**SELENIUM REPORT**



**Table of Contents:**

1. Introduction
2. Install Java & Selenium -Get Started with basic Steps of the Selenium WebDriver.
3. Locator Techniques& Tools used to identify Objects.
4. Advanced Locators Identification & Interview Questions on Parsing Text.
5. Selenium Webdriver->Techniques to automate Web elements.
6. Deep Dive into Functional testing with Selenium.
7. Synchronization usage in Selenium WebDriver.
8. Techniques to automate Ajax calls, Child Windows and IFrames.
9. Real Time Exercises (end to end Programming).
10. Practical problems and Methods to Handle them with Selenium.
11. Miscellaneous topics in Selenium WebDriver.
12. Selenium Coding with Java Streams
13. Selenium Relative Locators
14. Invoking Multiple windows/tabs from selenium using one driver instance.
15. Capturing Web element using selenium-Screenshot.
16. Capturing height and width of Web element using selenium
17. TestNG Framework
18. Java Object Oriented Principles. --Inheritance
19. Framework -Part 1:Create Maven Project and Prepare functional end to end testing using TestNG framework.
20. Framework -Part 2: Design Pattern, Page object & factory implementation.
21. Framework -Part 3:Test Configuration methods & Global Properties & Parallel run.
22. Framework -Part 4:Test Startegy-Control Tests Execution-Run Parallel Tests.
23. Framework Part 5 -Extent HTML reports & TestNG Listeners & Thread Safe execution.
24. Framework Part 6 - Test Execution from Maven & Integration with Jenkins CI/CD.
25. Common Selenium Framework OOPS Interview Questions,
26. Framework Part 8 - Integrating Cucumber Wrapper into Selenium Framework.
27. GIT- Version Control
28. CI/CD Integration of Selenium Framework with Jenkins & GitHub.
29. Cross Browser Testing with Selenium Grid.
30. Understand Excel Data Driven Testing functions with examples.
31. Upload Download functionalities with Selenium using external excel files.
32. Selenium 4 Chrome Dev tools Protocol (CDP) Integration Concepts.
33. DataBase connection to Selenium Testcases.
34. File Uploading (AUTO IT) & Downloading with Selenium.
35. ExtentSpark Report and Extent Spark Report PDF For Cucumber + TestNG
36. Allure Report
37. Logger Implementation using Log4j2

**SELENIUM**

**Introduction:**

Selenium is an open-source automation testing tool. It is used exclusively for web based applications. You can work on multiple operating systems using selenium.

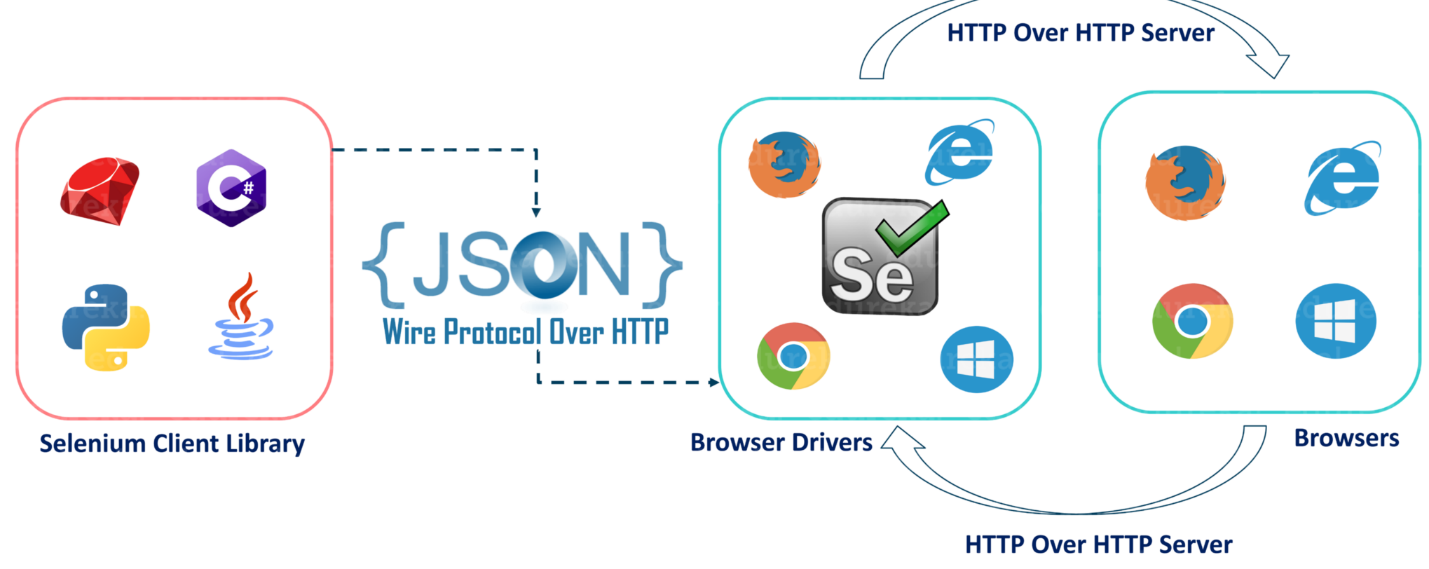
Out of the many open source test automation tools available today, there is one tool that become very popular in recent times Selenium. Selenium is a web application test automation tool which provides many features and a wide range of support which makes it an instant choice for automating most of the web applications.

Selenium provides the automation engineer the choice of running the automation scripts in a wide range of browsers and with the language support it has on wide range of operating systems too.

**Selenium Features:**

* Selenium is open-Source Automation Testing tool
* It is exclusively for Web Based applications.
* Selenium supports multiple browsers -   
  Chrome, Firefox, Internet Explorer, Safari
* Selenium works with Multiple Platforms   
  Windows, Apple OS ,Linux, Solaris
* Selenium can be coded in multiple languages -   
   Java, C#, Python, JavaScript, php, Ruby, Perl

**Selenium WebDriver Architectue Simplified:**



* After you trigger the Test, complete Selenium code (Client) which we have written will be converted to Json format
* Generated Json is sent to Browser Driver (Server) through http Protocol

Note: Each browser contains a separate browser driver

* . Browser drivers communicate with its respective browser and executes the commands by interpreting Json which It received on the browser.
* Browser Driver receives responses back from the browser and it sends Json response back to Client.

**Install Java & Selenium -Get Started with basic Steps of the Selenium WebDriver**

**5 Step by Step Instructions to run first basic Selenium Program**

1. Install Java and Set Java Home Path in System variables
2. Install Eclipse and Create new Maven Project with Selenium Dependencies
3. Understand creation of WebDriver object and its related classes
4. Run the First Selenium WebDriver Program with Browser Invocation
5. Different ways of setting Browser Driver executable files.

**Install Java and Set Java Home Path in System variables**

**Install Java & Eclipse-Setup Maven Project from Scratch**

**Install Java:**

Step1:

Search: Jdk Download

Download latest version of java for windows

Open the file and install

Step 2:

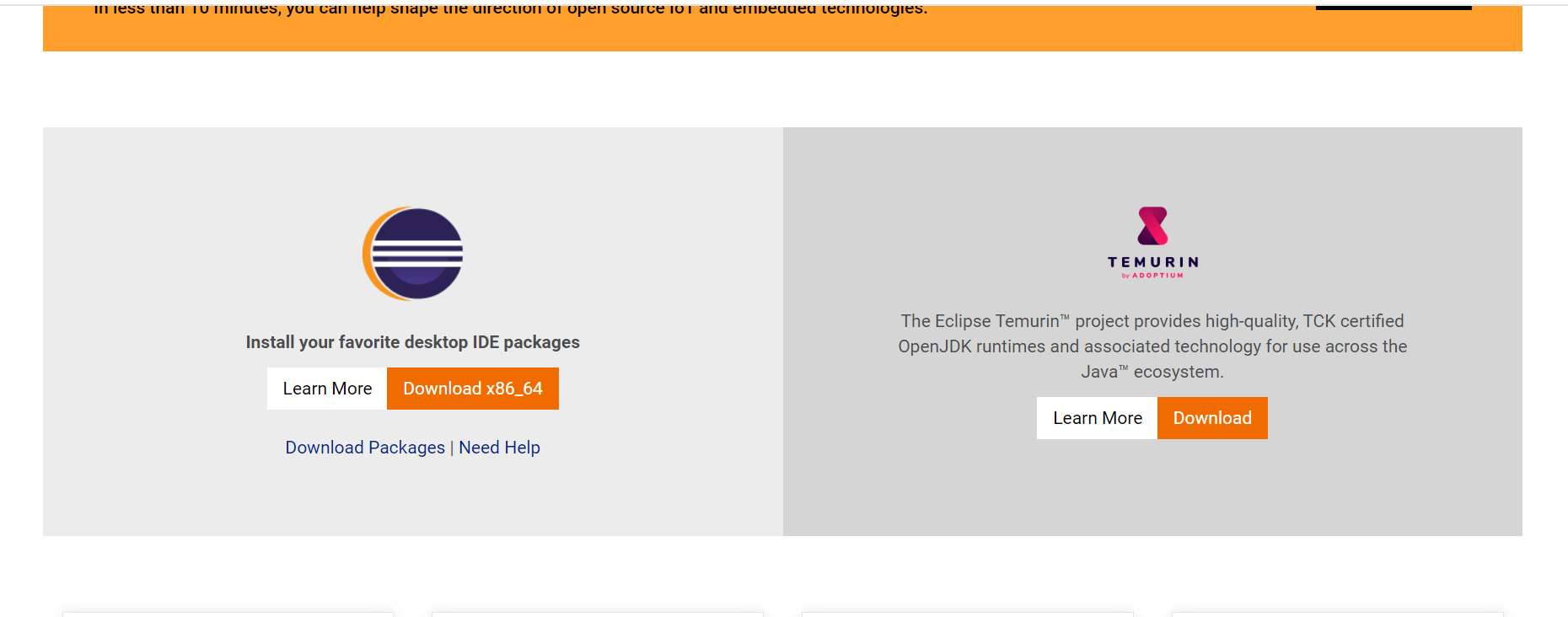
C drive > program files> java > latest version >copy the path > env>environment variables – new button > JAVA\_HOME – paste

C drive > program files> java > latest version> bin >env >environment Variables – path (select and click on edit)> new- paste

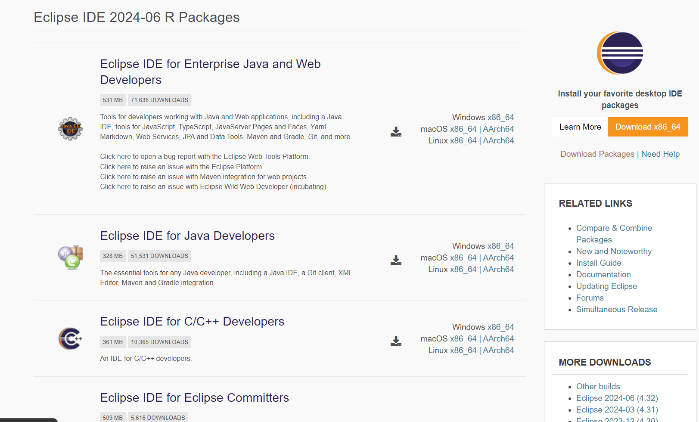
**Install Eclipse:**

Search: Eclipse download

Step 1: : <https://www.eclipse.org/downloads/>



Step 2: Click on Download Packages



These are the packages available.

Step 3: Click on “Eclipse IDE for Java Developers”

Download the IDE -The zip file get downloaded

Step 4: Open the Zip file and extract the file.

