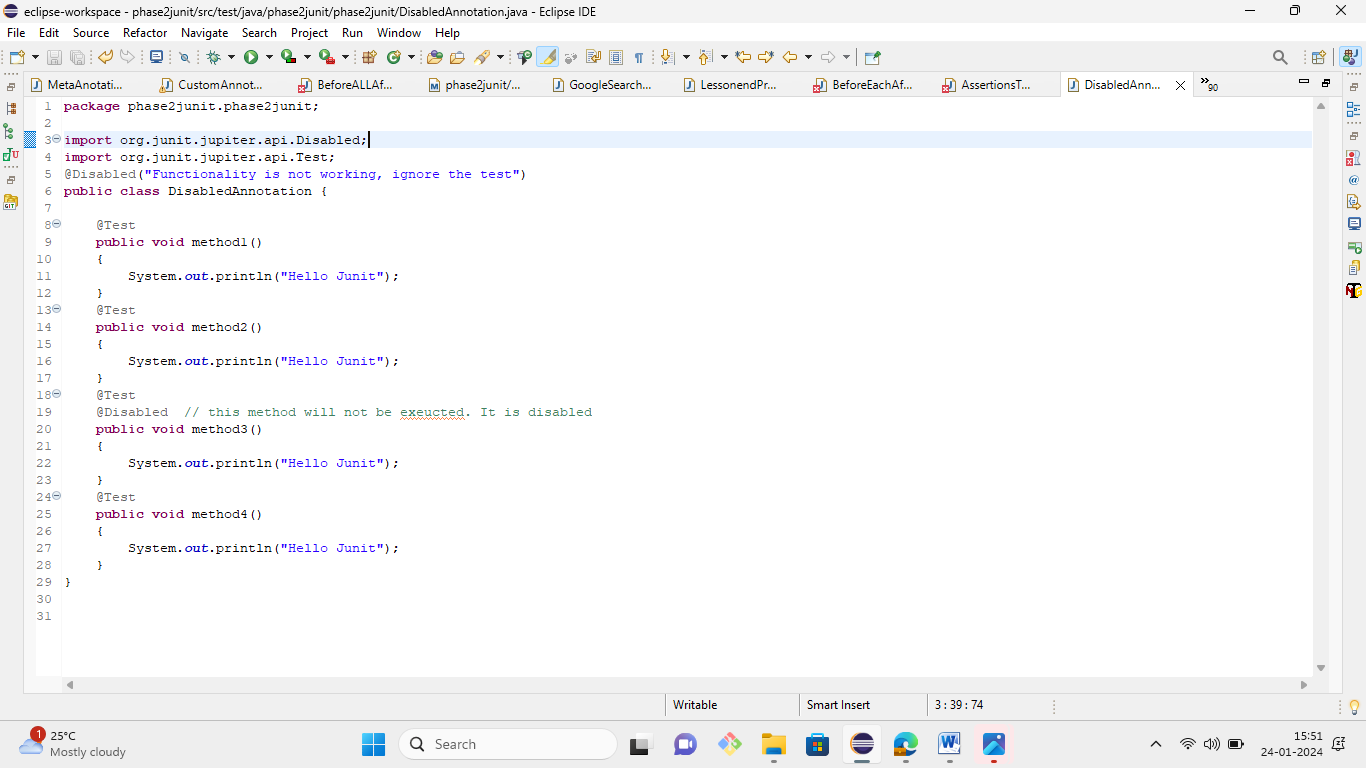
1. Write a JUnit program to demonstrate lifecycle methods.



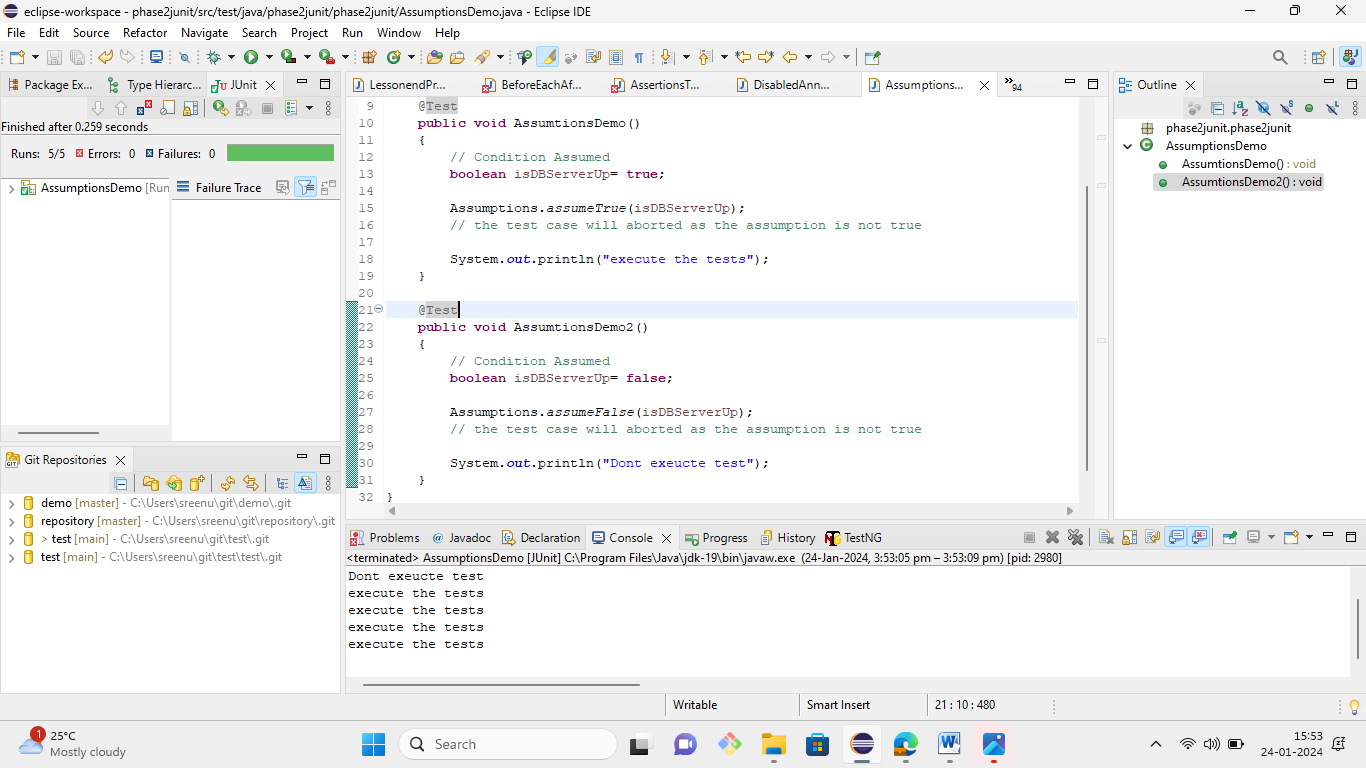
1. Write a JUnit program to demonstrate assertions.



