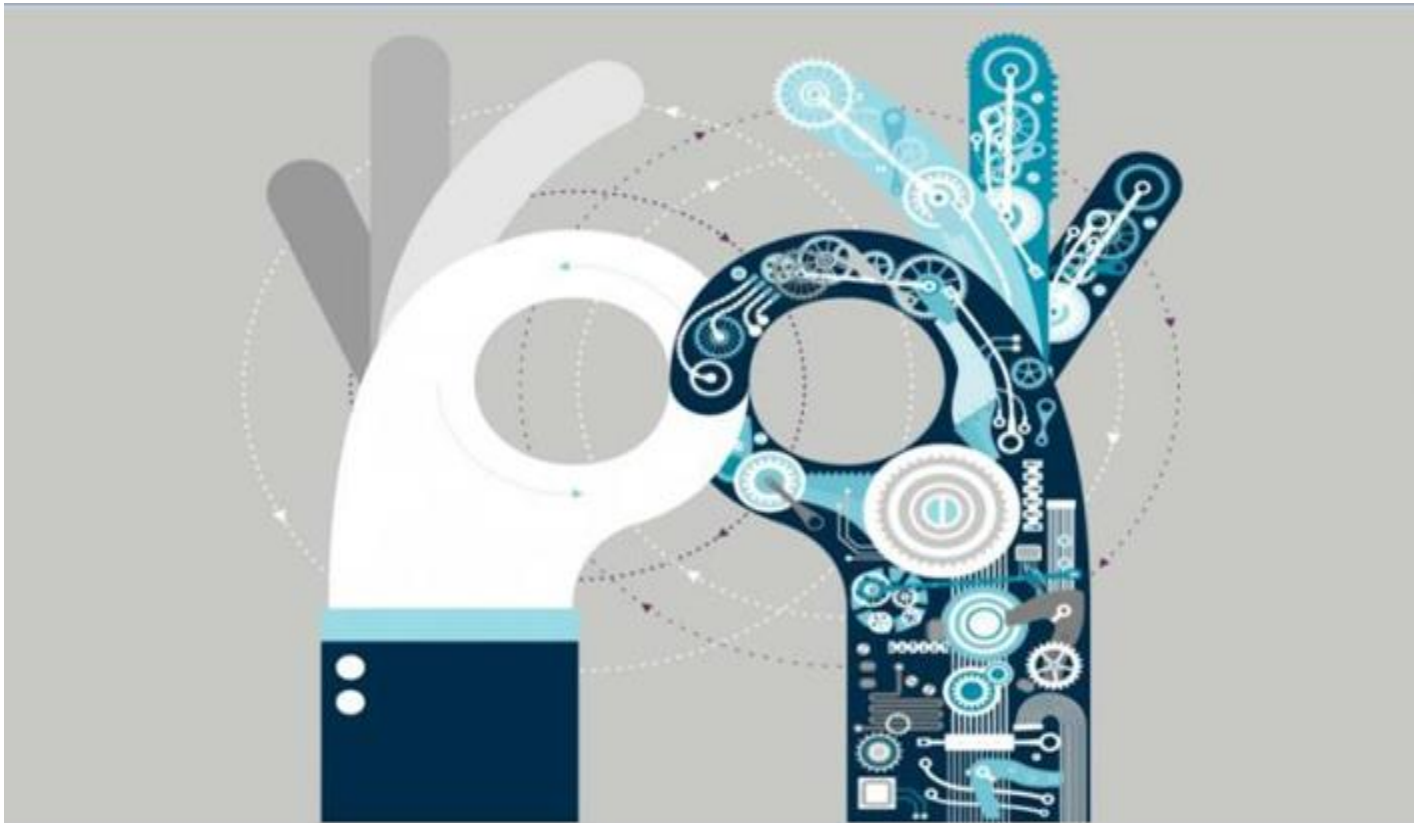


*Welcome to the world of software testing*



# *Agenda*

- Introduction to Automation Testing
- Manual Testing Vs Automation Testing
- When do we need Automation?
- Limitations of Automation Testing
- How to choose right tool for automation?
- Introduction to Selenium
- Components of Selenium
- Introduction to Selenium WebDriver
- Installation of Selenium
- WebElements
- Locators
- CSS Selector
- XPath - Absolute & Relative
- Waits in Selenium

# AUTOMATION TESTING

Process of converting manual test cases into the test scripts with the help of automation tools or any programming language

## *When we can go for Automation Testing?*

- ✓ When the cost makes sense
- ✓ When using repetitive tests
- ✓ When time will be saved
- ✓ When quality is sure to be improved
- ✓ When tests are run frequently
- ✓ When you need to run multiple tests at once

# Manual Vs Automation

- Test cases are executed manually.
- Difficult to ensure sufficient test coverage
- May difficult to test on different browsers
- you need to sit in front of your system and execute test cases
- Test cases are executed automatically with the help of tools.
- Easy to ensure greater test coverage.
- We can easily test on different
- You just have to run Automation scripts you can run it overnight!

## *Limitations of Automation Testing*

- Cannot perform testing for Images.
- Captcha, Barcodes.
- Continues maintenance of code.
- Cannot perform testing for audio or video

# Tools



# TOOLS FOR AUTOMATION TESTING



Web

Windows,  
Mac OS,  
Linux

Java, C#,  
Python,  
JavaScript,  
Ruby, PHP,  
Perl



Web, API,  
Mobile,  
Desktop

Windows,  
Mac OS,  
Linux

Java &  
Groovy



Mobile  
(iOS,  
Android)

Windows,  
Mac OS

Java, C#,  
Python,  
JavaScript,  
Ruby, PHP,  
Perl



Web

Windows,  
Mac OS,  
Linux

JavaScript



Web,  
Mobile,  
Desktop

Windows

C++,C#,  
Python,  
JavaScript,  
VBScript,  
Jscript, Delphi



# SELENIUM & ITS FEATUES

**Selenium** is a free automated testing tool

## Features

- Open Source
- Supports Multiple operating system
- Supports multiple browsers
- Supports multiple programming languages
- Supports multiple framework
- Supports parallel and cross browser execution

# How to select right tool for Automation?

- Project Requirements
- Team skills / Learning Curve
- Budget
- Ease of Test case Creation and Maintenance
- Reusability
- Data-Driven Testing
- Reporting
- Support for Collaboration