Step 6: Open the extracted folder click on application of eclipse (the application will launch)

Step 7: Specify the workspace

**Setting up Eclipse Maven Project with Selenium dependencies from Scratch**

Maven- Maven is a central repository for Java based projects.

**Selenium Dependencies/Jars**

Search for: maven repository

---selenium java dependency or/ Jar files

---TestNg – testing framework for java

Note: How to convert Java Project to Maven Project: Right Click on project > configure > convert to maven project.-Pom.xml file will be displayed.

In Maven project Group Id, Artifact Id, Packaging must specify.

**Understanding the core concept of Browser driver classes and Webdriver Interface**

**Invoking Browsers**

* Chrome - ChromeDriver Class
* FireFox - FireFoxDriver class
* Edge – EdgeDriver class
* Saafari – SaafariDriver class

Each class has there own methods.To access the methods we need to create the object of class.

Example 🡪 ChromeDriver driver = new ChromeDriver ();

driver object here has access to all the methods of Chrome driver

Note: \* ChromeDriver driver = new ChromeDriver();

// refers to both webdriver implemented method and chrome class methods but chrome methods are not applicable for firefox(do not support for all browsers)

In order to over come the above problem, there is Interface class WebDriver.

All the Driver like ChromeDriver , FireFoxDriver, EdgeDriver, SaafariDriver implements the WebDriver Interface.

WebDriver driver = new ChromeDriver (); //Webdriver methods + class methods

Note: <https://www.selenium.dev/selenium/docs/api/java/org/openqa/selenium/WebDriver.html>

1. **What is Interface in Java?**

An interface is a group of related methods with empty bodies.

Its class responsibility to implement the methods declared in the Interface

When class agreed to implement the interface, they must need to provide implementation/bodies to all the defined methods in Interface

In simple terms, Interface enforces the Contract to class to follow.

2. **WebDriver is an Interface which provides Set of Browser Automation methods with empty bodies (Abstract methods)**

Classes like ChromeDriver, FirefoxDriver, MicrosoftEdgeDriver , SafariDriver etc implement the WebDriver Interface

and provide their own implementation to the WebDriver methods

3. **We need to create the object of the class to access the methods present in the class.**

ChromeDriver driver = new ChromeDriver ();

driver object here has access to all the methods of Chrome driver

WebDriver driver = new ChromeDriver ();

driver object here has access to the methods of Chrome driver which are defined in web Driver Interface

**How to run tests in Google Chrome & Importance of Chromedriver.exe file.**

In case of browser restriction we will go for Chromedriver.exe file.

Chromedriver.exe file is a third-party file to run chrome browser

CHROME

chromeDriver.exe---one way by system.setProperty

Selenium Manager (library) ---2nd way by automatic [not in active state when chromeDriver.exe comes into picture]

Step to invoke chrome driver---chromedriver.exe acts like a third party file to invoke chrome browser

System.setProperty("webdriver.chrome.driver", "D:/Tools/chromedriver-win64/chromedriver.exe");

WebDriver driver = new ChromeDriver ();

**Download Chromedriver.exe file:**

1. Search Chrome driver download> <https://googlechromelabs.github.io/chrome-for-testing/> > Download the version which is compatible with the browser version.> download the zip file for windows.
2. Unzip the file to specific location, copy the path upto exe file is present > paste it on the code.

System.setProperty("webdriver.chrome.driver", "D:/Tools/chromedriver-win64/chromedriver.exe");

* In windows we need to provide .exe extension but not in mac.

Note: Troubleshooting steps if you fail to invoke chrome browser with Selenium

If you see any errors in invoking chrome browser as discussed in previous lecture, Then please perform below troubleshooting steps to make it work.

1. Update your Chrome browser to latest and Selenium version to latest and run again.

(  If it Still did not work,   - then as shown in Previous lecture, add System.SetProperty() step and get the exact chrome driver version from below link  which matches with your Chrome browser version and run again, It will work for sure. )

https://googlechromelabs.github.io/chrome-for-testing/

**Getting Started with Basic selenium webDriver methods.**

System.setProperty("webdriver.chrome.driver", "D:/Tools/chromedriver-win64/chromedriver.exe");

WebDriver driver = new ChromeDriver();

driver.get("https://integration.dotfoods.com/shop");

System.***out***.println(driver.getTitle());

System.***out***.println(driver.getCurrentUrl());

driver.close();

* get() – to launch on the website.
* getTitle() – to get title of the website.
* getCurrentUrl() – to get the current url after launching on website.
* close() - to close current window to which it is focusing
* quite() - close all the windows which are opened from selenium

**How to run tests in Firefox and Edge browser with Gecko and edge drivers**

**FireFox:**

System.setProperty("webdriver.gecko.driver", "D:/Tools/geckodriver.exe");

WebDriver driver = new FirefoxDriver();

Third Party file -geckodriver.exe

**Download geckodriver**

* Search > <https://github.com/mozilla/geckodriver/releases>
* Download zip file> unzip> copy the path upto geckodriver.exe file is present > paste it and give .exe extension in the code.

**Microsoft Edge:**

System.*setProperty*("webdriver.edge.driver", "D:/Tools/msedgedriver.exe");

WebDriver driver = **new** EdgeDriver();

Third Party file -msedgedriver.exe

**Download Edgedriver**

* Search >
* https://developer.microsoft.com/en-us/microsoft-edge/tools/webdriver?form=MA13LH
* Download zip file> unzip> copy the path upto msedgedriver.exe file is present > paste it and give .exe extension in the code.

**Locator Techniques& Tools used to identify Objects.**

**Importance of locators in Selenium WebDriver to identify the elements.**

* As part of Automation, Selenium Performs actions (such as click, typing) on the Page HTML Elements.
* The Locators are the way to identify an *HTML* element on a web page.   
  Selenium WebDriver uses any of the below locators to identify the element on the page and performs the Action.

ID

Xpath

CSS Selector

name

Class Name

Tag Name

Link Text

Partial Link Text

Example: <input type="text" placeholder="Username" id= “inputUsername” value=" ">

Input -> tag name

Red-> attribute   
Green-> attribute associated value.

**By ID** 🡪driver.findElement(By.*id*("inputUsername")).sendKeys("gagana");

**By Name** 🡪driver.findElement(By.*name*("inputPassword")).sendKeys("hello123");

**By Class Name** 🡪driver.findElement(By.*className*("submit")).click();

**Note: CSS And Xpath are generic selectors.**

1. **CSS Selector**

* **Class name -> tagname.classname**

Example: Button.signInBtn -> .error

* **Id -> tagname#id** ->

Example: input#inputUsername -> #inputUserName

* **Tagname[attribute=’value’]**

Example: <input type="text" placeholder="Username” value=" ">

Input [placeholder=’ Username’]

driver.findElement(By.*cssSelector*("input[placeholder='Email']")).sendKeys("gaganacm2001@gmail.com");

* **//Tagname[@attribute=’value’]:nth-child(index). - Child items**

driver.findElement(By.*cssSelector*("input[type='text']:nth-child(3)")).sendKeys("gaganacmgmail.com");

* **Parenttagname childtagname**
* **input[type\*='pass'] – CSS Regular expression**

Values of attributes keeps on changing every time in dynamic loading hence we can use regular expression.

* **tagname**

Note: Selector hub plugin download to chrome.

Css By console -> $(‘ Button.signInBtn’)

1. **LinkText**

Should have <a> tag

driver.findElement(By.*linkText*("Forgot your password?")).click();

1. **Xpath**

* **//Tagname[@attribute=’value’]**

**Example:** //input[@placeholder=’ Username’’]

<input type="text" placeholder="Name">

//input[@placeholder=’ Name’]

driver.findElement(By.*xpath*("//input[@placeholder='Name']")).sendKeys("Gagana");

* **//Tagname[@attribute=’value’][index]**

driver.findElement(By.*xpath*("//input[@type='text'][2]")).clear();

* **//parentTagname/childTagname**

**//parentTagname/childtagname[index]**

Example: //form/input

//form/input[3]

* **//button[contains(@class,'submit')]. – Regular expression**
* **//tagname**
* **//tagname[text() =’logout’] -applicable only for xpath**

**//\*[text() =’logout’]**

* **//header/div/button[1]/following-sibling::button[1]**
* **//header/div/button[1]/parent::div**

**//header/div/button[1]/parent::div/parent::header**

Note: console : $x(‘ ‘)

* Absolute xpath -> /html/header -🡪 when starts from root.
* Relative xpath -> //header -> starts in between the code.

**Automate Browser navigations and window properties with Selenium Webdriver.**

**public** **class** WindowsActivities {

**public** **static** **void** main(String[] args) {

System.*setProperty*("webdriver.chrome.driver", "D:/Tools/chromedriver-win64/chromedriver.exe");

WebDriver driver = **new** ChromeDriver();

driver.manage().window().maximize();

driver.get("https://google.com");

driver.navigate().to("https://rahulshettyacademy.com");

driver.navigate().back();

driver.navigate().forward();

}

}

**Selenium Webdriver->Techniques to automate Web elements**

**Handling static dropdowns with Select webDriver API**

<https://rahulshettyacademy.com/dropdownsPractise/>

If any tag has <select> tag then there is special class to handle static drop downs i.e, Select class

Select dropdown = **new** Select(staticDropdown);



**Handling Dynamic dropdowns with WebDriver API**

****

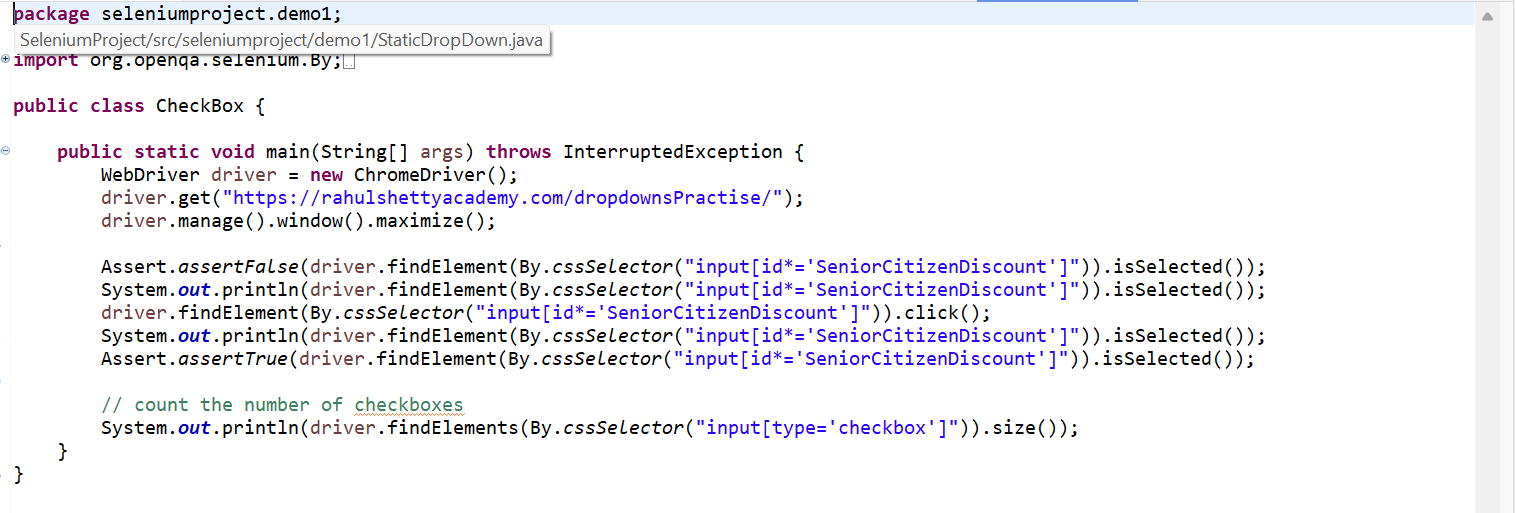
Parent child relationship locators to identify the objects uniquely.

