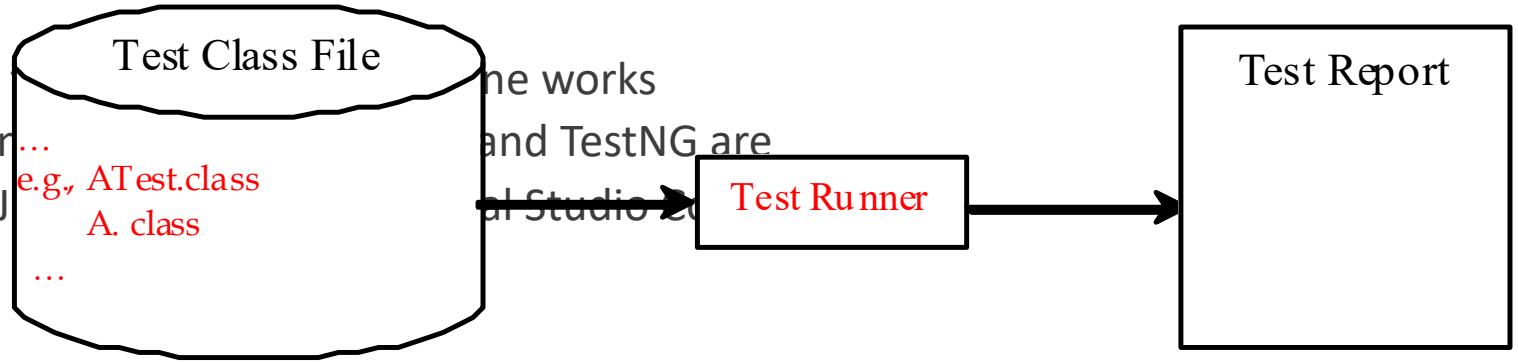
Testing Java

WITH JUNIT AND TESTNG

Objectives

- To know what a test [illegible] ne works
- To know the test fram[illegible] and TestNG are
- To create and run a J[illegible] al Studio [illegible]



Introduction

- The software development process includes requirements specification, analysis, design, implementation, testing, deployment, and maintenance.
- Testing is an essential practice to ensure the quality and reliability of your code.
- Test frameworks
  - provide a structured and systematic approach to writing, organizing, and running tests for your Java applications
  - help you automate the testing process, making it easier to catch bugs and ensure your code functions as expected
- There are two popular Java test frameworks: JUnit and TestNG

Test Frameworks

Test frameworks are tools that simplify the creation, execution, and management of test cases.

They provide a structured way to:

- Define test cases
- Automate test execution
- Report results
- Customize testing behavior

Testing Basics

- The test framework library is packaged in a jar file
- The jar file contains a tool called ***test runner***, which is used to run test programs
- You have a class named **`Process`**
- To test this class, you write a test class named **`ProcessTest`**
- This test class, called a test runner, contains the methods you for testing the class **`Process`**

The Testing Process :: Setting up a framework

1. Project Setup Create or Open a Java Project
2. Add Test Framework Dependency
3. Write Test Cases
    a. Create a new Java class(es) for writing test cases. Methods are annotated.
    b. Annotated test methods use assertions to verify code correctness
4. Running Tests - VS Code provides built-in support for running tests and display the results
5. Analyzing Test Results
    a. Test frameworks generate test reports
    b. Analyze the test results. Identify errors

Visual Studio Code Project Setup

- VS Code is enabled by the Test Runner for Java extension
- VS Code Extension Pack for Java includes the Test Runner extension
- Test Runner supports JUnit 4, JUnit 5 and TestNG
- Testing Explorer can enable a test framework for your unmanaged folder project
  (a project without any build tools)
- You can also enable a test framework for Gradle and Maven

To Run/Debug test cases see Run/Debug test cases

See also

- Testing Explorer
- View test results
- Generate tests

See Enable testing and adding test framework JARs to your project

What are annotations?

Annotations

- A form of metadata added to Java code
- Provide additional information about the code's behavior, structure, or characteristics
- Do not directly affect the execution;
- Used by tools, frameworks, and libraries to
  - configure, process, or generate code automatically
  - typically introduced with the @ symbol.

Key points about annotations

1. **Metadata:** used for various purposes such as documentation, validation, and code generation
2. **Predefined and Customizable**
   - Java provides a set of predefined annotations like **@Override**, **@Depr**ecated, and **@SuppressWarnings**
   - Developers can create custom annotations to suit specific requirements
3. **Retained at Runtime (if specified):**
   - some annotations can be configured to be retained using the **@Retention** annotation with the RUNTIME retention policy
   - allows runtime inspection and processing of annotations by reflection.
4. **Have Target Elements**
   - can be applied to specific program elements, e.g., **@Override** is typically applied to methods
5. **Can Have Values:** accept values or parameters that provide additional information or configuration; values can be primitive types, strings, enums, classes, or arrays
6. **Can Be Used for Code Generation:** Many Java frameworks and tools use annotations to generate code or configuration files automatically
7. **Processed by Annotation Processors**
   - Tools that read and process annotations during compilation
   - Can generate code, perform validation, or generate configuration files based on the annotations present in the code

A crucial role

- Annotations enable
  - better documentation,
  - code analysis, and
  - automation through various tools and frameworks
- Enhance code readability and maintainability by providing context and intent within the codebase.
- An integral part of testing frameworks

JUnit 4 & JUnit 5

JUnit 4

- A widely used Java testing frameworks
- Provides a simple and efficient way to write and execute tests

JUnit 5

- next iteration of JUnit
- Introduces many new features and improvements
- Offers greater flexibility and extensibility.

Annotations

JUnit

@Test: Indicates that the method is a test case

@Before: Executed before each test method.

@After: Executed after each test method.

@BeforeClass: Executed once before all test methods in the class.

@AfterClass: Executed once after all test methods in the class.

@Ignore: Skips the annotated test method.

TestNG

@Test: Marks a method as a test case

@BeforeMethod: Method to be run before each test method in the test class, used for setup

@AfterMethod: Method to be run after each test method in the test class, used for cleanup.

@BeforeClass: Method to be run once before any test methods in the test class, used for one-time setup.

@AfterClass: Method to be run once after all test methods in the test class, used for one-time cleanup

@Ignore: A test method to be ignored during test execution

```
import org.junit.*;
import static org.junit.Assert.*;
import java.util.*;
public class ArrayListTest {
  private ArrayList<String> pastaTypes = new ArrayList<String>();

  @Before
  public void setUp() throws Exception {
  }

  @Test
  public void testInsertion() {
    pastaTypes.add(”Rigatoni");
    assertEquals(”Rigatoni", pastaTypes.get(0));
    pastaTypes.add(”Macaroni");
    list.add(”Linguine");
    assertEquals(”Linguine", list.get(list.size() - 1));
  }

  @Test
  public void testDeletion() {
    pastaTypes.clear();
    assertTrue(list.isEmpty());

    pastaTypes.add(”Orzo");
    list.add(”Farfalle");
    list.add(”Spaghetti");
    list.remove(”Farfalle");
    assertEquals(2, list.size());
  }
}
```

Additional reading & video

- JUnit 4
- JUnit 5
- TestNG
- How to Setup JUnit for VS Code | JUnit in Visual Studio Code | Java Test with Visual Studio Code