# SELENIUM COMPONENTS

Selenium



Selenium IDE



Selenium RC

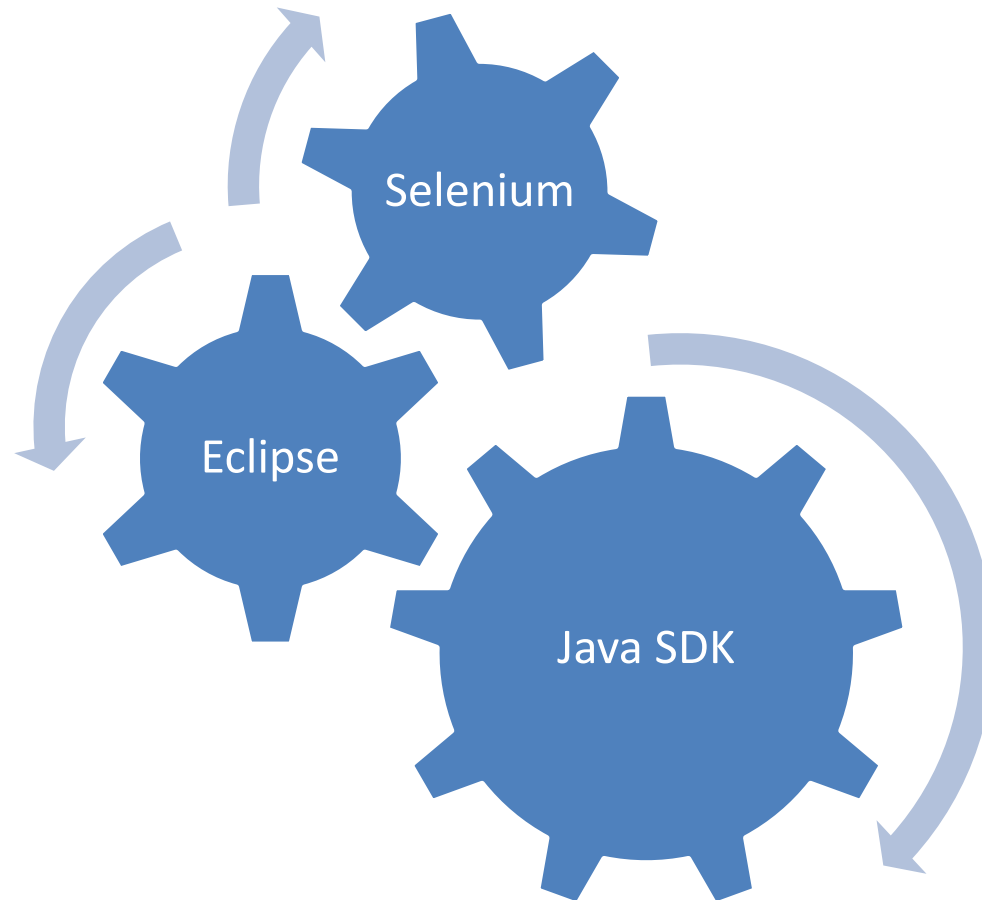


Selenium Grid



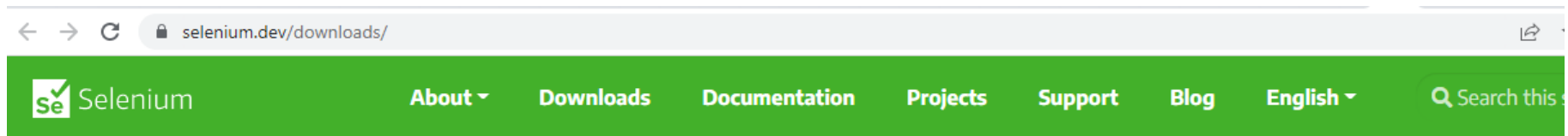
Selenium Web Driver

# PREREQUISITES - TO USE SELENIUM



# INSTALLATION

Navigate to <https://selenium.dev/downloads/> and check for **Selenium Clients and Web Driver Language Bindings**



## Selenium Clients and WebDriver Language Bindings

In order to create scripts that interact with the Selenium Server (Remote WebDriver) or create local Selenium WebDriver scripts, you need to make use of language-specific client drivers.

While language bindings for [other languages exist](#), these are the core ones that are supported by the main project hosted on GitHub.



C#

Stable: [4.1.0 \(November 22, 2021\)](#)

[Changelog](#)

[API Docs](#)



Ruby

Stable: [4.1.0 \(November 22, 2021\)](#)

[Changelog](#)

[API Docs](#)

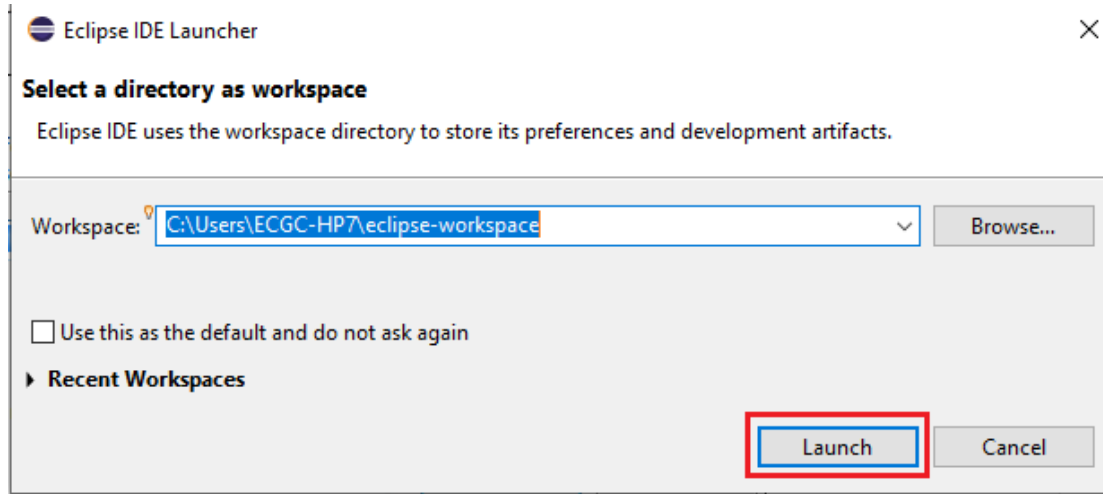


Java

Stable: [4.1.2 \(January 30, 2022\)](#)

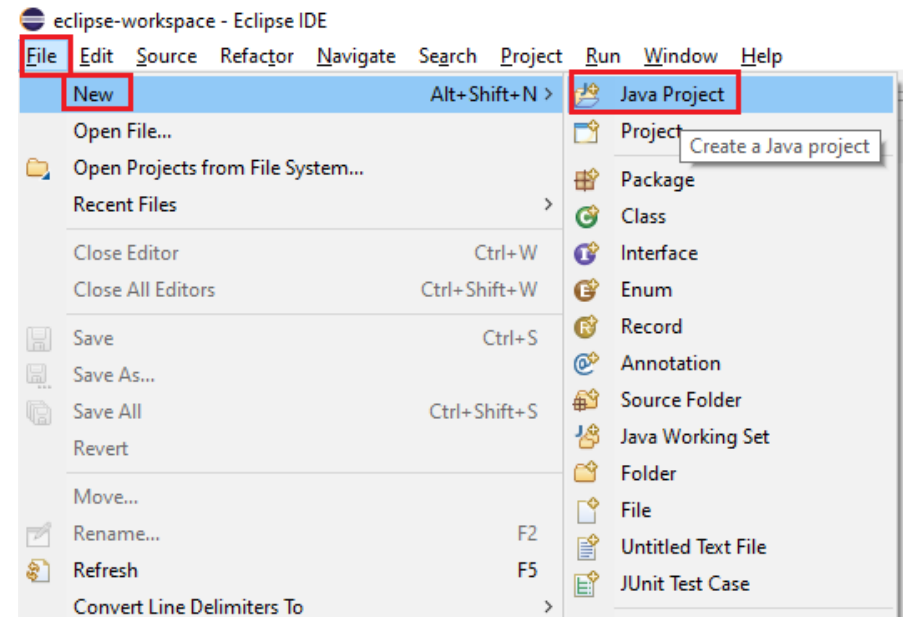
[Changelog](#)

[API Docs](#)