Example: //div[@id='glsctl00\_mainContent\_ddl\_originStation1\_CTNR'] //a[@value='BLR']

**Handling AutoSuggestive dropdowns using selenium.**



**Handling checkbox and getting the size of them with selenium.**



**Importance of Assertions in automation testing and how to use them.**

TestNg is a testing framework

Assertions is one of the method to validate the results . This assertions need testNg depencency.

<dependency>

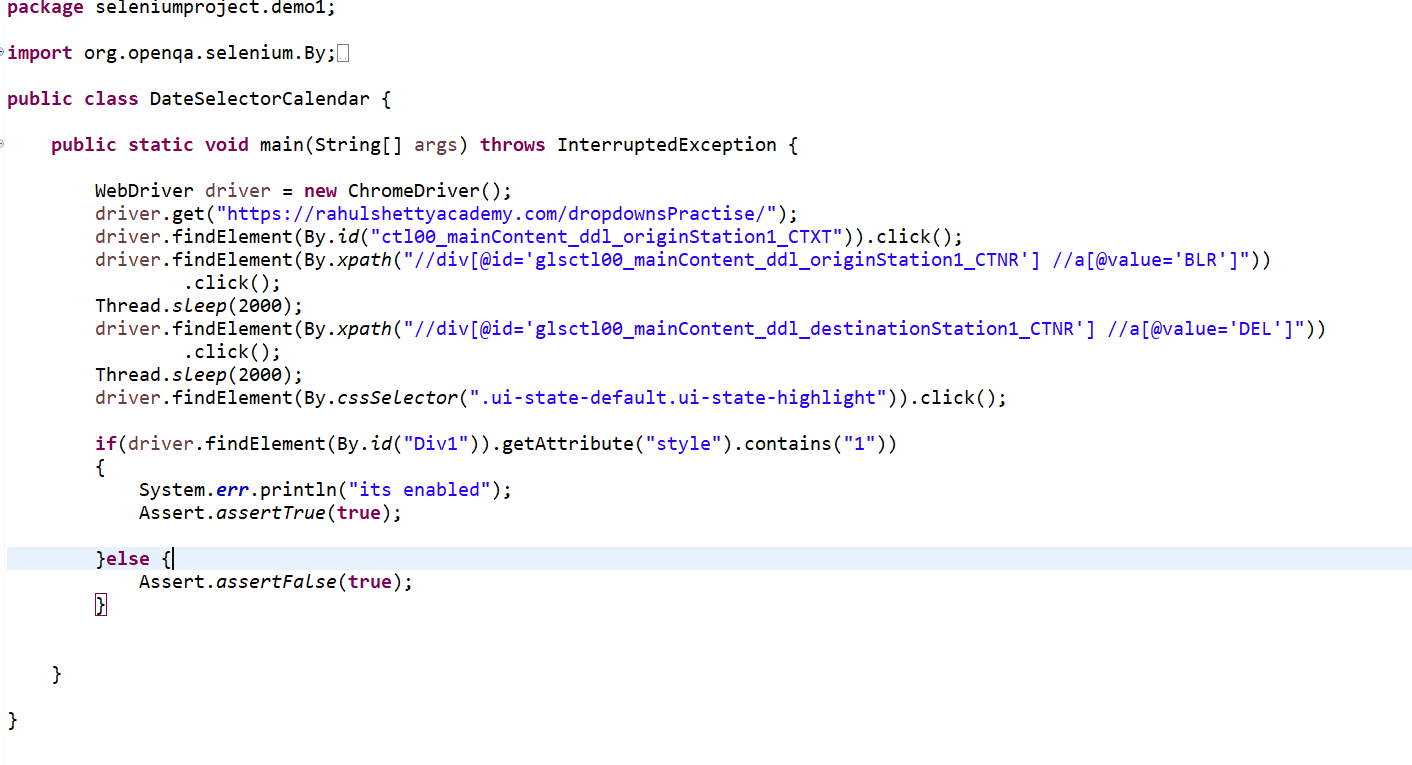
<groupId>org.testng</groupId>

<artifactId>testng</artifactId>

<version>7.10.2</version>

</dependency>

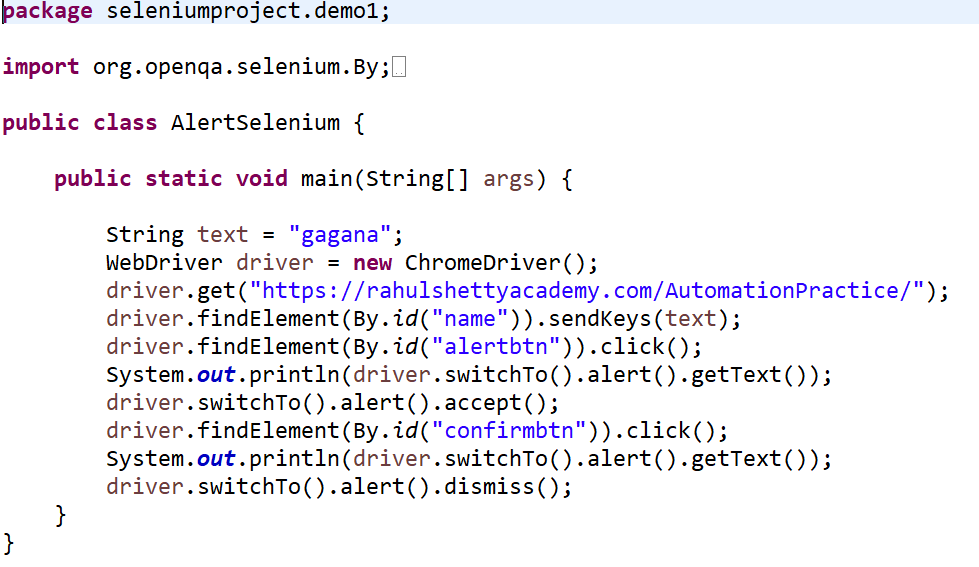
**Handling Calendar UI in travel website using selenium.**

****

**Handling java Alerts using selenium webDriver.**

driver.switchTo().alert()

The above line switch from context to alert.

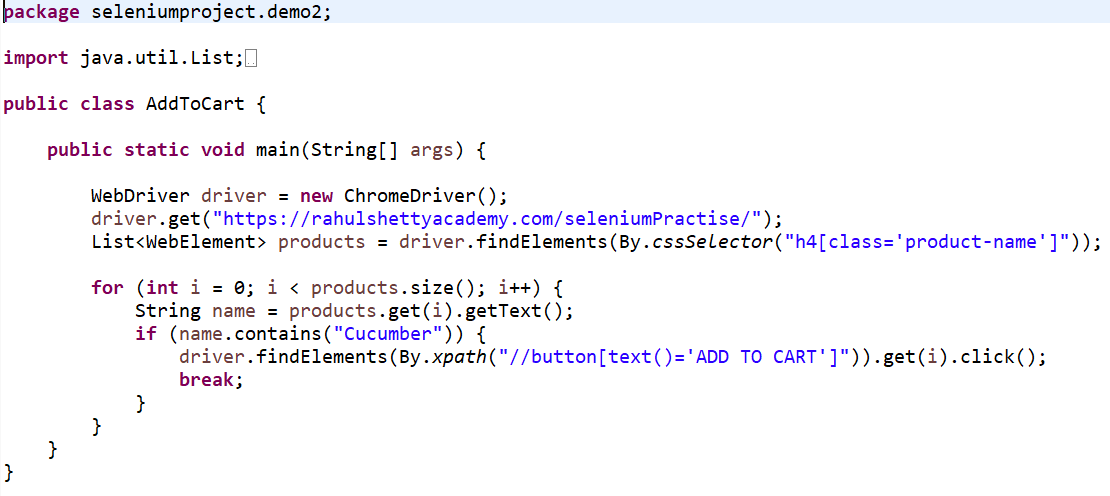
****

Note: driver.switchTo().alert().accept(); // accepts the alert (positive scope)

driver.switchTo().alert().dismiss(); // cancel the alert (negative scope)

**Deep Dive into Functional testing with Selenium**

Problems -Adding Items into cart for Ecommerce app.



Sending array of products to cart for checkout

A computer screen shot of a program

Description automatically generated

Building programming logic to process items in array for cart

A screenshot of a computer program

Description automatically generated

**Synchronization usage in Selenium webdriver**

**Implicit wait**

Define Globally

Hey wait for n number of seconds before you throw exception.

This is declared globally it would be applicable for each lines/steps.

driver.manage().timeouts().implicitlyWait(Duration.*ofSeconds*(5));

Example: Entered form and select movie/flight enter

Search:

Load 3 seconds

First Flight result

Advantage: Readable code

Disadvantage: Loading 1000 serach results from db takes 15 seconds then the implicit wait does not works.

* Performance cause issues are not aught
* If page loads within 5seconds it will lead to performance.
* If time exceeds for any one step then it will fail

**Explicit wait**

To target specific element.

WebDriverWait w = **new** WebDriverWait(driver, Duration.*ofSeconds*(5));

w.until(ExpectedConditions.*visibilityOf*(element));

w.until(ExpectedConditions.*invisibilityOf*(element));

w.until(ExpectedConditions.*elementToBeClickable*(element));

w.until(ExpectedConditions.*visibilityOfAllElementsLocatedBy*(findBy));

WebDriverWait -> is a class to declare explicit wait.

Arguments -> driver object

How many seconds to wait.

Advantages:

* Applied whenever it is required.
* Wait is applied at targeted elements, so no performance issue.

Disadvantage:

* No. of codes is increased/more code.

**Thread.sleep()**

Part of java to halt/ stop the execution of script for n seconds.

**Fluent wait**

The another explicit wait mechanism type.

2ways

1. WebDriverWait
2. FluentWait

Fluent wait finds the webElement repeatedly at regular interval of time until the timeout or till the object gets found.

Example : WebDriverWait – 10 seconds

FluentWait – 10 seconds, pooling 2 seconds



Disadvantage:

Messy code

Note: FluentWait and WebDriverWait are class implements Wait interface.

Unlinke WebDriverWait , we need to build customized wait methods based on conditions in case of fluentwait.

**Techniques to automate Ajax calls, Child Windows and IFrames**

**Handling Ajax/Mouse interaction/Actions class**

* How to MouseOver on object with selenium
* Performing Mouse and keyboard interactions with Selenim
* Context click on element.
* Double click on element
* Drag and dropping the element.

The above things can be achieved through the Actions class

Actions a = **new** Actions(driver);



**MouseOver-Move to specific element:** a.moveToElement(move).build().perform();

**Enter the capital letters:** a.moveToElement(entry).click().keyDown(Keys.***SHIFT***).sendKeys("hello").doubleClick().build().perform();

**Double Click**: doubleClick()

**Right Click**: contextClick()

**Window handle concepts -real time example**

Interaction between child window and parent window.