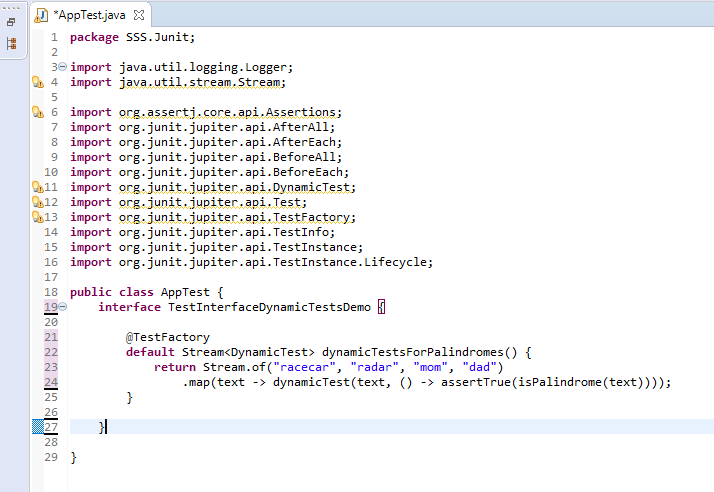
1. Write a JUnit program to demonstrate how tests are disabled.



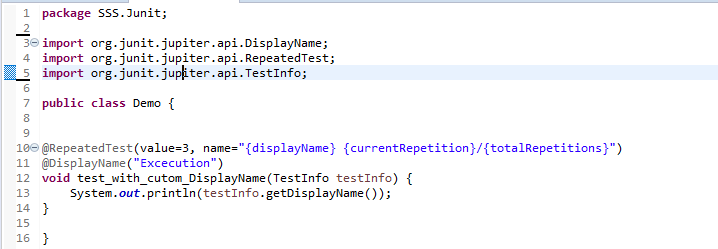
1. Write a JUnit program to demonstrate assumptions in JUnit.