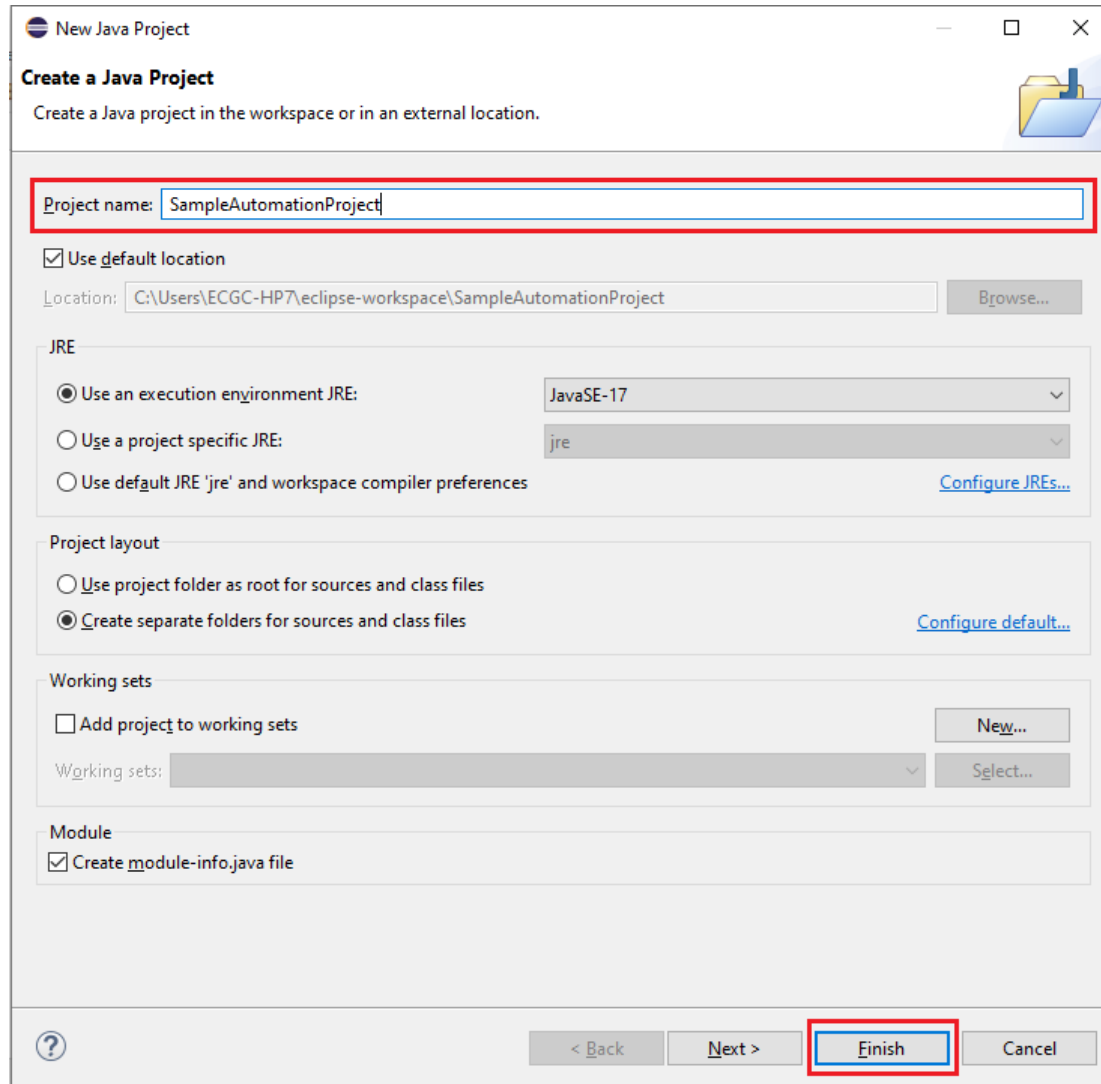


1. Launch **Eclipse** and select the default workspace. Click on **Launch**

2. Click on **File -> New -> Java Project**



### 3. Enter a **Project name** and click on **Finish**.



**New Java Project**

**Create a Java Project**

Create a Java project in the workspace or in an external location.

Project name:

☒ Use default location

Location:  [Browse...](#)

**JRE**

☒ Use an execution environment JRE:

☐ Use a project specific JRE:

☐ Use default JRE 'jre' and workspace compiler preferences [Configure JREs...](#)

**Project layout**

☐ Use project folder as root for sources and class files

☒ Create separate folders for sources and class files [Configure default...](#)

**Working sets**

☐ Add project to working sets [New...](#)

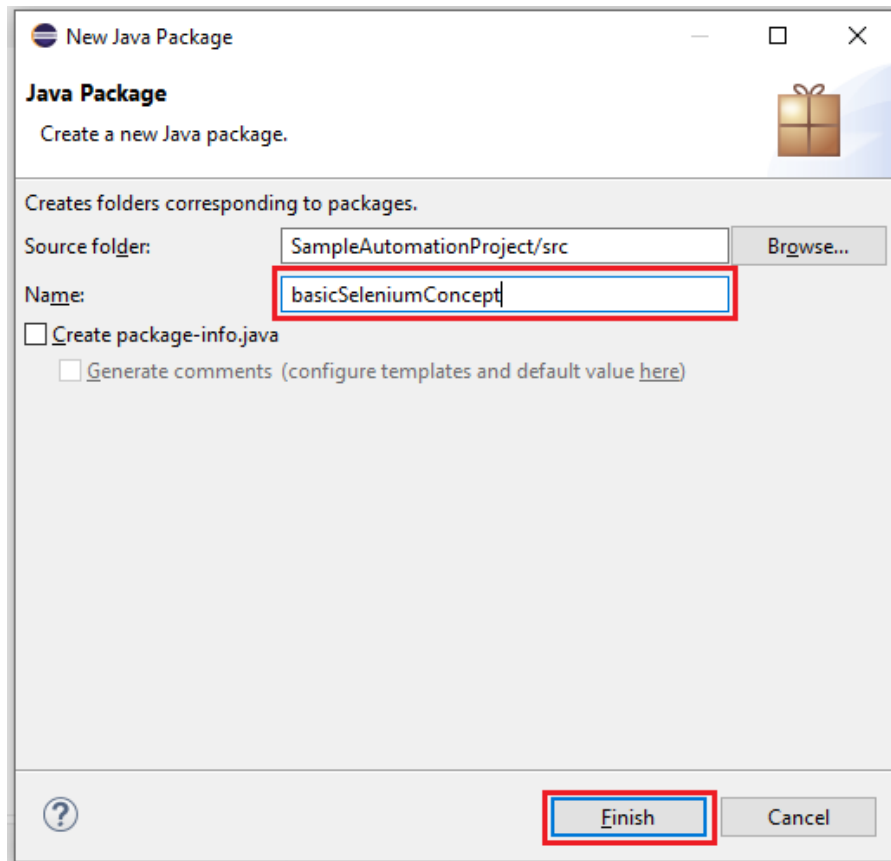
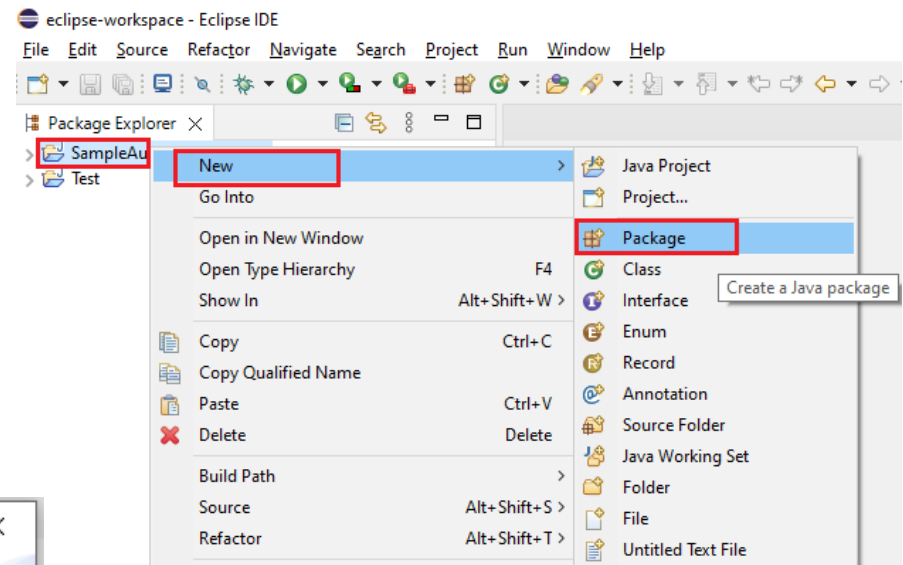
Working sets:  [Select...](#)