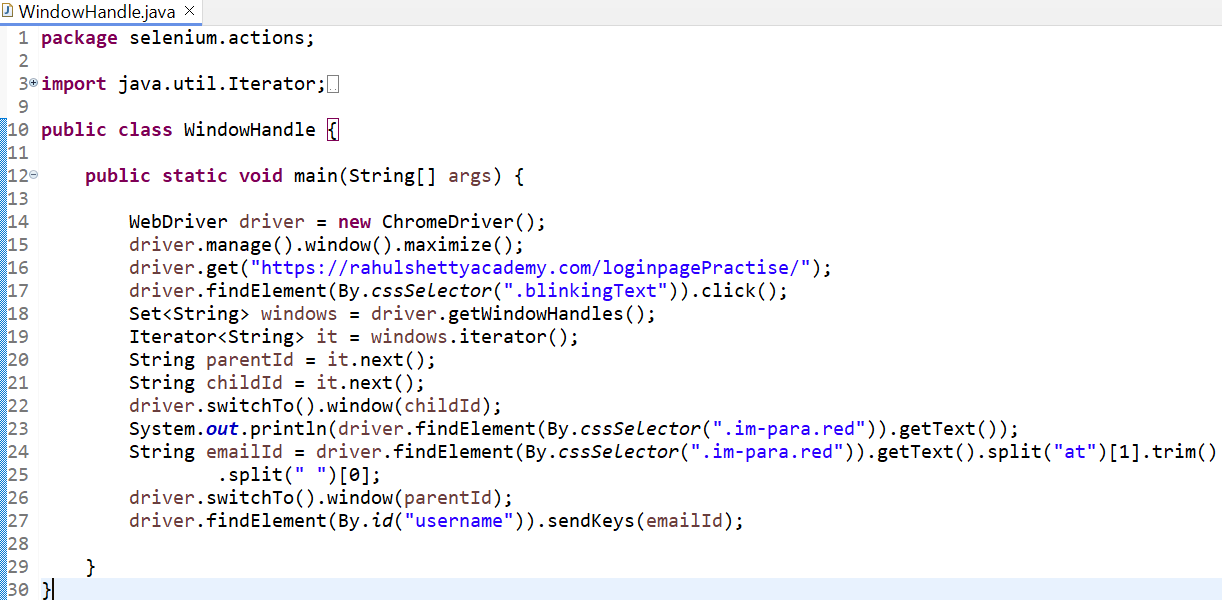
Set<String> windows = driver.getWindowHandles();

Iterator<String> it = windows.iterator();

String parentId = it.next();

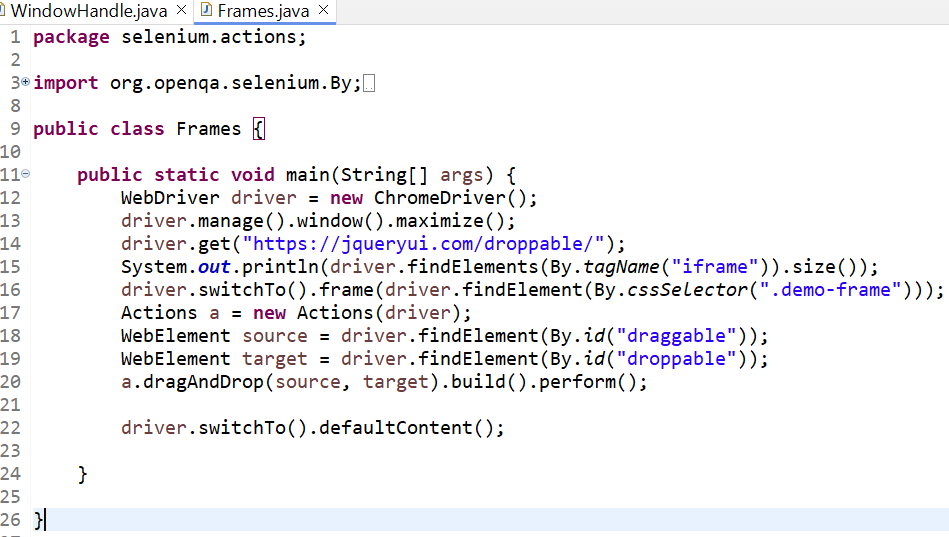
String childId = it.next();

driver.switchTo().window(childId);



**How to Handle Frames/Frames Technique-real time example(drag and drop)**

Frames are hosted in container, In the context of a web browser, a frame is a part of browser window which displays context independent of its container with the ability to lead content idependently.



driver.switchTo().frame(driver.findElement(By.*cssSelector*(".demo-frame")));

**drag and drop**

Actions a = **new** Actions(driver);

WebElement source = driver.findElement(By.*id*("draggable"));

WebElement target = driver.findElement(By.*id*("droppable"));

a.dragAndDrop(source, target).build().perform();

**Through index:**

driver.switchTo().frame(0);

Once completed with working on frames give the below step to switch to context:

driver.switchTo().defaultContent();

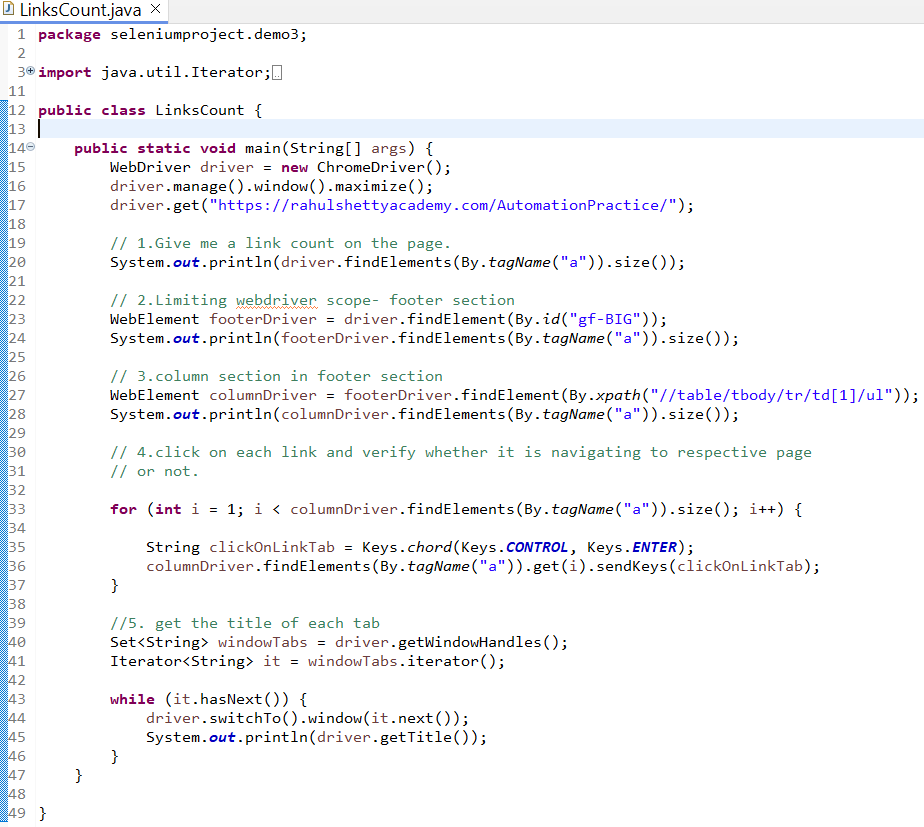
**Real Time Exercises (end to end Programming)**

**-Practice Exercise -print the links count in the page.**

**-Limiting WebDriver scope**

**-How to open the links in separate tabs**

**-Getting the title of child tabs with optimized while loop**

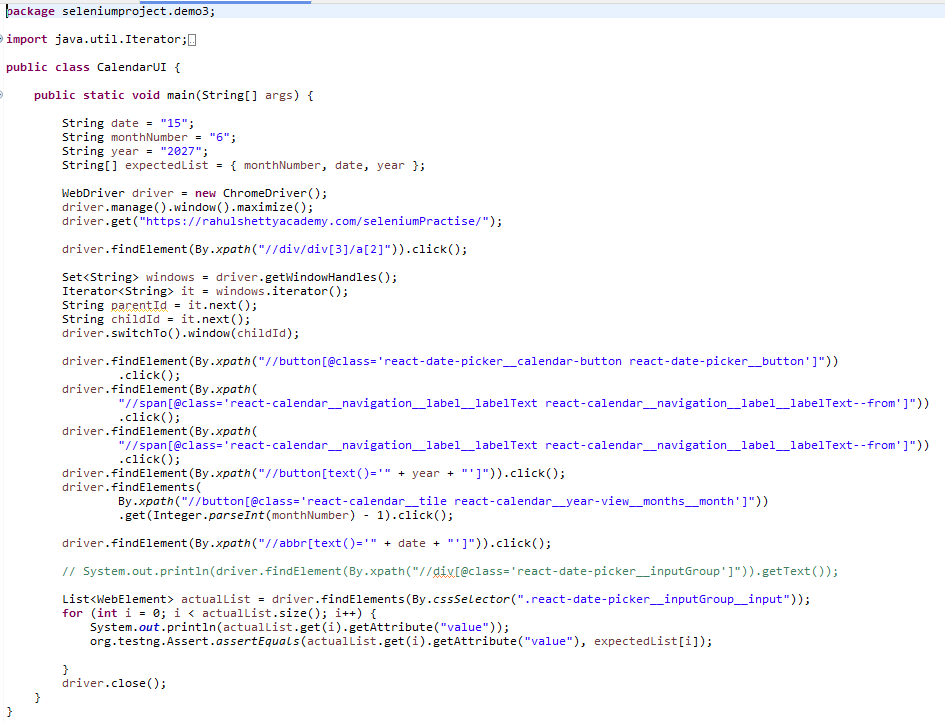
****

**Note**: hasNext() -> tells whether next index is presentor not

next() -> move to next index.

**- Handling calendar UI in Ecommerce/Travel websites.**

**-Generic methods to handle calendar with month and date.**

****

**Practical problems and Methods to Handle them with Selenium**

**-How to Perform Scrolling with in table and Window level using JavaScriptExecutor**

**-How to handle table grids in webpage.**

**-Parsing String and comparing with Generated sum value**



A computer screen shot of a code

Description automatically generated

JavascriptExecutor js = (JavascriptExecutor) driver;

js.executeScript("window.scrollBy(0,800)");

JavascriptExecutor is class to execute java scripts , we need to cast the driver to JavascriptExecutor.

Note: JS

Dom object -> document

js.executeScript("document.querySelector('.tableFixHead').scrollTop=5000");

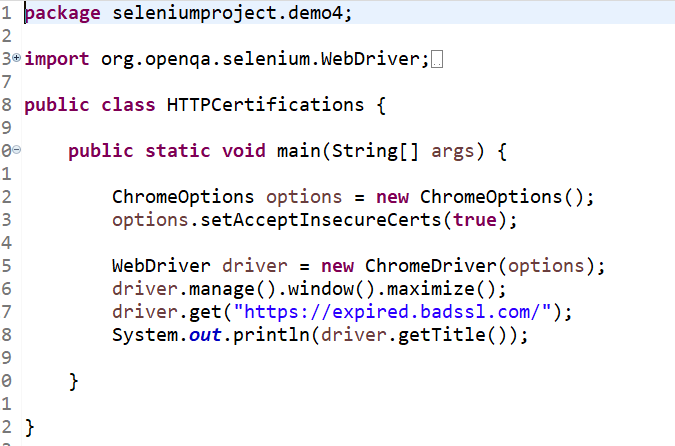
.scrollTop

.scrollLeft

**Miscellaneous topics in Selenium WebDriver**

**-Handling HTTP certifications in automated browsers**

**-Explore Chrome options to set proxies, plugins and paths on chrome browsers**

****

**Not accessible websites.**

ChromeOptions options = **new** ChromeOptions();

options.setAcceptInsecureCerts(**true**);

WebDriver driver = **new** ChromeDriver(options);

**FireFoxOptions**

**EdgeOptions**

**Note: When website needs proxy to access**

ChromeOptions options = new Chromeoptions();

Proxy proxy = new Proxy();

Proxy.setHttpProxy(“ipadress:44444”);

Options.setCapability(“proxy”,proxy);

Options.setAcceptInsecureCerts(true);

WebDriver driver = new ChromeDriver(options);

**-Maximizing window and deleting cookies.**

****

**-How to take screenshots in selenium**

**A screenshot of a computer program

Description automatically generated**

FileUtils -> method to copy the file from the src object to local machine.