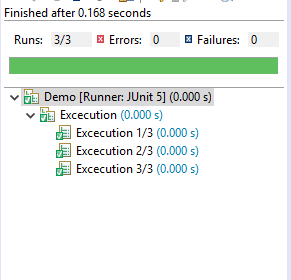


1. Write a JUnit program to demonstrate test interfaces and default methods in JUnit.

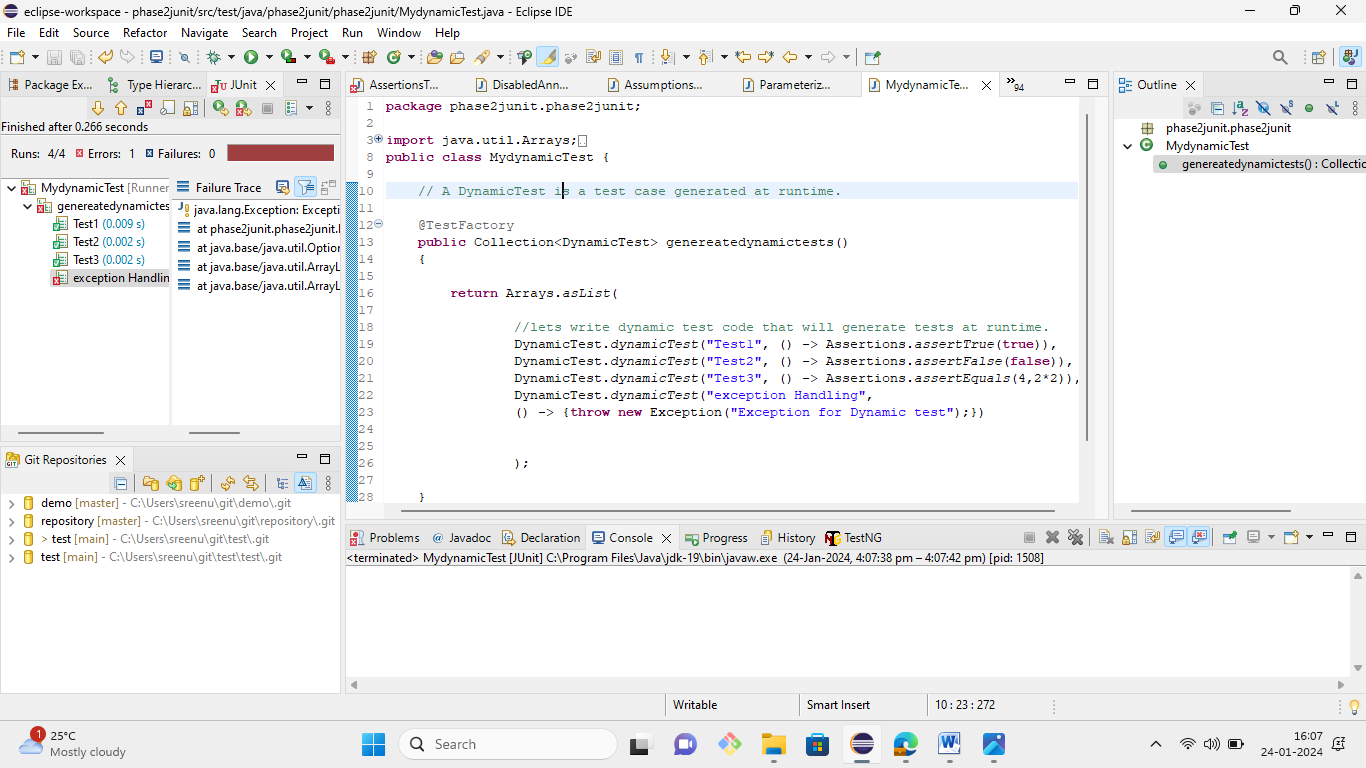


1. Write a JUnit program to demonstrate