**Module**

☒ Create module-info.java file

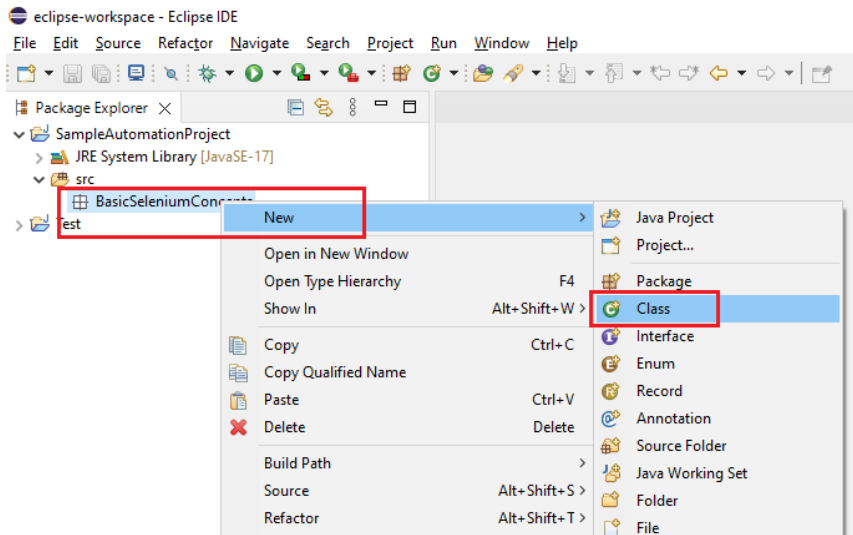
[?](#) [< Back](#) [Next >](#) **Finish** [Cancel](#)

4. **Right Click** on the newly created Project, then **New -> Package**



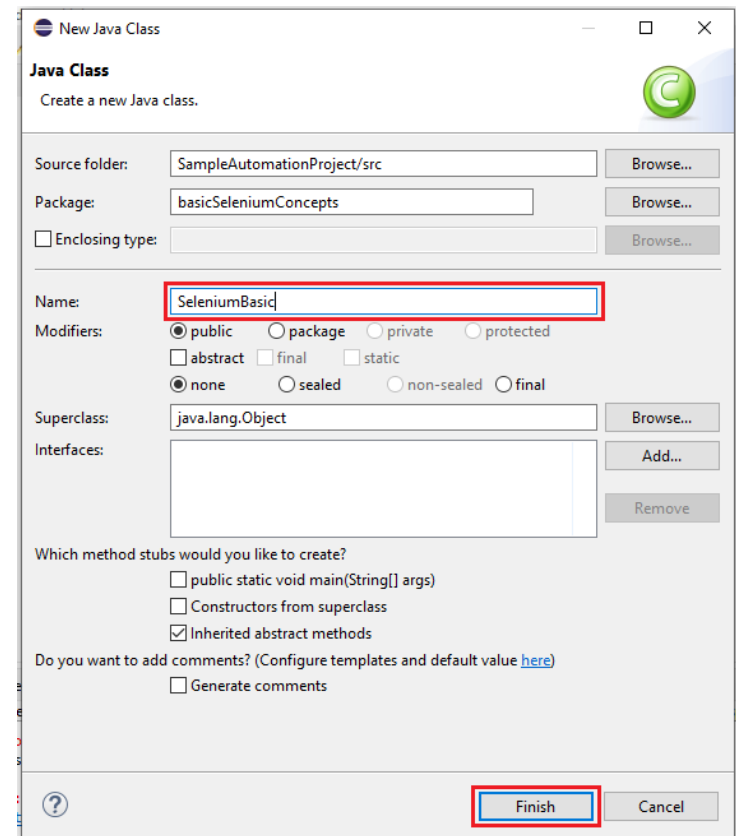
5. Enter **Package Name** and click on **Finish**.



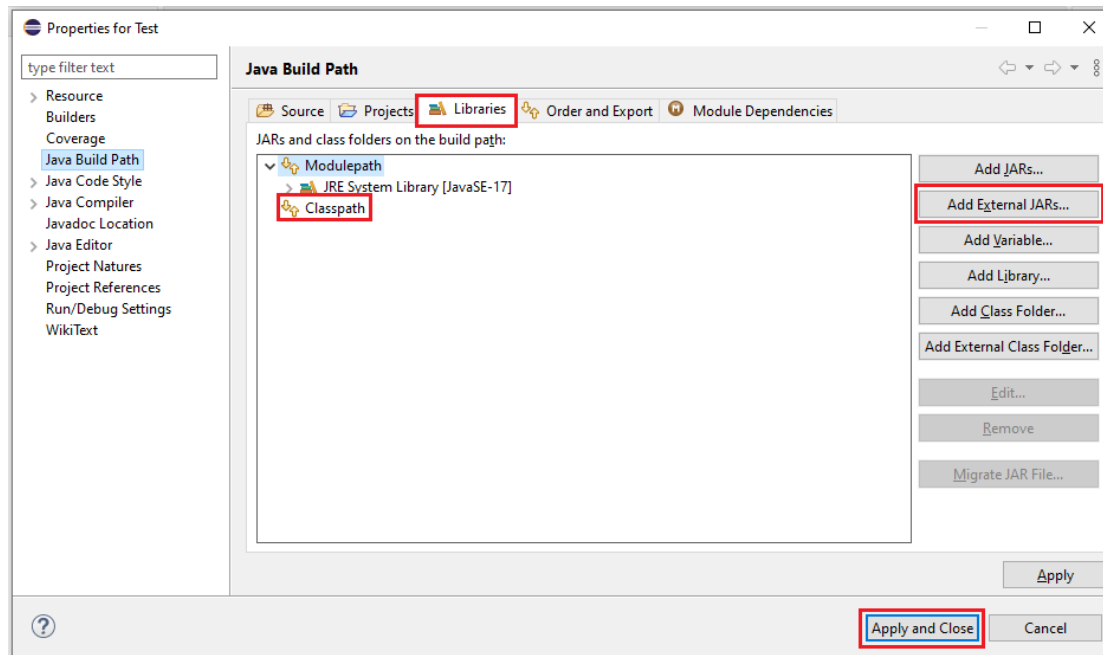
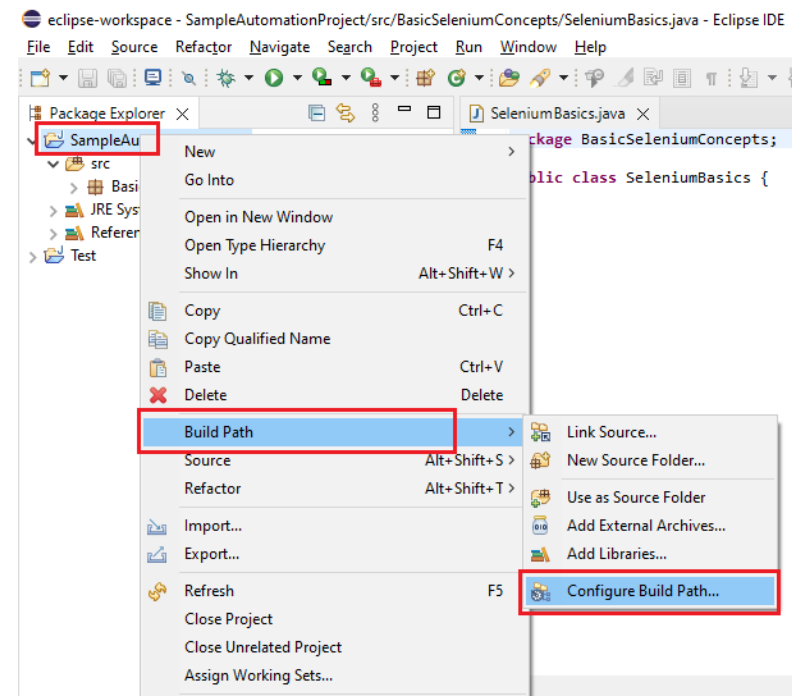