Need jar file -> commons.io

**-Strategy to automate the broken links with selenium**

**-Open connection method to identify status codes of the links**

**-Iterate over all links in the page to validate broken links mechanism**

**-Importance of soft assertions in selenium webdriver.**

HttpURLConnection conn = (HttpURLConnection) **new** URL(url).openConnection();

HttpURLConnection-> class

openConnection() -> method

A screenshot of a computer program

Description automatically generated

A screenshot of a computer code

Description automatically generated

**Selenium Coding with Java Streams**

Java streams was introduced in java 8 which comes with many helper method to write simplified and optimized code.

**What are Streams?**

Stream API is a new feature available from Java8

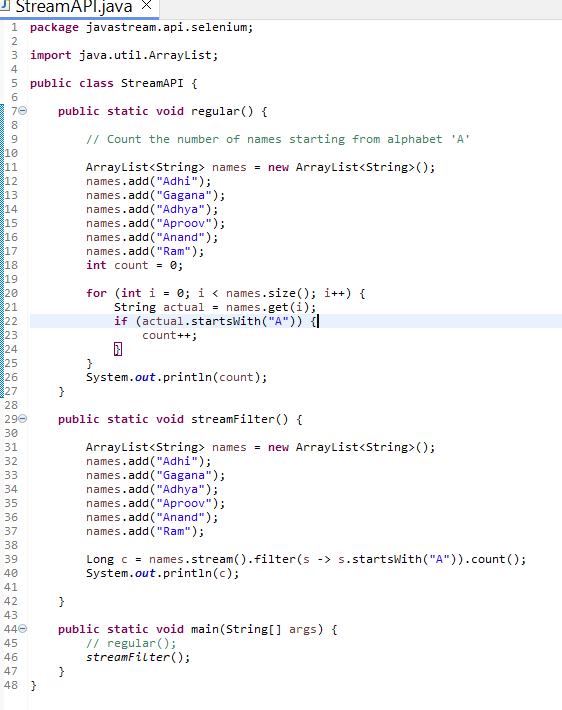
By using this, we can perform aggregate operations on the data returned from collections classes by drastically reducing the complexity of the code.

**Program:**To store the elements in the list and to identify the elements stored in list which starts from alphabet ‘A’(count)

**What is Lambda Expression?**

Lambda Expression introduces new arrow operator -> into java. It divides into two parts.

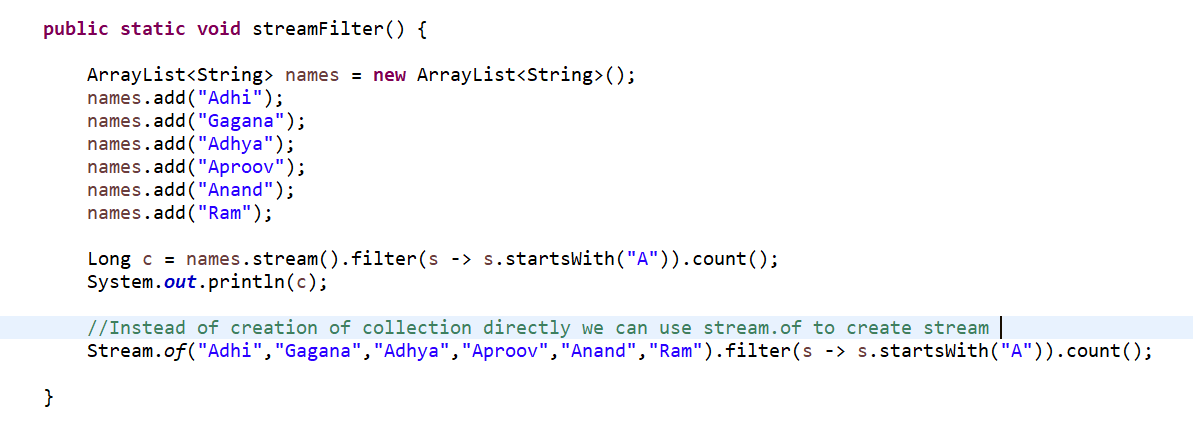
First part is left side specifies the parameters required by the expression, which could also be empty when no parameter is required.

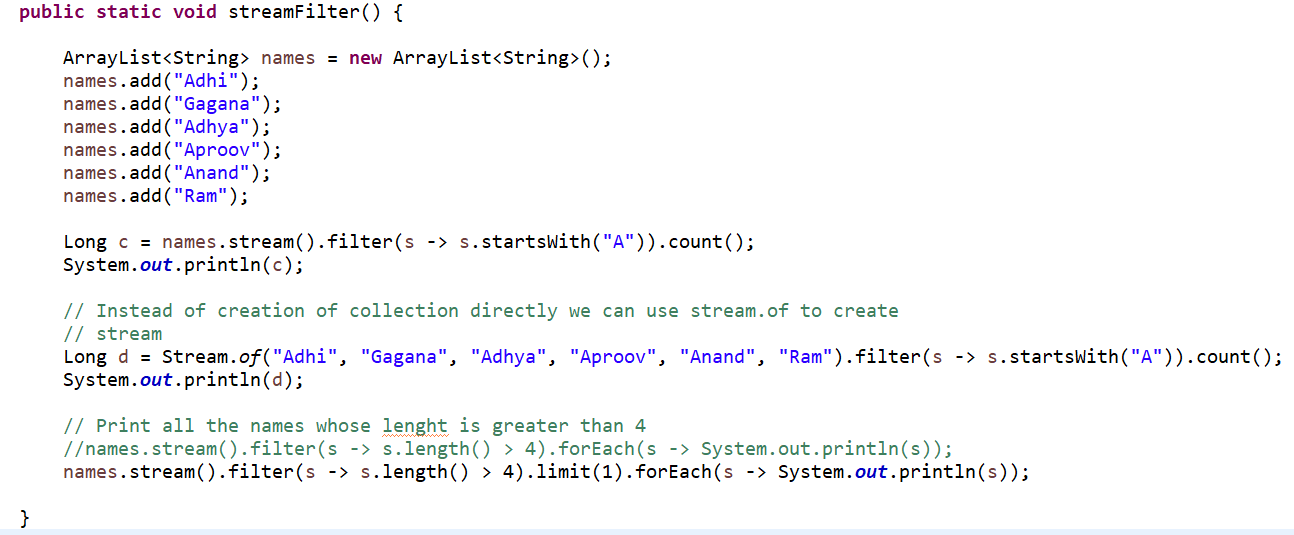


Note: when we perform aggregate operation on collection, array or any other data source do not change data of the source, they simply returned the new stream.

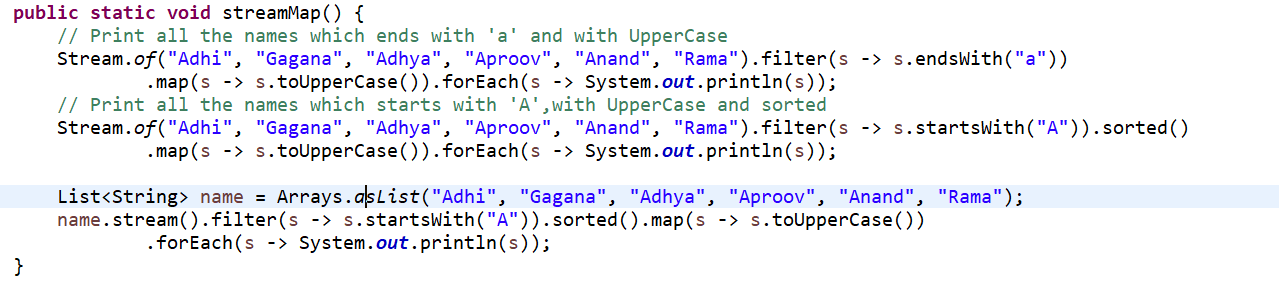
There is no life for intermediate option when there is no terminal operation.

Terminal operation executes only when intermediate operation returns true.

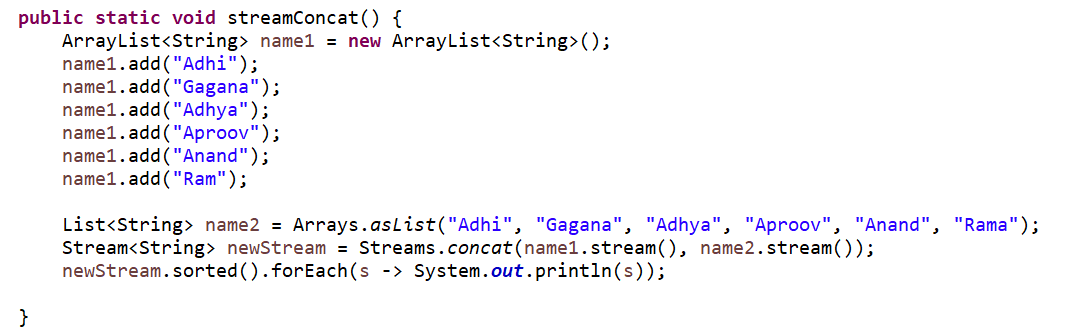


****

**StreamMap:**

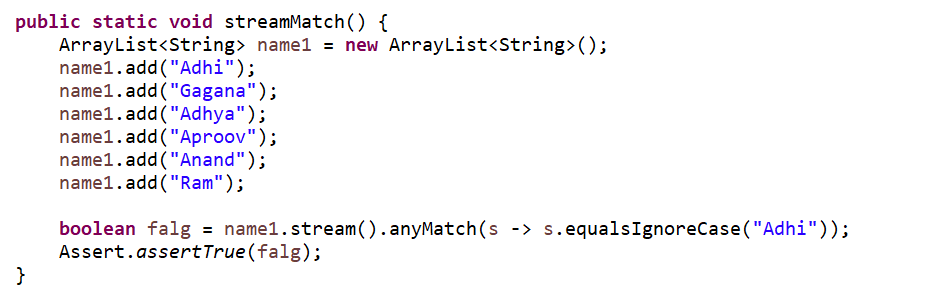
****

**Stream Concat:**

****

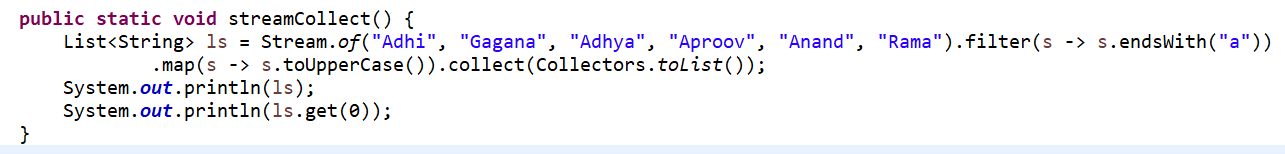
**Stream-match:**

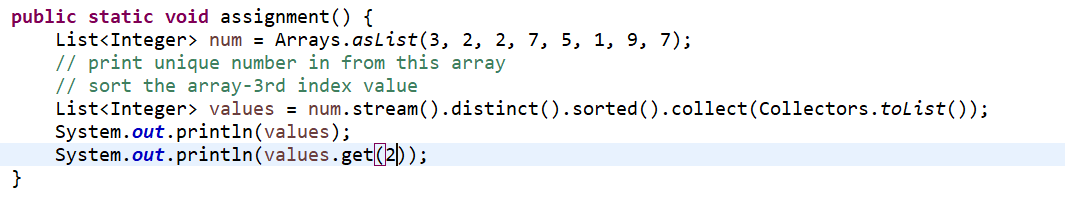
match method return value is Boolean.