how tests are repeated in JUnit.

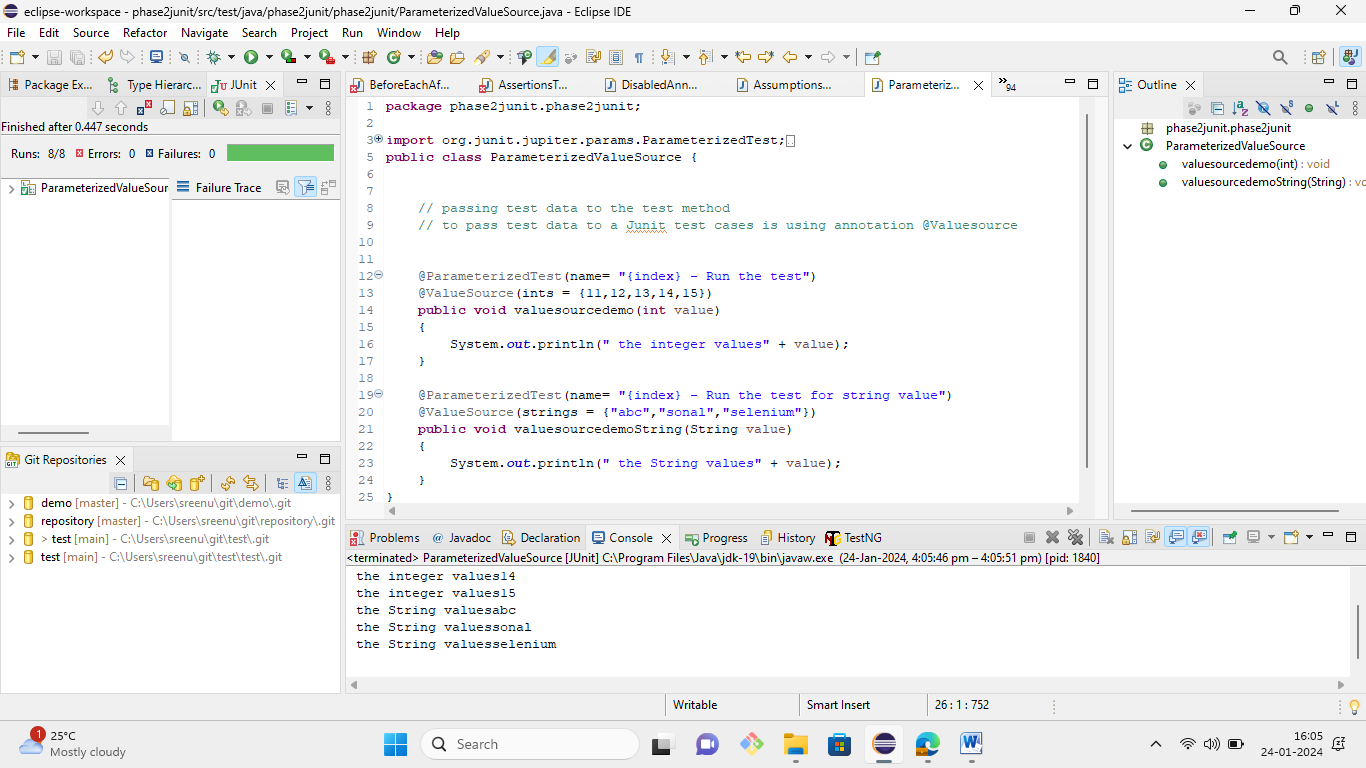


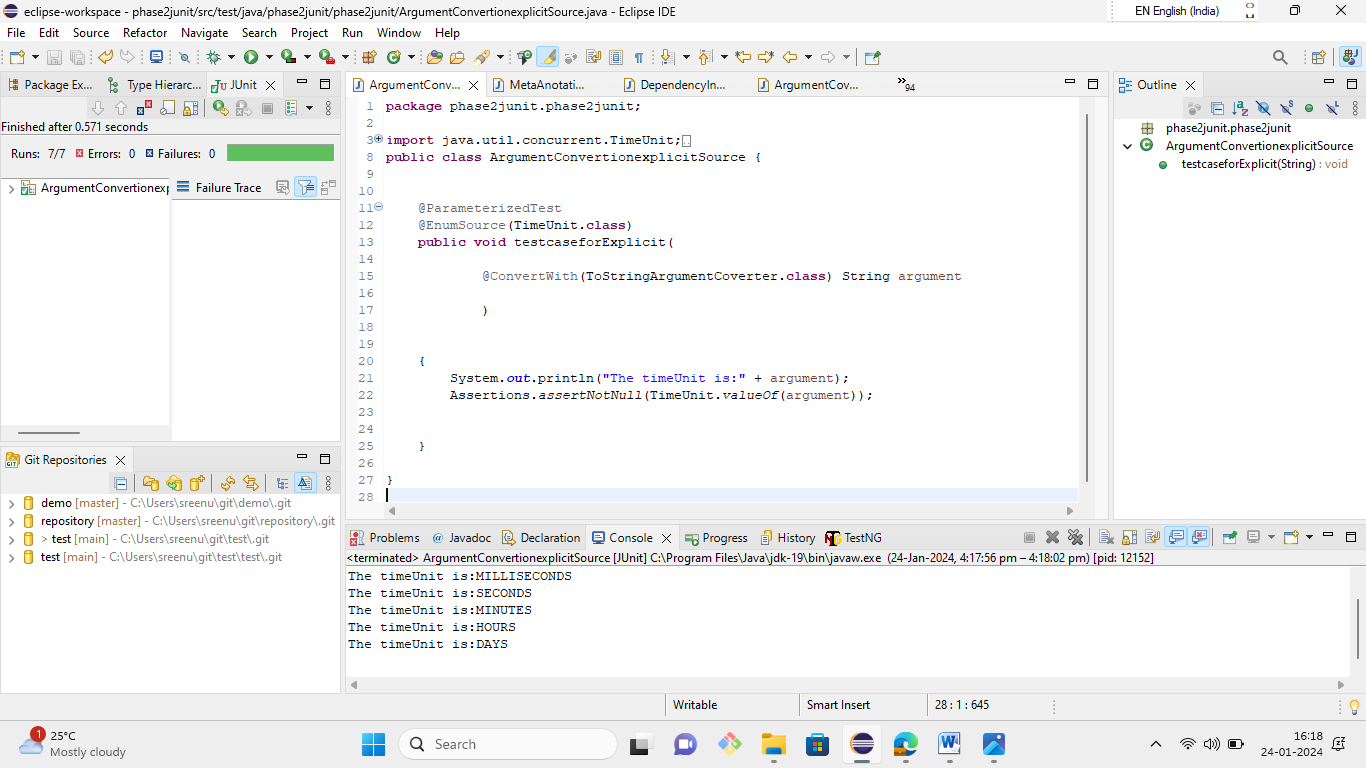


1. Write a JUnit program to demonstrate how dynamic tests are created in JUnit.