7. Enter **Class Name** and then click on **Finish**.

6. **Right Click** on newly created package and then **New -> Class**

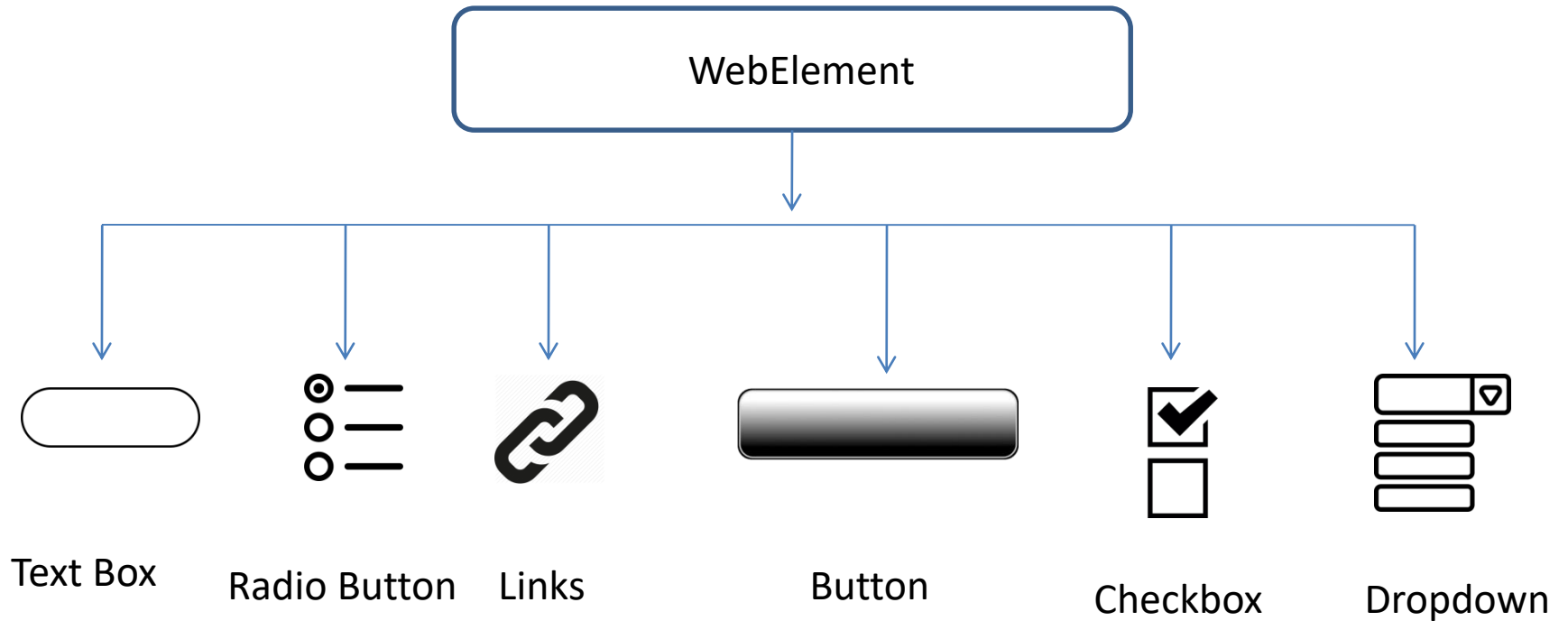


8. Right Click on project folder and then Build Path -> Configure Build Path

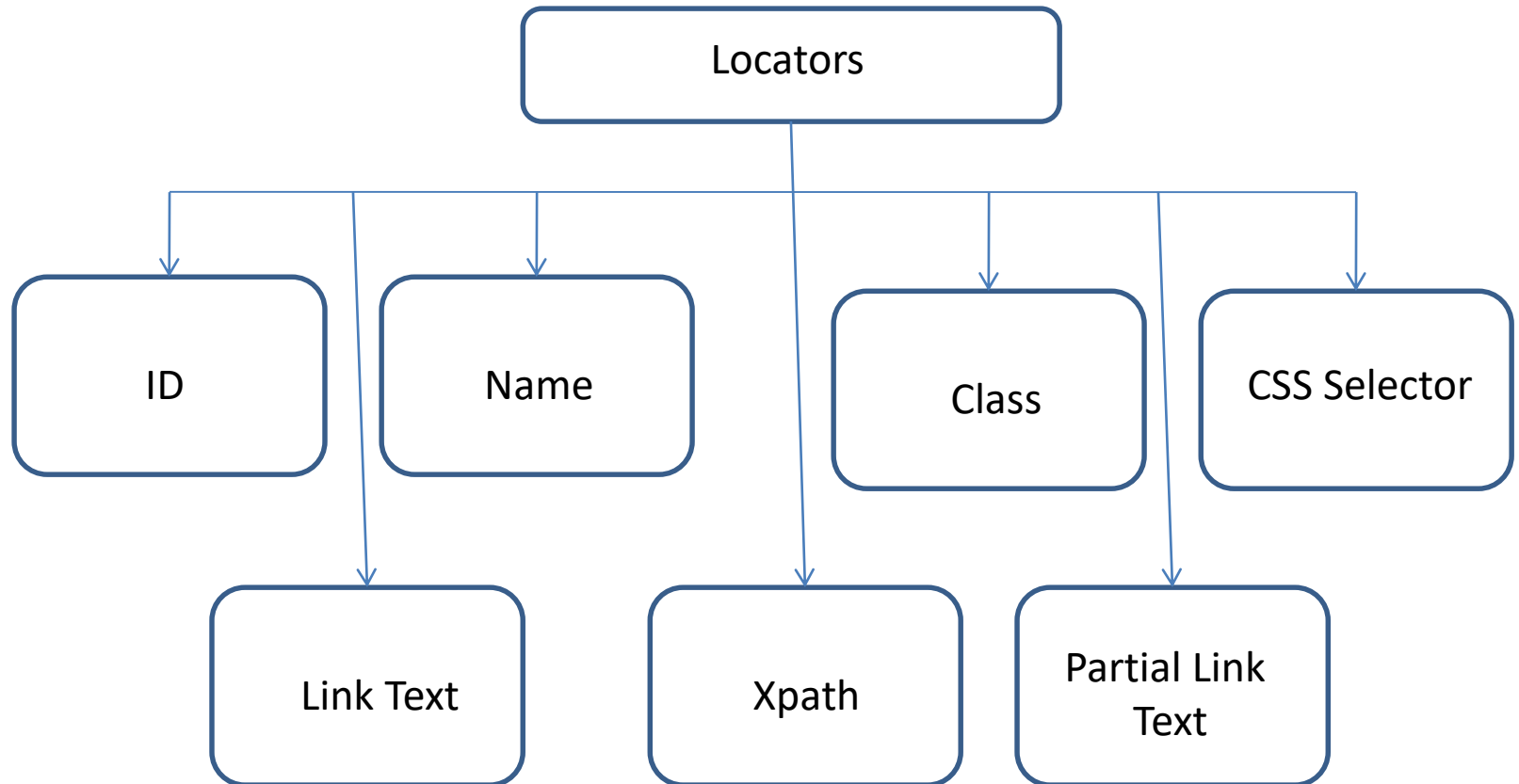


9. Click on **Libraries** and then Select **Classpath**, then click On **Add External JARs**. Add the downloaded Selenium Jar files and then **Apply and Close**.