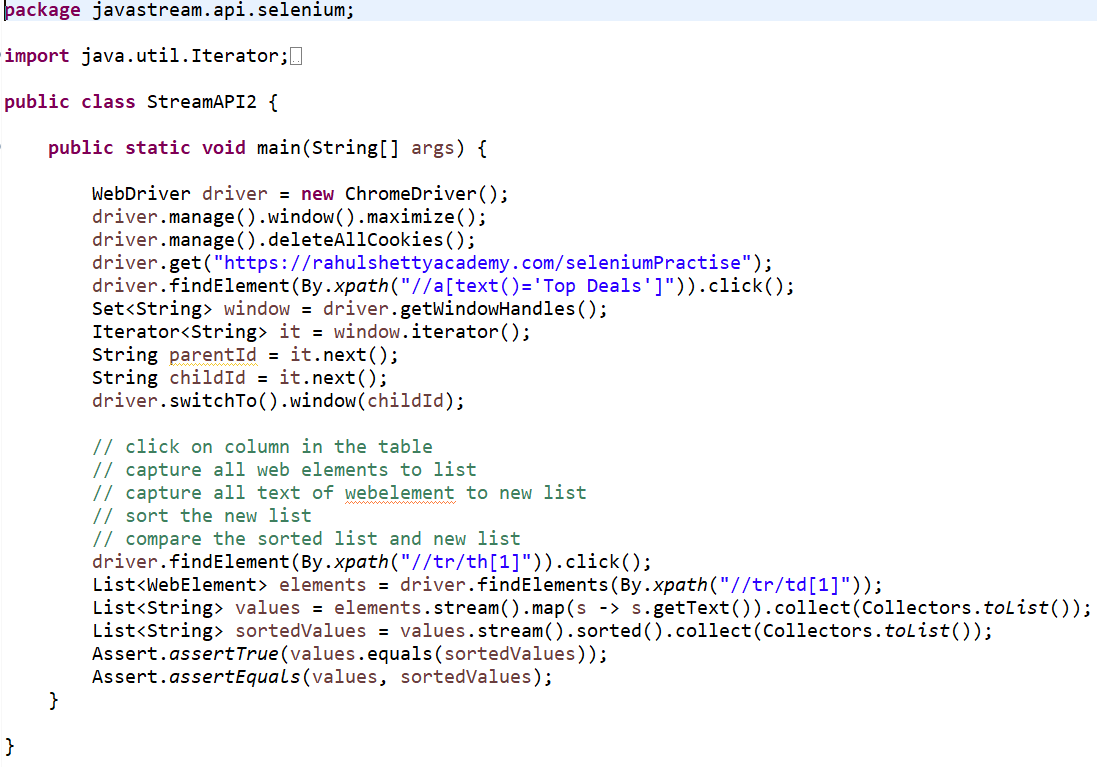
**Stream-collect:**

collect method- takes list covert it into stream perform operations and finaly convert back into list.





**Web Table Sorting using Selenium Java Stream.**

****

**-Custom selenium methods using mapper method**

****

**- Pagination**

**A computer screen shot of a program

Description automatically generated**

**-Filter the WebTable**

**A screen shot of a computer code

Description automatically generated**

**Selenium Relative Locators**

above()= Element located above with respect to specified element.

below ()= Element located below with respect to specified element.

toLeftOf() = Element located to the left of specified element.

toRightOf()=Element located to the right of specified element.

**Syntax:**

**Driver.findElement(with(By.TagName( “XX”)).above(WebElement));**

**Note:static import: import** **static** org.openqa.selenium.support.locators.RelativeLocator.\*;

Relative locators do not support flex elements.

**Invoking Multiple windows/tabs from selenium using one driver instance.**

driver.switchTo().newWindow(WindowType.TAB);

driver.switchTo().newWindow(WindowType.***WINDOW***);

**Capturing Webelement using selenium-Screenshot**

WebElement name = driver.findElement(By.*name*("name"));

File file = name.getScreenshotAs(OutputType.***FILE***);

FileUtils.*copyFile*(file, **new** File("name.png"));

**Capturing height and width of Webelement using selenium**

WebElement name = driver.findElement(By.*name*("name"));

System.***out***.println(name.getRect().getDimension().getHeight());

System.***out***.println(name.getRect().getDimension().getWidth());

**TestNG Framework**

TestNG is a testing framework inspired by Junit and Nunit.

**Installation and configuration in Eclipse:**

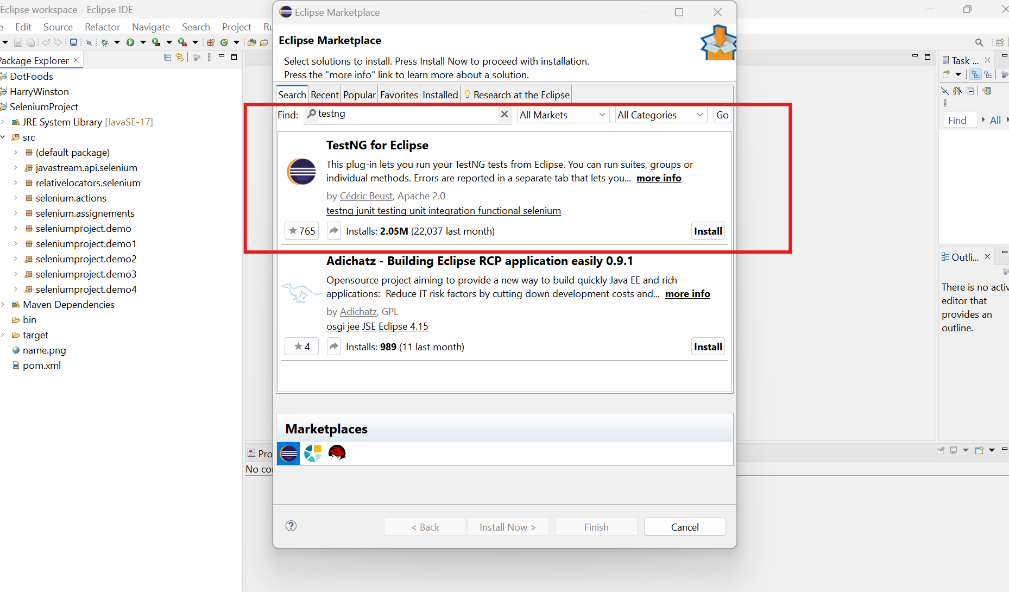
1.Click on help option:

A screenshot of a computer

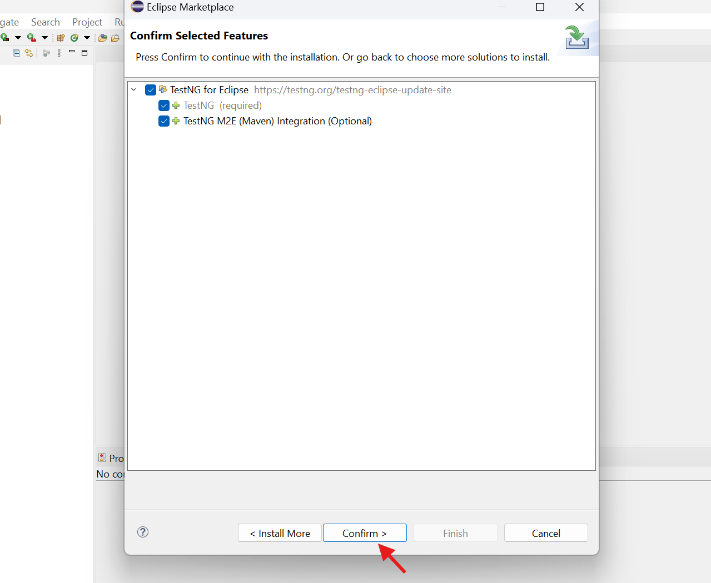
Description automatically generated

2.Select Eclipse marketPlace

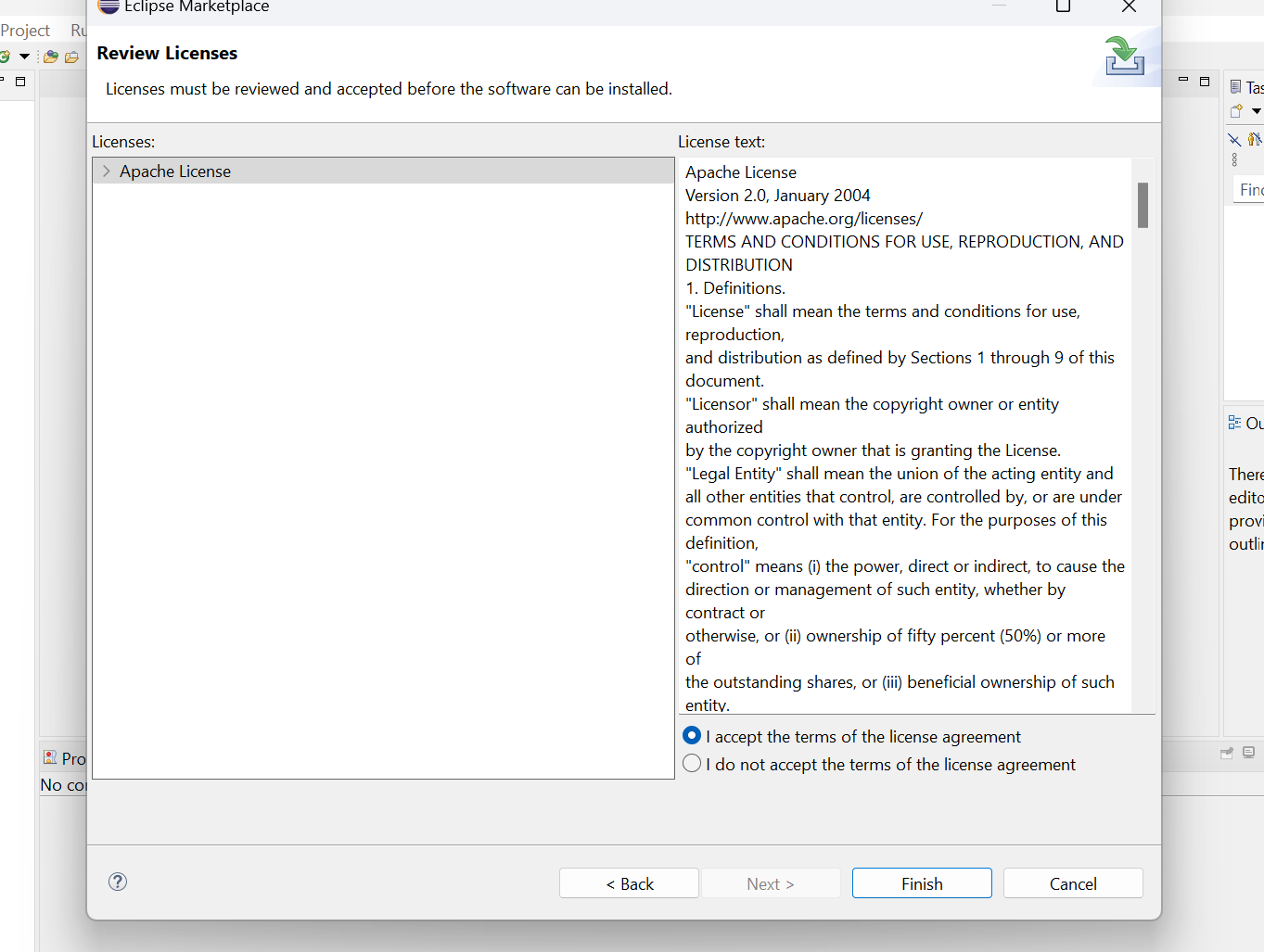
3.Search for TestNg and install first result



4.Click on confirm.