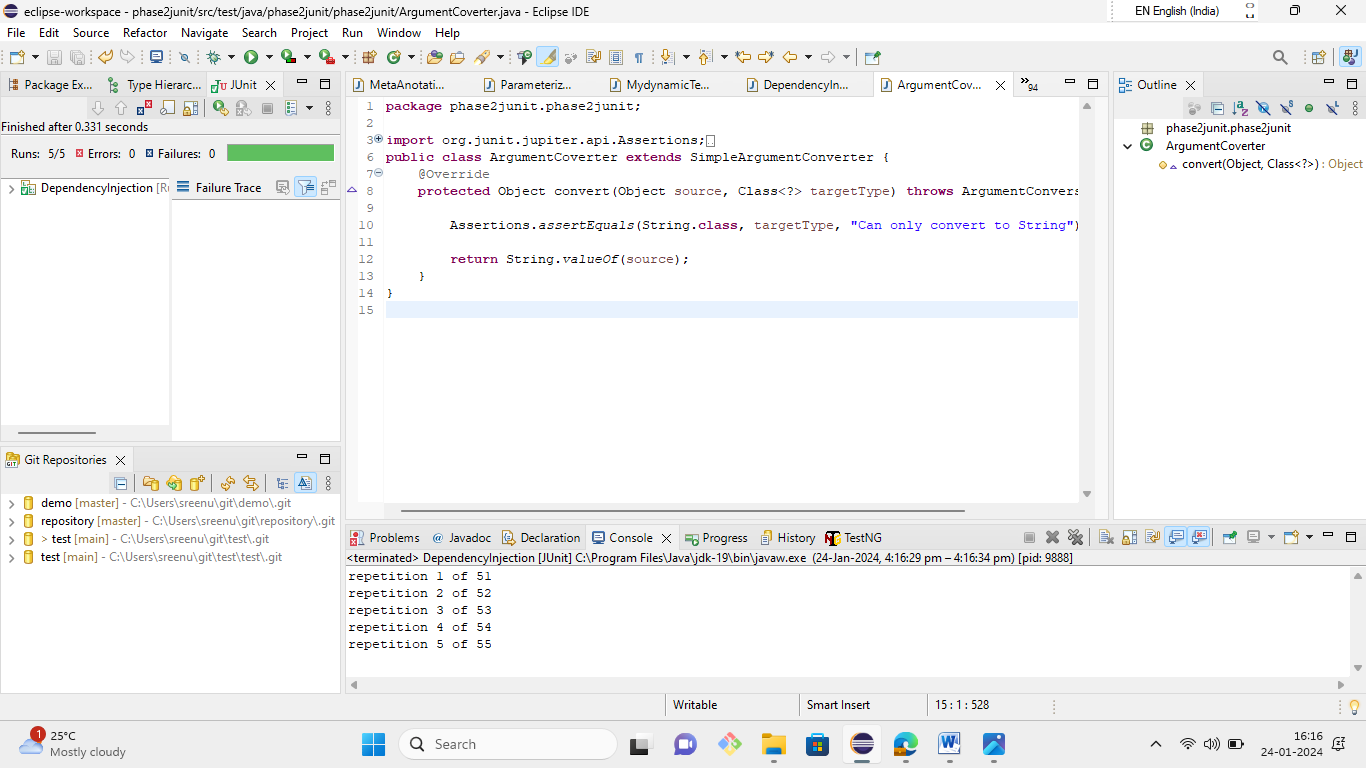


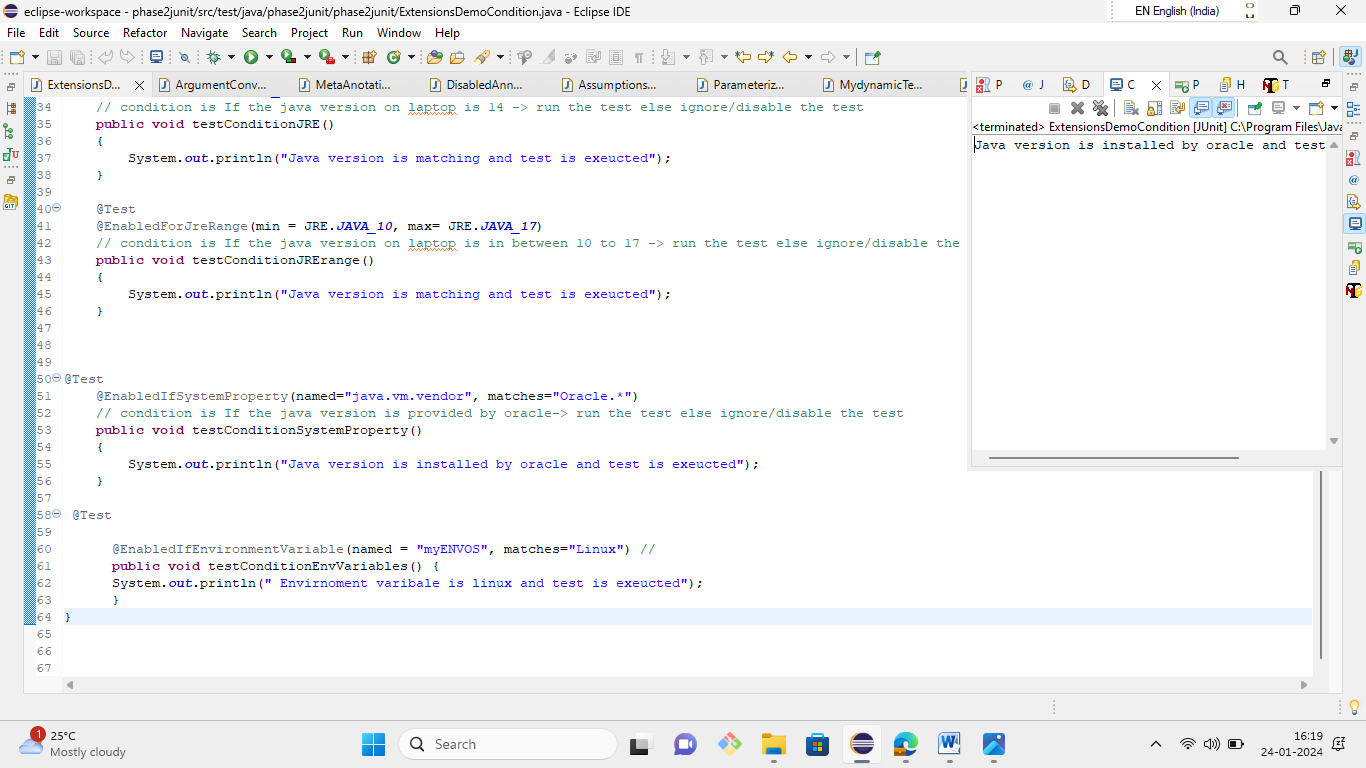
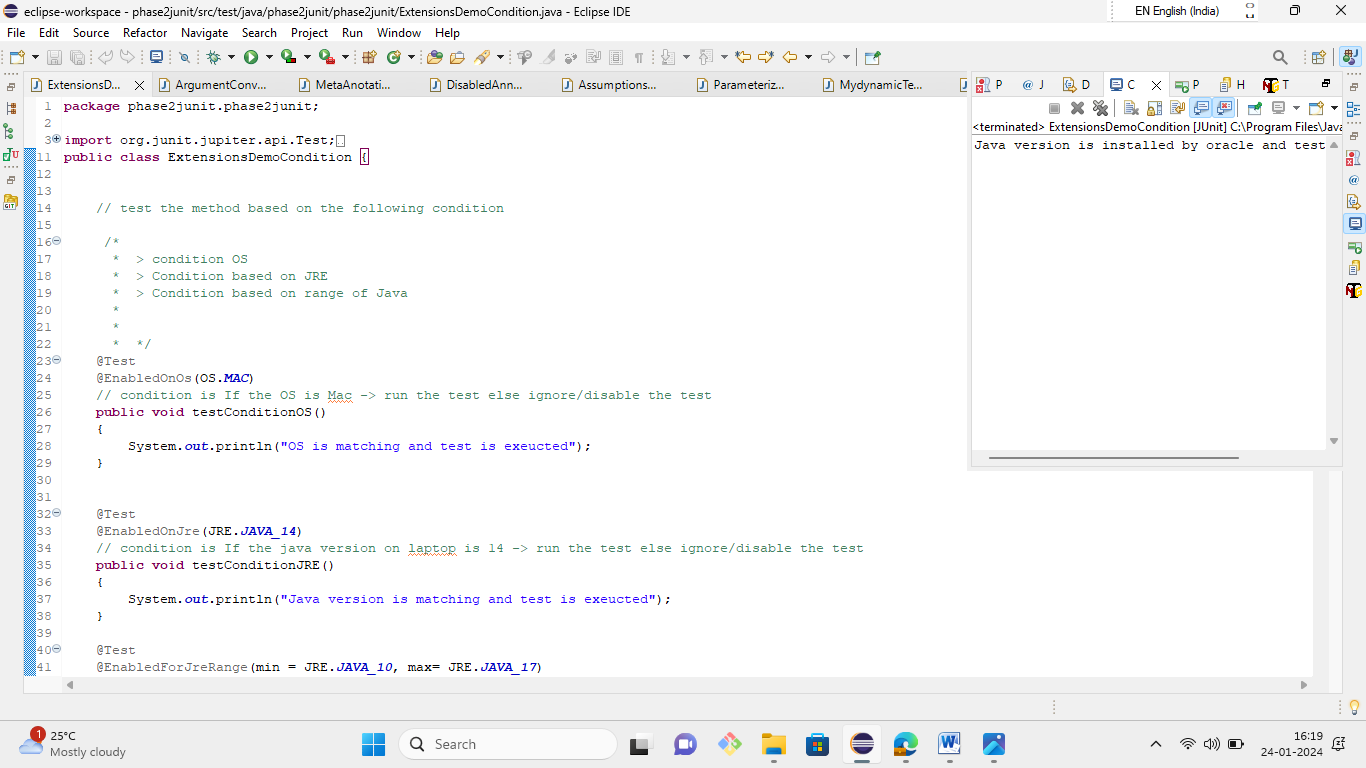
1. Write a JUnit program to demonstrate how parameterized tests are created in JUnit.