# WEBELEMENT & ITS TYPES



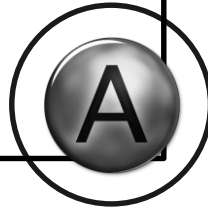
# LOCATORS & ITS TYPES



# XPATH

- Direct way to select an element
- Starts from the root node
- Uses single slash ( / )

**Absolute  
XPath**



- Easy way to select an element
- Starts from middle of HTML DOM structure
- Uses double slash ( // )

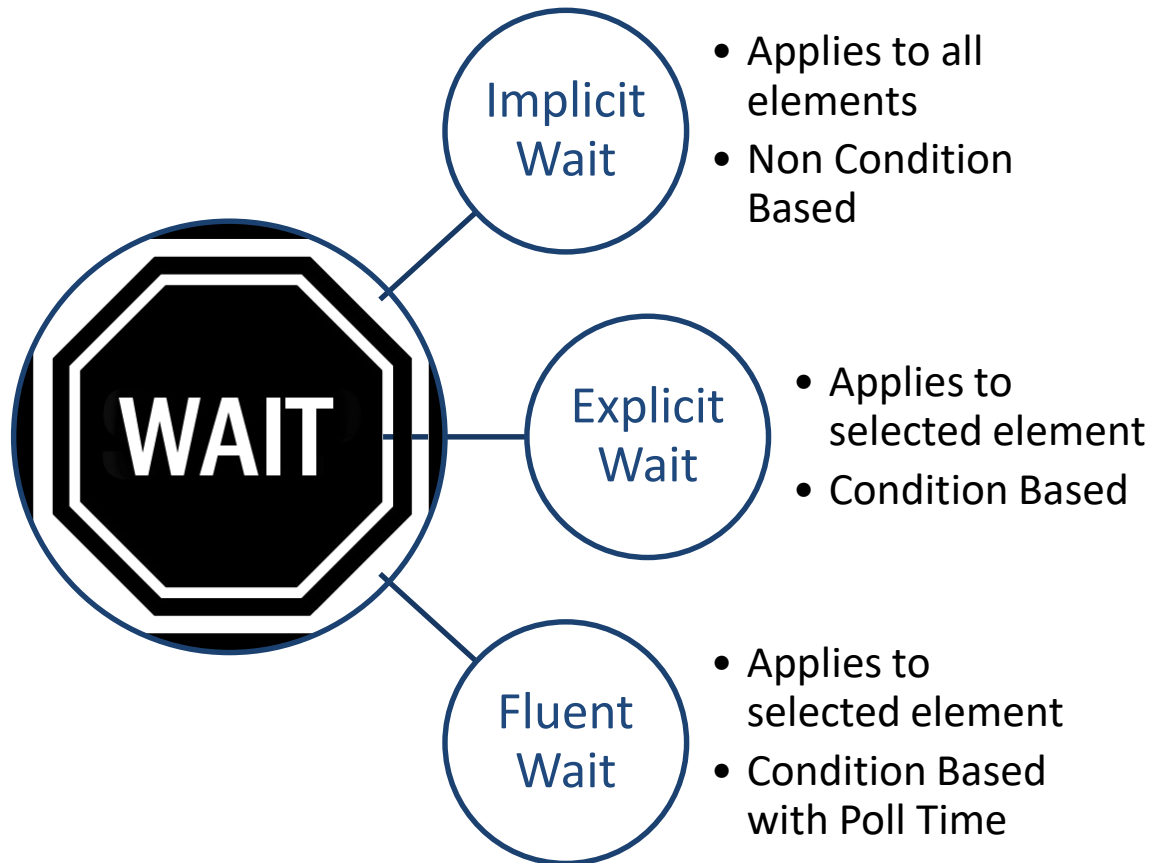
**Relative  
XPath**



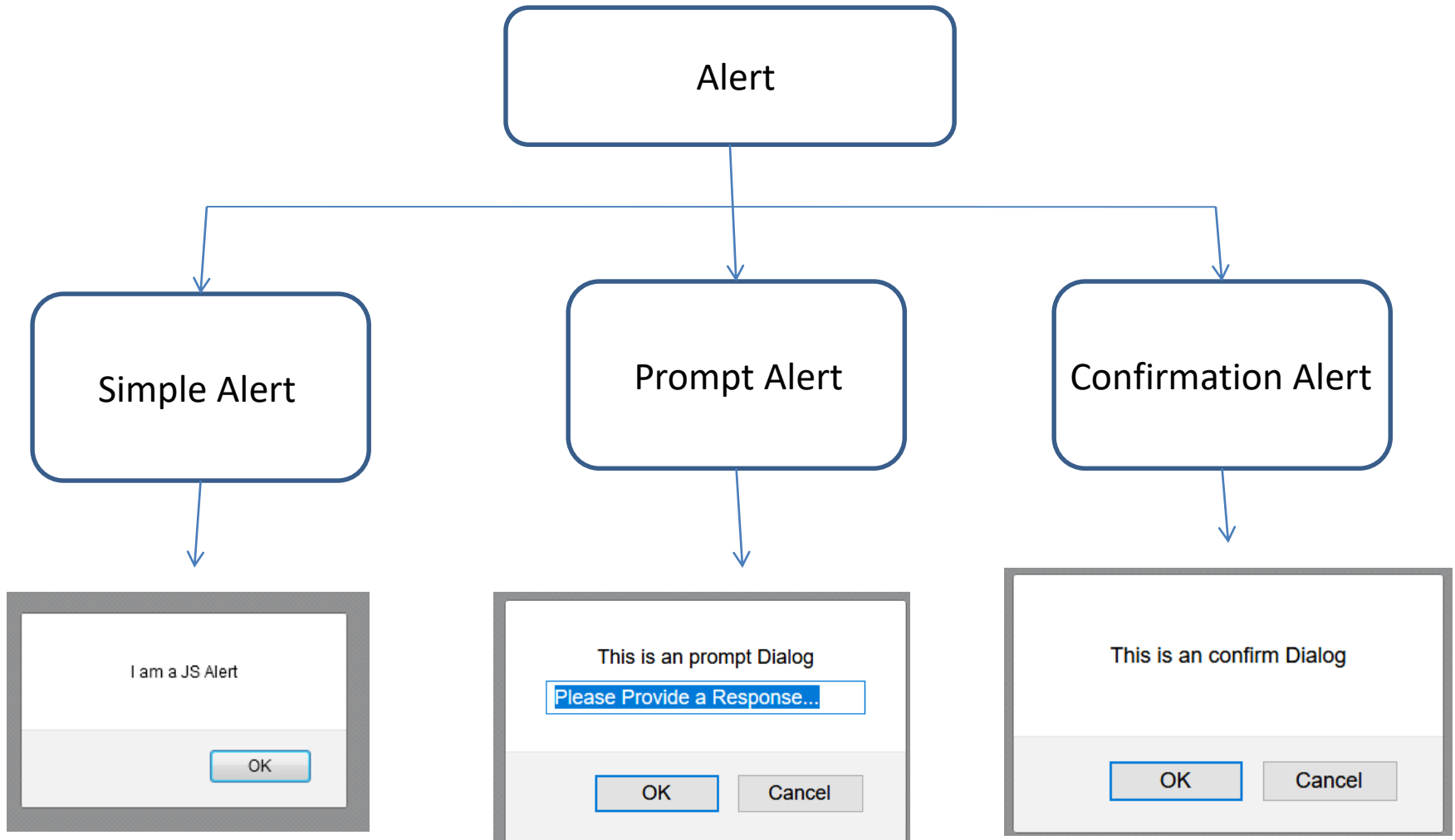
# RELATIVE XPATH

- Basic Xpath
- Contains
- OR & AND
- Starts-with
- Text()
- Last()
- Position()
- Following
- Preceding

# WAITS



# ALERTS & ITS TYPES





# HANDLING WINDOWS & FRAMES

- Selenium helps to handle multiple windows through **window handlers** and **javascript executors**.
- Window Handling is a unique identifier that holds the address of all the windows.
- **Why do we need to handle multiple windows in Selenium?**  
When a user is working on a web application, there might arise a scenario where a new window will open inside your main window.  
To handle such type of scenario we require window handle
- **What is a window handle in Selenium?**  
A window handle stores the unique address of the browser windows.  
Each window will have a unique ID that we can get using the methods provided by Selenium WebDriver
- **Different methods used for window handling in Selenium:**  
getWindowHandle(), getWindowHandles(), switchto(), action.