5.Accept the license and click on finish and restart the eclipse IDE



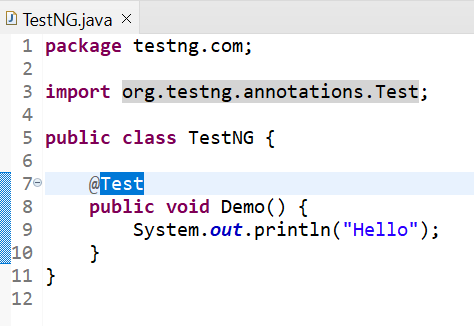
Note: Confirmation: Go to window option>Preferences>search bar: testNg.

**Running Testcases in TestNg without Java Compiler:**

* TestNg itself acts a java compiler(no need of java compiler-main() method).
* TestNg need all execution inside a method.
* By @Test annotation from TestNg library-which will recognize a method.

Note:How to run test in TestNg?

You need to have @Test annotation followed by method.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Note:You can define multiple tests from single class.

**Importance of xml file in TestNg Configuration:**

Hierarchy: Test Suite->Test Folder (Module)->test cases

<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd">

<suite name="Suite">

<test thread-count="5" name="Test">

<classes>

<class name="testng.com.TestNG" />

<class name="testng.com.TestNG2" />

</classes>

</test>

</suite>

**Prioritizing the test cases with TestNg:**

<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd">

<suite name="Loan Department">

<test name="Personal Loan">

<classes>

<class name="testng.com.TestNG" />

<class name="testng.com.TestNG2" />

<class name="testng.com.HomeLoan"/>

</classes>

</test>

<test name="Car Loan">

<classes>

<class name="testng.com.CarLoan" />

</classes>

</test>

</suite>

Note: You can modularize the test cases based on functionality and trigger them accordingly.

**Include and Exclude Mechanism to control test cases:**

<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd">

<suite name="Loan Department">

<test name="Personal Loan">

<classes>

<class name="testng.com.TestNG">

<methods>

<include name="Demo" />

</methods>

</class>

<class name="testng.com.TestNG2" />

<class name="testng.com.HomeLoan" />

</classes>

</test>

<test name="Car Loan">

<classes>

<class name="testng.com.CarLoan">

<methods>

<exclude name='MobileLoginCarLoan' />

</methods>

</class>

</classes>

</test>

</suite>

Note:You can also get control to run specific methods from test cases.

**Executing test cases at package level with regex:**

<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd">

<suite name="Loan Department">

<test name="Personal Loan">

<classes>

<class name="testng.com.TestNG">

<methods>

<include name="Demo" />

</methods>

</class>

<class name="testng.com.TestNG2" />

<class name="testng.com.HomeLoan" />

</classes>

</test>

<test name="Car Loan">

<classes>

<class name="testng.com.CarLoan">

<methods>

<exclude name='Mobile.\*' />

</methods>

</class>

</classes>

</test>

</suite>

**Package level:**

<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd">

<suite name="Loan Department">

<test name="Personal Loan">

<packages>

<package name="testng.com" />

</packages>

</test>

</suite>

**TestNg Annotations:**

* **@BeforeTest**-executes before all the test cases. (example: to clear the data in data base before test execution).

**Note**: scope is only within the test <test> </test>

* **@AfterTest**- executes after all the test cases.

**Note**: scope is only within the test <test> </test>

* **@BeforeSuite-**executes before the suite i.e, before the test execution of tests within the suite.
* **@AfterSuite-** executes after the suite i.e, after the test execution of tests within the suite.
* **@BeforeMethod-**executes before each tests within the scope of class.
* **@AfterMethod-**executes after each tests within the scope of class.
* **@BeforeClass-**Before executing any method in the class.
* **@AfterClass-**After executing any method in the class.

**Usage of Groups functionality in TestNg**

* @Test(groups="Smoke")

**XML configuration:**

<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd">

<suite name="Loan Department">

<test name="Regression">

<groups>

<run>

<include name="Smoke" />

</run>

</groups>

<classes>

<class name="testng.com.TestNG" />

<class name="testng.com.TestNG2" />

<class name="testng.com.HomeLoan" />

<class name="testng.com.CarLoan" />

</classes>

</test>

</suite>

**Annotations Helper attributes with examples**

* @Test(dependsOnMethods = { "webLoginCarLoan" })

-After execution of weLoginCarLoan only the test is executed.

* @Test(enabled=**false**)-skips that particular method
* @Test(enabled=**True**)-includs that particular method
* @Test(timeOut = 4000) -waits for that particular method/test to execute before throwing any error.

**Parameterising from TestNG xml file.**

Parameterising may be for both global level as well as test level.

Drive the data from xml file.

<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd">

<suite name="Loan Department">

<parameter name="URL" value="selenium.com" />

<test name="Personal Loan">

<parameter name="URL" value="personalLoan.com" />

<classes>

<class name="testng.com.TestNG">

<methods>

<include name="Demo" />

</methods>

</class>

<class name="testng.com.TestNG2" />

<class name="testng.com.HomeLoan" />

</classes>

</test>

<test name="Car Loan">

<parameter name="URL" value="carLoan.com" />

<classes>

<class name="testng.com.CarLoan">

<methods>

<exclude name='Mobile.\*' />

</methods>

</class>

</classes>

</test>

</suite>

We cant use both global level and test level parameterizing simultaneously.

Note:In the above case the test level parameterize will work not the global level.

@Parameters({ "URL" })

@Test(groups = "Smoke")

**public** **void** Demo(String URL) {

System.***out***.println("Hello");

System.***out***.println(URL);

}

@Parameters({ "URL" })

@Test

**public** **void** webLoginCarLoan(String urlName) {

// selenium

System.***out***.println("Web Login Car");

System.***out***.println(urlName);

}

Global variables can be incorporated through selenium.

<suite name="Loan Department">

<parameter name="URL" value="selenium.com" />

<test name="Personal Loan">

<parameter name="URL" value="personalLoan.com" />

<parameter name="APIKey/username" value="12345" />

<classes>

<class name="testng.com.TestNG">

<methods>

<include name="Demo" />

</methods>

</class>

<class name="testng.com.TestNG2" />

<class name="testng.com.HomeLoan" />

</classes>

</test>

<test name="Car Loan">

<parameter name="URL" value="carLoan.com" />

<classes>

<class name="testng.com.CarLoan">

<methods>

<exclude name='Mobile.\*' />

</methods>

</class>

</classes>

</test>

</suite>

@Parameters({ "URL", "APIKey/username" })

@Test(groups = "Smoke")

**public** **void** Demo(String URL, String key) {

System.***out***.println("Hello");

System.***out***.println(URL);

System.***out***.println(key);

}

**Data Provider Annotations**

Parameterizing with multiple data sets by running tests with multiple combination.-With the help of Data Provider(independent to xml file).

- to achieve parameterization with multiple data sets with multiple combinations in TestNG

@DataProvider

**public** Object[][] getData() {

// first combination -username password -good credit history = row

// 2nd -username password -no credit history

// 3rd- fraudelent credit history

Object[][] data = **new** Object[3][2];

// 1st set

data[0][0] = "firstUsername";

data[0][1] = "password";

// columns in the row are nothing but values for that particular

// combination(row)

// 2nd set

data[1][0] = "secondUsername";

data[1][1] = "secondPassword";

// 3rd set

data[2][0] = "thirdUsername";

data[2][1] = "thirdPassword";

**return** data;

}

@Test(dataProvider = "getData")

**public** **void** MobilesigninCarLoan(String username, String password) {

// Appium

System.***out***.println("Mobile SIGN IN");

System.***out***.println(username);

System.***out***.println(password);

}

**Output:**

Mobile SIGN IN

firstUsername

password

Mobile SIGN IN

secondUsername

secondPassword

Mobile SIGN IN

thirdUsername

thirdPassword

Note: The above test will run 3 times.

* We can drive data from testing xml file-with the help of @parameters annotation.
* We can drive data with the help of @DataProvider annotation as well.

**Listeners Interface for TestNG Framework**

ITestListener is an interface that implements TestNg Listeners.

**package** testng.com;

**import** org.testng.ITestContext;

**import** org.testng.ITestListener;

**import** org.testng.ITestResult;

**public** **class** Listeners **implements** ITestListener {

**public** **void** onTestStart(ITestResult result) {

// not implemented

}

**public** **void** onTestSuccess(ITestResult result) {

System.***out***.println("The test case successfully executed");

}

**public** **void** onTestFailure(ITestResult result) {

// Screen shot code can be written here

System.***out***.println("Test is failed here " + result.getName());

}

**public** **void** onTestSkipped(ITestResult result) {

// not implemented

}

**public** **void** onTestFailedButWithinSuccessPercentage(ITestResult result) {

// not implemented

}

**public** **void** onTestFailedWithTimeout(ITestResult result) {

onTestFailure(result);

}

**public** **void** onStart(ITestContext context) {

// not implemented

}

**public** **void** onFinish(ITestContext context) {

// not implemented

}

}

Note : ITestResult is a class describes the result of a test.

**Xml file:**

<listeners>

<listener class-name="testng.com.Listeners" />

</listeners>

Note: After suite declaration i.e.,after suite opening tag.

**Running Tests in Parallel and generating Reports:**

Commonly most of the test in suite run sequentially but we can run parallelly like,

<suite name="Loan Department" parallel="tests" thread-count="6">

Note: Thread-count :How many test to trigger simultaneously.

* For Mobile devices it will not work.
* Scope can be within test and package level as well.

<test name="Personal Loan" parallel="classes" thread-count="4">

**Report:**

Refresh the project > test-output folder > index.html file- right click > properties > location (path) copy > paste it on browser ---Report will be displayed.

**Java Object Oriented Principles**

**How TestNg annotation helps with inheritance to remove boiler-plate code:**

**ParentClass.java**

**public** **class** ParentClass {

**public** **void** doShow() {

System.***out***.println("I am here");

}

@BeforeTest

**public** **void** beforeRun() {

System.***out***.println("Before run....");

**}**

@AfterTest

**public** **void** afterRun() {

System.***out***.println("After run....");

}

}

**ChildClass.java**

**public** **class** ChildClass **extends** ParentClass {

@Test

**public** **void** run() {

doShow();

}

}

**How to pass values from tests through parametrized constructor and this keyword:**

this keyword refers to current class.

**ChildClass.java**

**public** **class** ChildClass **extends** ParentClass {

@Test

**public** **void** run() {

**int** a = 3;

doShow();

PS3 p = **new** PS3(a);//Parametrized constructor

System.***out***.println(p.increment());

System.***out***.println(p.decrement());

}

}

**PS3.java**

**public** **class** PS3 {

**int** a;

**public** PS3(**int** a) {

**this**.a = a;

}

**public** **int** increment() {

a = a + 1;

**return** a;

}

**public** **int** decrement() {

a = a - 1;

**return** a;

}

}

**Usage of Super keyword in the constructor to pass values to the parent class:**

**ChildClass.java**

**public** **class** ChildClass **extends** ParentClass {

@Test

**public** **void** run() {

**int** a = 3;

doShow();

PS3 p = **new** PS3(a); // Parametrized constructor

System.***out***.println(p.increment());

System.***out***.println(p.decrement());

System.***out***.println(p.multiplyThree());

}

}

**PS3.java**

**public** **class** PS3 **extends** PS4 {

**int** a;

**public** PS3(**int** a) {

**super**(a);

**this**.a = a;

}

**public** **int** increment() {

a = a + 1;

**return** a;

}

**public** **int** decrement() {

a = a - 1;

**return** a;

}

}

**PS4.java**

**public** **class** PS4 {

**int** a;

**public** PS4(**int** a) {

**this**.a = a;

}

**public** **int** multiplyThree() {

a = a \* 3;

**return** a;

}

}

**FrameWork -Part 1:** **Create Maven Project and Prepare functional end to end testing using TestNG framework.**

**Create Maven project and all framework dependencies:**

**Archetype:** maven-archetype-quickStart

**groupId** :org.apache.maven.archetypes

**dependencies:** selenium java ,TestNg, WebDriverManager

-End to End testing for -https://rahulshettyacademy.com/client/

-Login

-Add to Cart

-check-out

-place order

-Confirmation page

**FrameWork -Part 2:** **Design Pattern, Page object & factory implementation.**

**Page Object Models**

@FindBy(id = "userEmail")

This annotation will locate the element by loctors similar to that of below one.