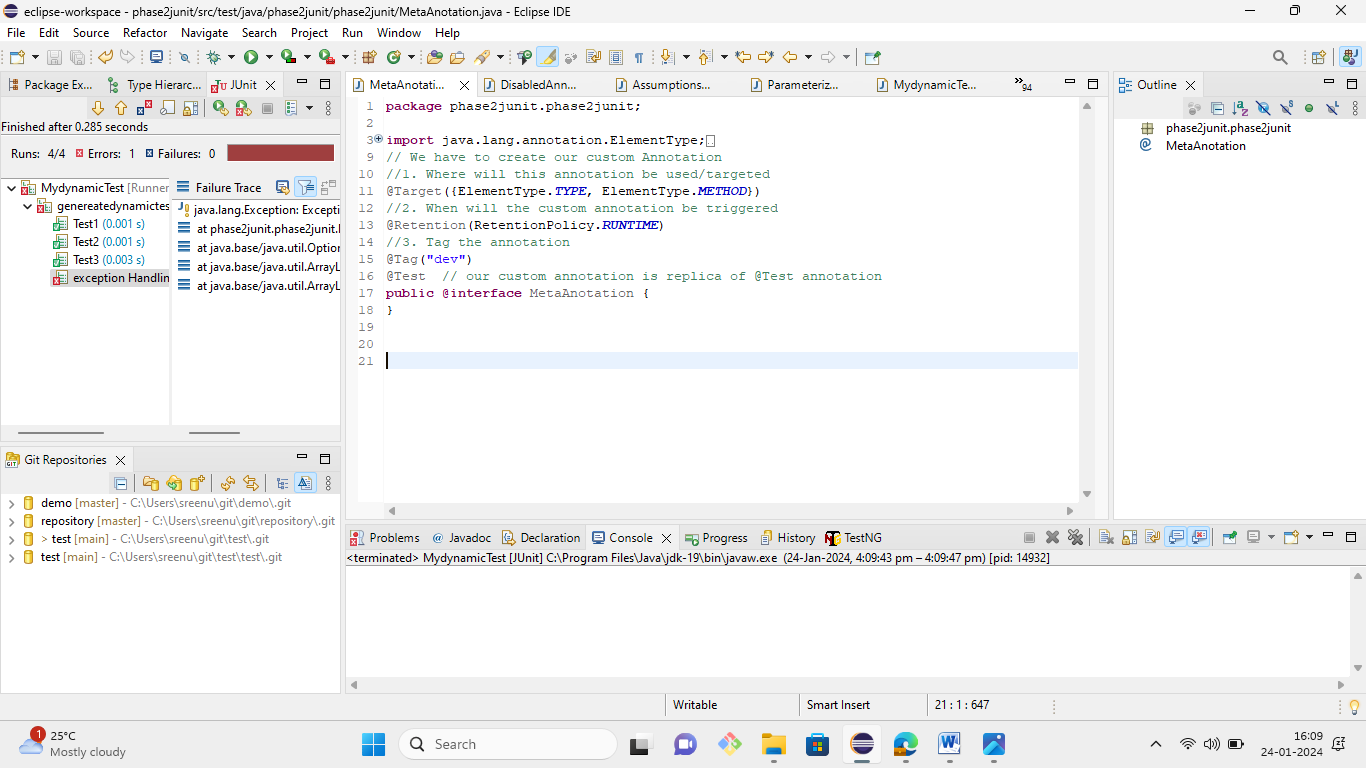
9. Write a JUnit program to demonstrate how argument sources are used in in JUnit.. 