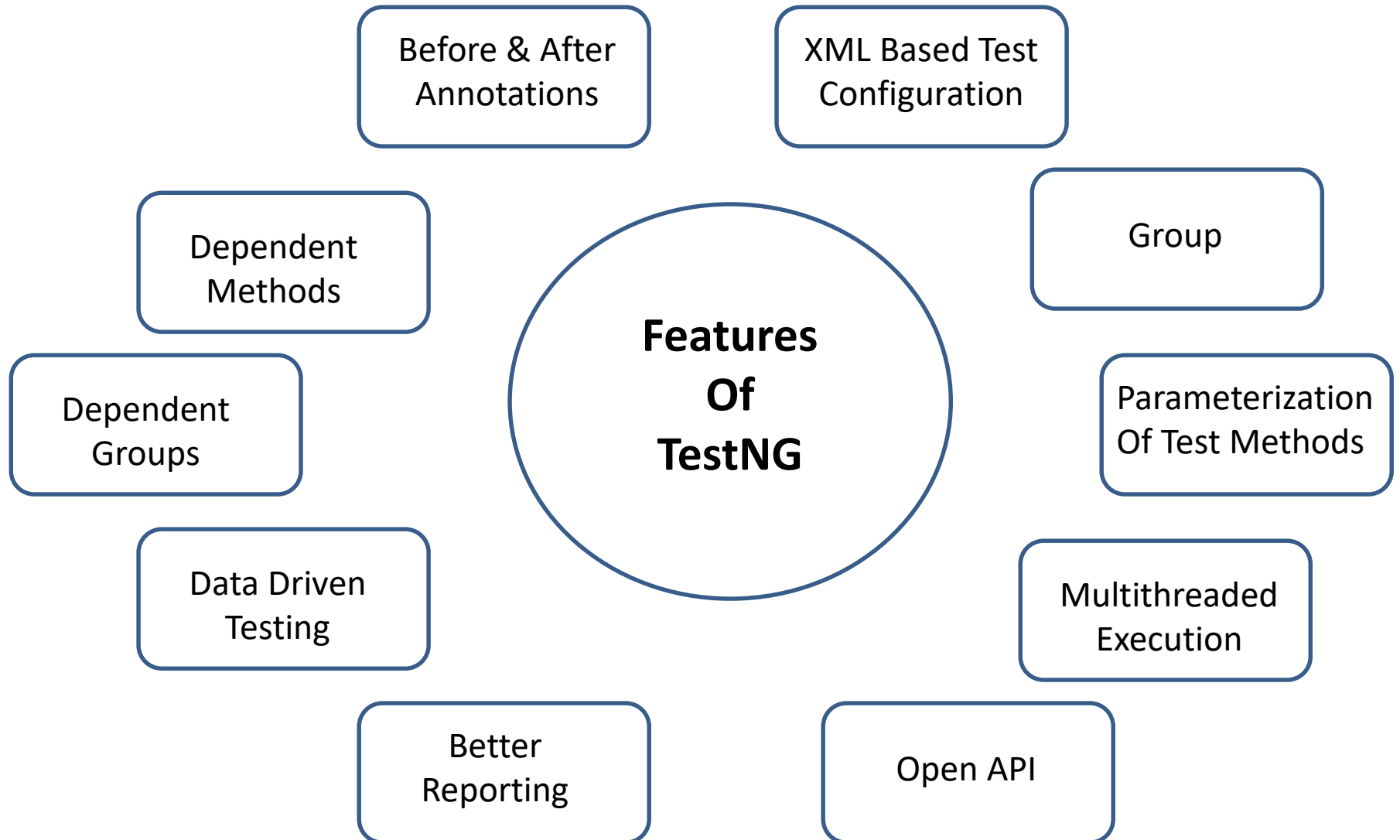
# KEYBOARD & MOUSE ACTIONS

- **Action class** in Selenium is a built-in feature provided by the Selenium for handling **keyboard and mouse events**.
- **Types of mouse actions can be performed:**
  - Click
  - Double Click
  - Click and Hold
  - Drag and Drop
- **Types of keyboard actions can be performed:**
  - Key Press
  - Key Release
  - Send Keys

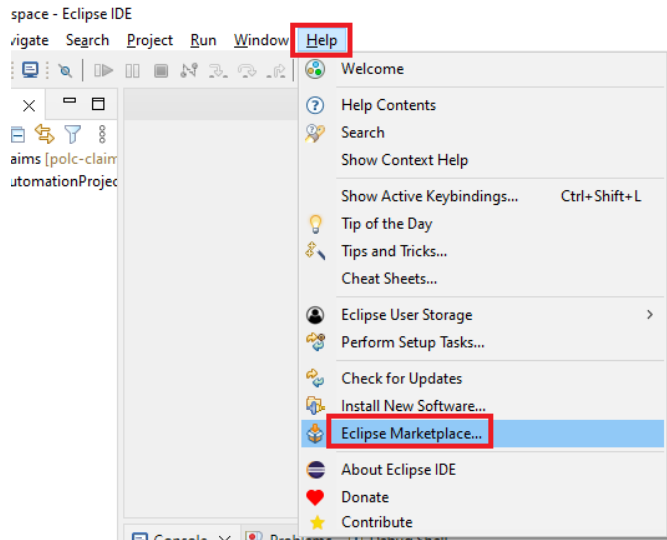
# TESTNG

- **TestNG** is an automation framework in which '**NG**' stands for '**Next Generation**'
- TestNG is an advance framework designed to take advantage of the benefits of both the developer and the testers.
- TestNG is an open source framework that is distributed under the Apache software license and is easily available for download.
- Due to its advance facilities, TestNG is considered to be better than JUnit.

# FEATURES OF TESTING

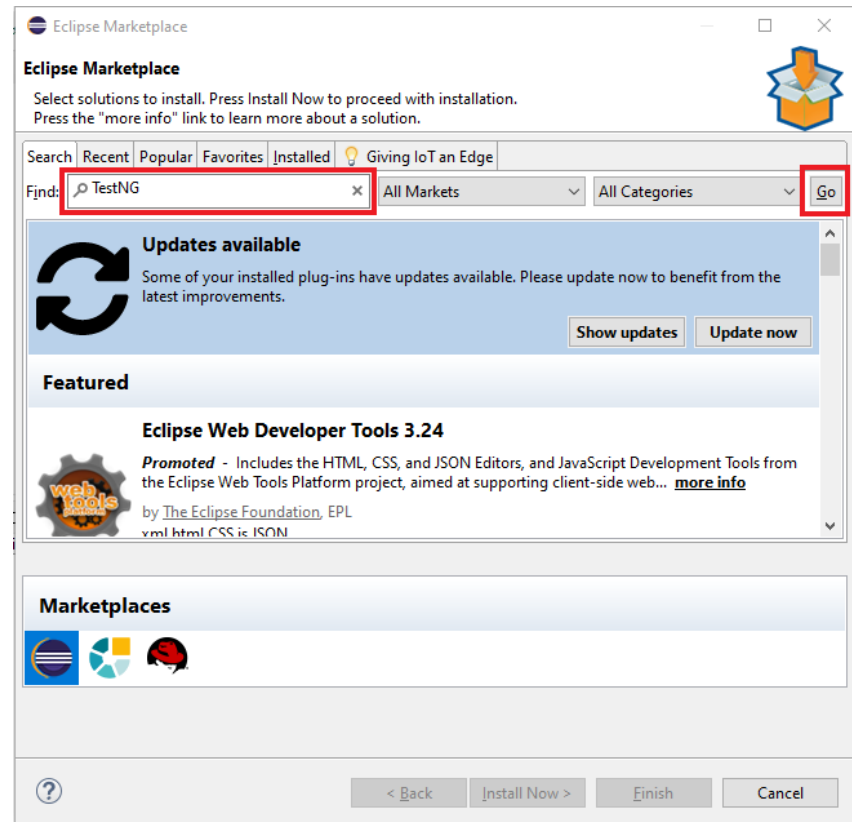


# Install TestNG in Eclipse

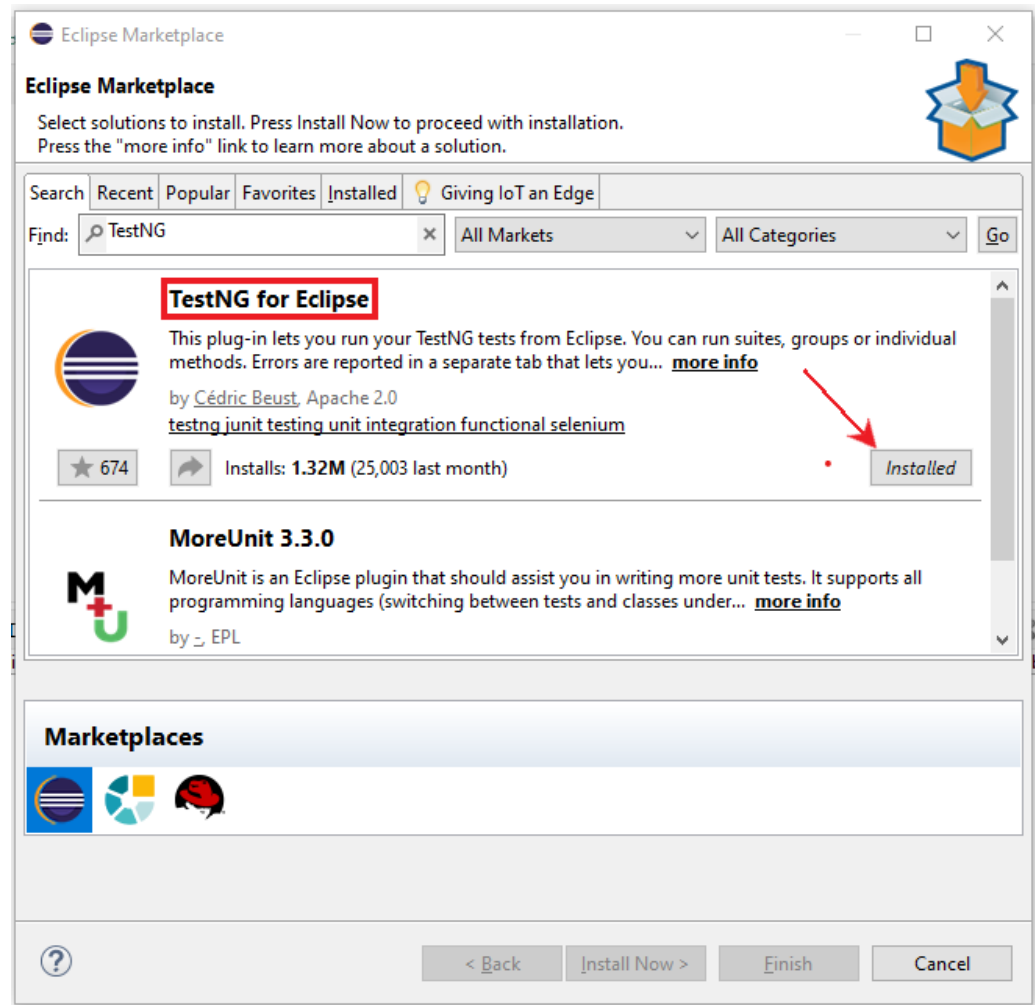


1. Click on **Help** -> **Eclipse Marketplace**

2. In **Find**, enter **TestNG** and click on **Go**



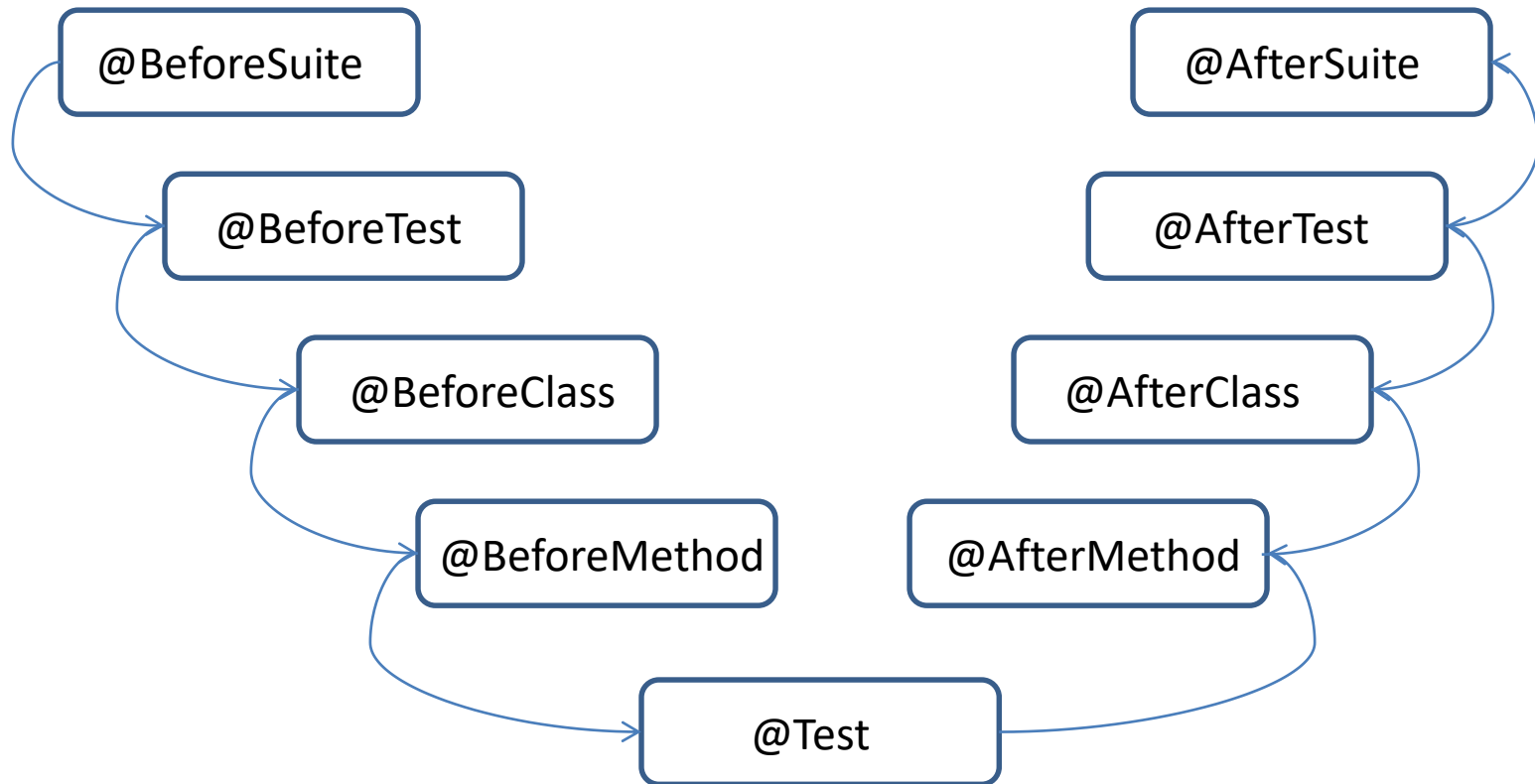
3. Look for **TestNG for Eclipse** and then click on **Install**
4. Then proceed with the installation process and after the installation click on **Restart Now**.
5. To confirm TestNG installed, click **Window -> Show View -> Other** and then check whether TestNG is there inside **Java folder** from a pop-up window.



# ANNOTATIONS

- @Test
- @BeforeSuite
- @AfterSuite
- @BeforeTest
- @AfterTest
- @BeforeClass
- @AfterClass
- @BeforeMethod
- @AfterMethod
- @BeforeGroups
- @AfterGroups
- @Parameters
- @DataProvider
- @Factory
- @Listeners

# EXECUTION SEQUENCE OF ANNOTATIONS





## *Annotation Attributes*

- description
- timeOut
- priority
- dependsOnMethods
- enabled
- groups