10.Write a JUnit program to demonstrate argument conversion in JUnit.

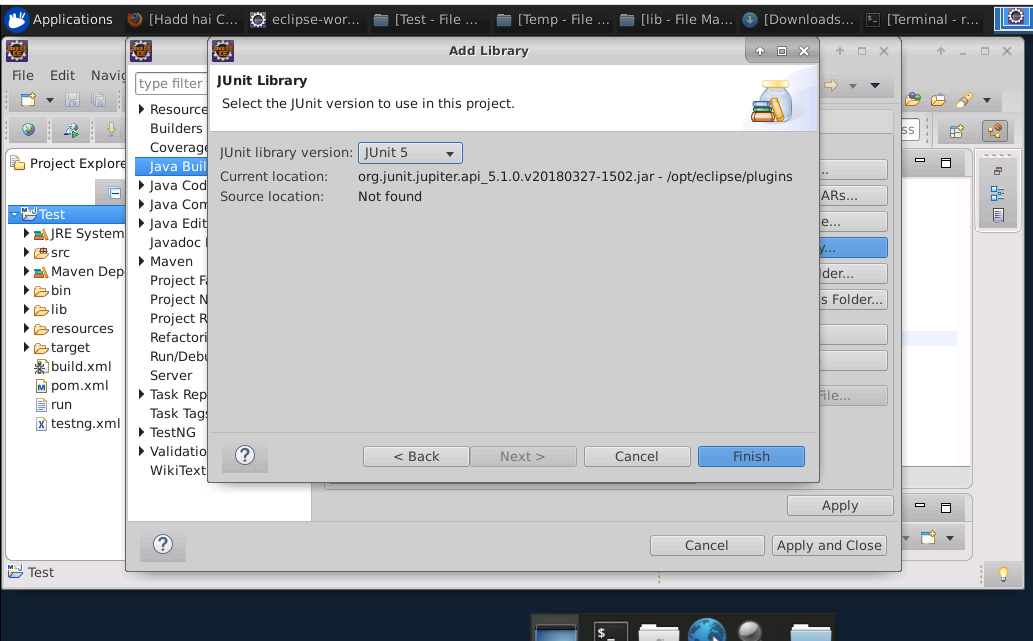


11.Write a JUnit program to demonstrate extension points. 

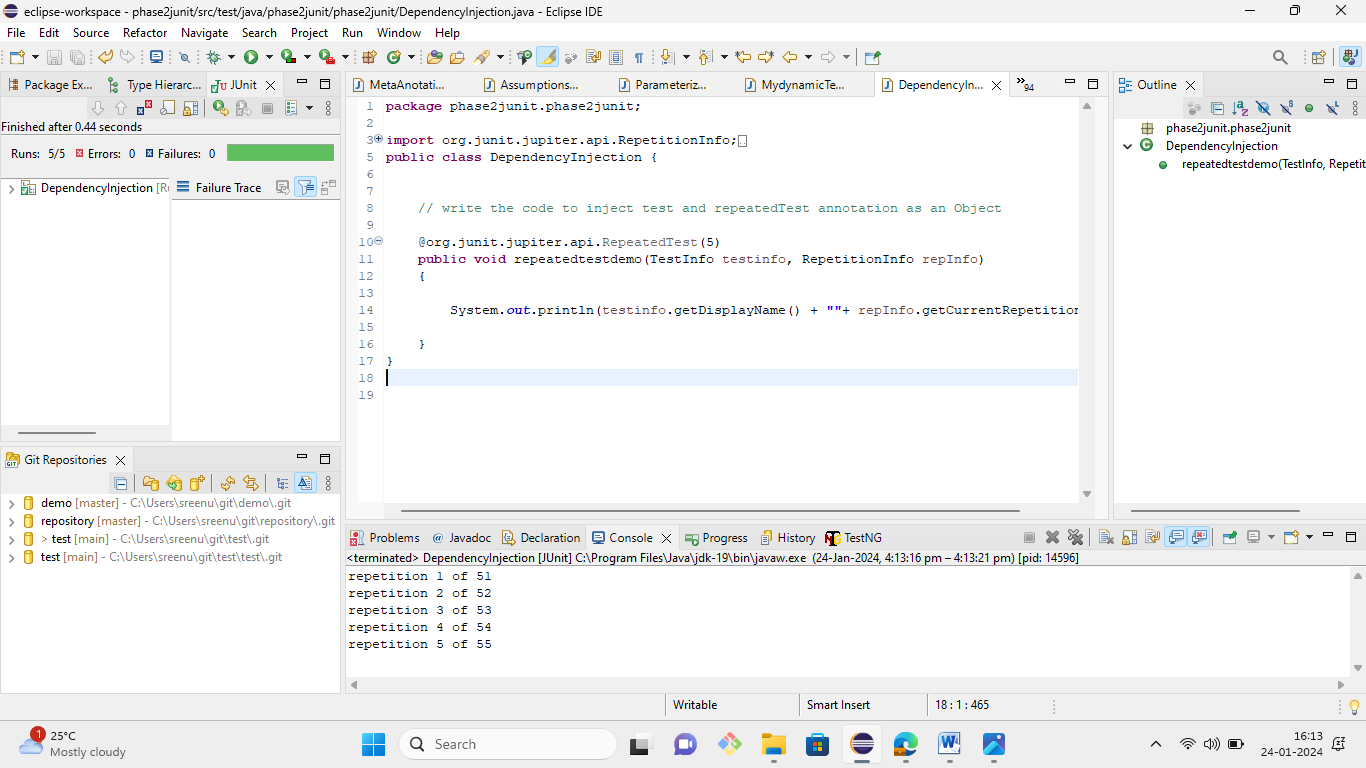
12.Write a JUnit program to demonstrate meta-annotation.



13.Write a JUnit program to demonstrate how tests are run from the console.



14.Write a JUnit program to demonstrate how tests are run on Gradle.



15.Write a JUnit program to demonstrate how tests are run on Maven.

MagicBuilder.java

**package** SSS.Junit;  
public class MagicBuilder {

public static int getLucky() {

return 7;

}

}

Textbuild.java

**package** SSS.Junit;  
  
public class Textbuild {

public static String getHelloWorld(){

return "hello world";

}

public static int getNumber10(){

return 10;

}

}

**4.17.4.2 Test class for MagicBuilder**

TestMagicBuilder.java

**package** SSS.Junit;  
  
import org.junit.jupiter.api.Test;

import static org.junit.jupiter.api.Assertions.assertEquals;

public class TestMagicBuilder {

@Test

public void testLucky() {

assertEquals(7, MagicBuilder.getLucky());

}

}

* + - 1. **Test class for Textbuild**

pom.xml

**package** SSS.Junit;  
  
import org.junit.jupiter.api.Test;

import static org.junit.jupiter.api.Assertions.assertEquals;

public class Msgbuild {

@Test

public void testHelloWorld() {

assertEquals("hello world", Textbuild.getHelloWorld());

}

@Test

public void testNumber10() {

assertEquals(10, Textbuild.getNumber10());

}

}

## **Step 4.17.5:** Executing Maven Test

* While we run the POM and class files with Maven, the below results will be generated at the console:
  + - 1. **Run all test classes**

$ mvn test

[INFO] -------------------------------------------------------

[INFO] T E S T S

[INFO] -------------------------------------------------------

[INFO] Running SSS.Junit.TestMagicBuilder

[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.004 s - in SSS.Junit.TestMagicBuilder

[INFO] Running SSS.Junit.Msgbuild

[INFO] Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.001 s - in SSS.Junit.Msgbuild

[INFO]

* + - 1. **Run a single test class Msgbuild**

Terminal

$ mvn -Dtest=Msgbuild test

[INFO] -------------------------------------------------------

[INFO] T E S T S

[INFO] -------------------------------------------------------

[INFO] Running SSS.Junit.Msgbuild

[INFO] Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.004 s - in SSS.Junit.Msgbuild

[INFO]

* + - 1. **Run a single test method testHelloWorld() from the test class Msgbuild**

Terminal

$ mvn -Dtest=Msgbuild#testHelloWorld test

[INFO] -------------------------------------------------------

[INFO] T E S T S

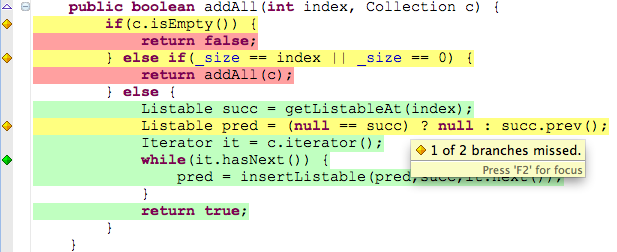
[INFO] -------------------------------------------------------

[INFO] Running SSS.Junit.Msgbuild

[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.004 s - in SSS.Junit.Msgbuild

[INFO]

16.Write a JUnit program to demonstrate code coverage



17.Write a JUnit program to demonstrate how tests with tags are included or excluded. 