WebElement userEmail=driver.findElement(By.*id*("userEmail")).sendKeys(userName);

* This annotation will have WebDriver knowledge through PageFactory by calling initElements(driver,this) methods.

-two arguments one -driver second-this(current class)

**public** LandingPage(WebDriver driver) {

**this**.driver = driver;

PageFactory.*initElements*(driver, **this**);

}

Note:Hands on- SeleniumDesignFramework project

**FrameWork -Part 3:Test Configuration methods & Global Properties & Parallel run**

Global Properties file must be under Java.

Test Component must be under Test.

* Creating Base Test which holds common Test configuration method.
* Initialize Driver and create utility to launch app with BeforeMethod annotation
* Create new Error validation Test as per framework Standards developed until now
* Implementation Test Strategy for Framework on how tests are divided based on modules
* Create new Test methods with Dependency attribute based on Test Strategy design
* How to run tests/classes in parallel & apply groups using TestNG.xml

**FrameWork -Part 4:Test Strategy-Control Tests Execution-Run Parallel Tests**

**Agenda of implementing Parameterization into tests with TestNG Data provider**

@Test(dataProvider = "getData", groups = "Purchase")

**public** **void** submitOrder(String userName, String password, String productName)

**throws** InterruptedException, IOException {

String countryName = "india";

ProductCatalog catalog = landingPage.login(userName, password);

catalog.addToCart(productName);

CartPage cartPage = catalog.goToCartPage();

Boolean match = cartPage.verifyProductDisplay(productName);

Assert.*assertTrue*(match);

CheckoutPage checkOutPage = cartPage.goToCheckOutPage();

checkOutPage.selectCountry(countryName);

ConfirmationPage confPage = checkOutPage.confirmationPage();

String confirmMessage = confPage.validateOrderPlacement();

Assert.*assertTrue*(confirmMessage.equalsIgnoreCase("THANKYOU FOR THE ORDER."));

}

@Test(dependsOnMethods = { "submitOrder" })

**public** **void** orderHistoryPage() {

ProductCatalog catalog = landingPage.login(userName, password);

OrderHistoryPage orderPage = catalog.goToOrderPage();

Assert.*assertTrue*(orderPage.verifyOrderHistory(productName));

}

@DataProvider

**public** Object[][] getData() {

**return** **new** Object[][] { { "gagana@gmail.com", "Gagana@12345", "ZARA COAT 3" },

{ "kiranguru@gmail.com", "Kiran@12345", "ADIDAS ORIGINAL" } };

}

**Integration of Hashmap to Data provider to send the data as one Hash object**

@Test(dataProvider = "getData", groups = "Purchase")

**public** **void** submitOrder(HashMap<String, String> input) **throws** InterruptedException, IOException {

String countryName = "india";

ProductCatalog catalog = landingPage.login(input.get("email"), input.get("password"));

catalog.addToCart(input.get("productName"));

CartPage cartPage = catalog.goToCartPage();

Boolean match = cartPage.verifyProductDisplay(input.get("productName"));

Assert.*assertTrue*(match);

CheckoutPage checkOutPage = cartPage.goToCheckOutPage();

checkOutPage.selectCountry(countryName);

ConfirmationPage confPage = checkOutPage.confirmationPage();

String confirmMessage = confPage.validateOrderPlacement();

Assert.*assertTrue*(confirmMessage.equalsIgnoreCase("THANKYOU FOR THE ORDER."));

}

@Test(dependsOnMethods = { "submitOrder" })

**public** **void** orderHistoryPage() {

ProductCatalog catalog = landingPage.login(userName, password);

OrderHistoryPage orderPage = catalog.goToOrderPage();

Assert.*assertTrue*(orderPage.verifyOrderHistory(productName));

}

@DataProvider

**public** Object[][] getData() {

HashMap<String, String> map = **new** HashMap<String, String>();

map.put("email", "gagana@gmail.com");

map.put("password", "Gagana@12345");

map.put("productName", "ZARA COAT 3");

HashMap<String, String> map1 = **new** HashMap<String, String>();

map1.put("email", "kiranguru@gmail.com");

map1.put("password", "Kiran@12345");

map1.put("productName", "ADIDAS ORIGINAL");

**return** **new** Object[][] { { map }, { map1 } };

}

**How to read the data from Json files and create the list of Hashmaps for testing**

* First Read the json file to String using FileUtils.readFileToString(),
* Convert String file to hashMap – for that dependency ->Jackson DataBind add to pom.xml

**public** List<HashMap<String, String>> getJsonData() **throws** IOException {

// Read File to String

String jsonContent = FileUtils.*readFileToString*(

**new** File(System.*getProperty*("user.dir") + "\\src\\test\\java\\selenium\\com\\data\\PurchaseOrder.json"),

StandardCharsets.***UTF\_8***);

// Convert string file to hash map -> dependency JackSon DataBind

ObjectMapper mapper = **new** ObjectMapper();

List<HashMap<String, String>> data = mapper.readValue(jsonContent,

**new** TypeReference<List<HashMap<String, String>>>() {

});

**return** data;

}

@DataProvider

**public** Object[][] getData() **throws** IOException {

List<HashMap<String, String>> data = getJsonData(

System.*getProperty*("user.dir") + "\\src\\test\\java\\selenium\\com\\data\\PurchaseOrder.json");

**return** **new** Object[][] { { data.get(0) }, { data.get(1) } };

}

**How to Create Screenshot Utility in Base Test class for catching Failed tests**

**public** String takeScreenShot(String testCase) **throws** IOException {

TakesScreenshot ts = (TakesScreenshot) driver;

File source = ts.getScreenshotAs(OutputType.***FILE***);

File file = **new** File(System.*getProperty*("user.dir") + "//reports//" + testCase + ".png");

FileUtils.*copyFile*(source, file);

**return** System.*getProperty*("user.dir") + "//reports//" + testCase + ".png";

}

**Framework Part 5 -Extent HTML reports & TestNG Listeners & Thread Safe execution**.

**What is Extent Report:**

**Phase 1:**

Understand the basic configuration of extent report with Standalone test.

**Phase 2:**

Generate Report

Extent Report + testNGListeners

**Dependencies-**>selenium java, testNg, Extent Report

Two classes are important in extentReport : ExtentReports ,ExtentSparkReporter

**public** **class** ExtentReportDemo {

ExtentReports extent;

@BeforeMethod

**public** **void** config() {

// ExtentReports ExtentSparkReporter

String path = System.*getProperty*("user.dir") + "\\reports\\index.html";

ExtentSparkReporter reporter = **new** ExtentSparkReporter(path);

reporter.config().setReportName("Web Automation Report");

reporter.config().setDocumentTitle("Test Results");

extent = **new** ExtentReports();

extent.attachReporter(reporter);

extent.setSystemInfo("Tester", "Gagana");

}

@Test

**public** **void** initialize() {

ExtentTest test = extent.createTest("Intili Demo");

WebDriver driver = **new** ChromeDriver();

driver.manage().window().maximize();

driver.get("https://integration.dotfoods.com/shop");

driver.getTitle();

driver.close();

test.fail("Results do not Match");

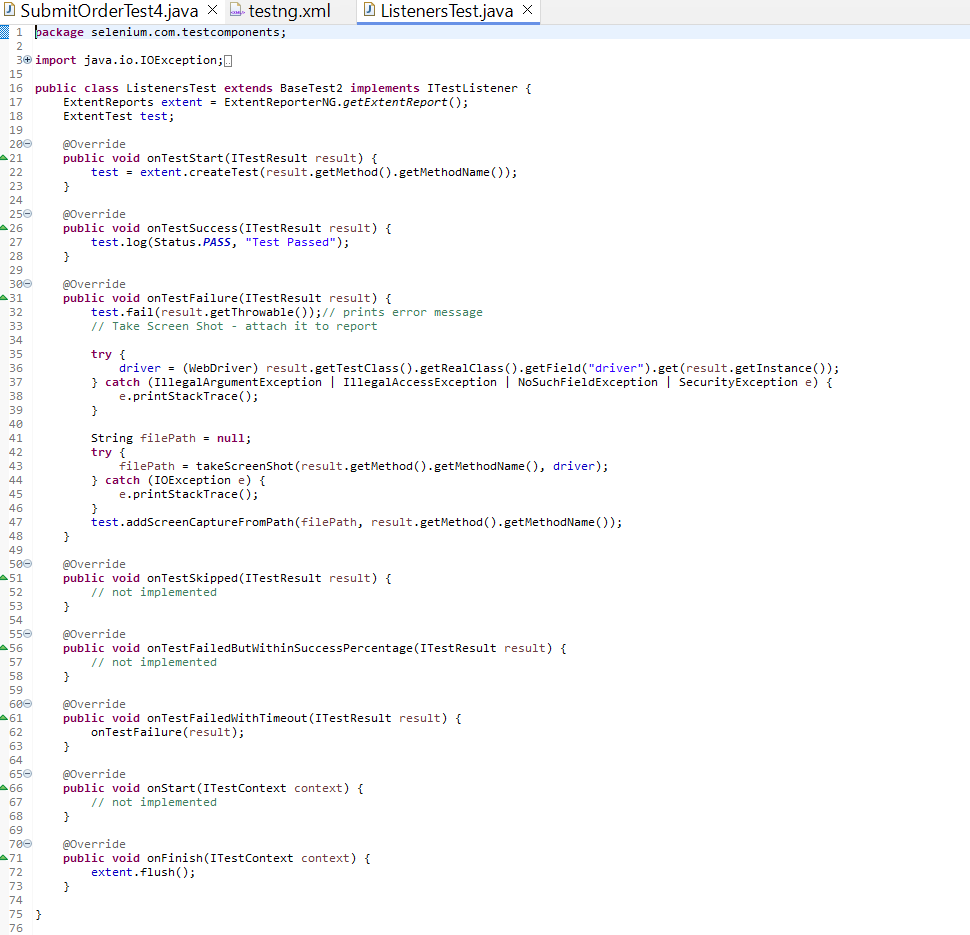
extent.flush();

}

}

**Integrating Extent reports to existing framework with help of TestNG listeners**

ListenersTest.java



**ExtentReporterNG.java**



**BaseTest2.java -Screen shot Code:**

**public** String takeScreenShot(String testCase, WebDriver driver) **throws** IOException {

TakesScreenshot ts = (TakesScreenshot) driver;

File source = ts.getScreenshotAs(OutputType.***FILE***);

File file = **new** File(System.*getProperty*("user.dir") + "//reports//" + testCase + ".png");

FileUtils.*copyFile*(source, file);

**return** System.*getProperty*("user.dir") + "//reports//" + testCase + ".png";

}

**TestNg.xml**

<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd">

<suite name="Suite">

<listeners>

<listener class-name="selenium.com.testcomponents.ListenersTest" />

</listeners>

<test thread-count="5" name="Submit Order Test">

<classes>

<class name="selenium.com.SubmitOrderTest4" />

</classes>

</test>

<test thread-count="5" name="Error Validation Test">

<classes>

<class name="selenium.com.ErrorValidationTest" />

</classes>

</test>

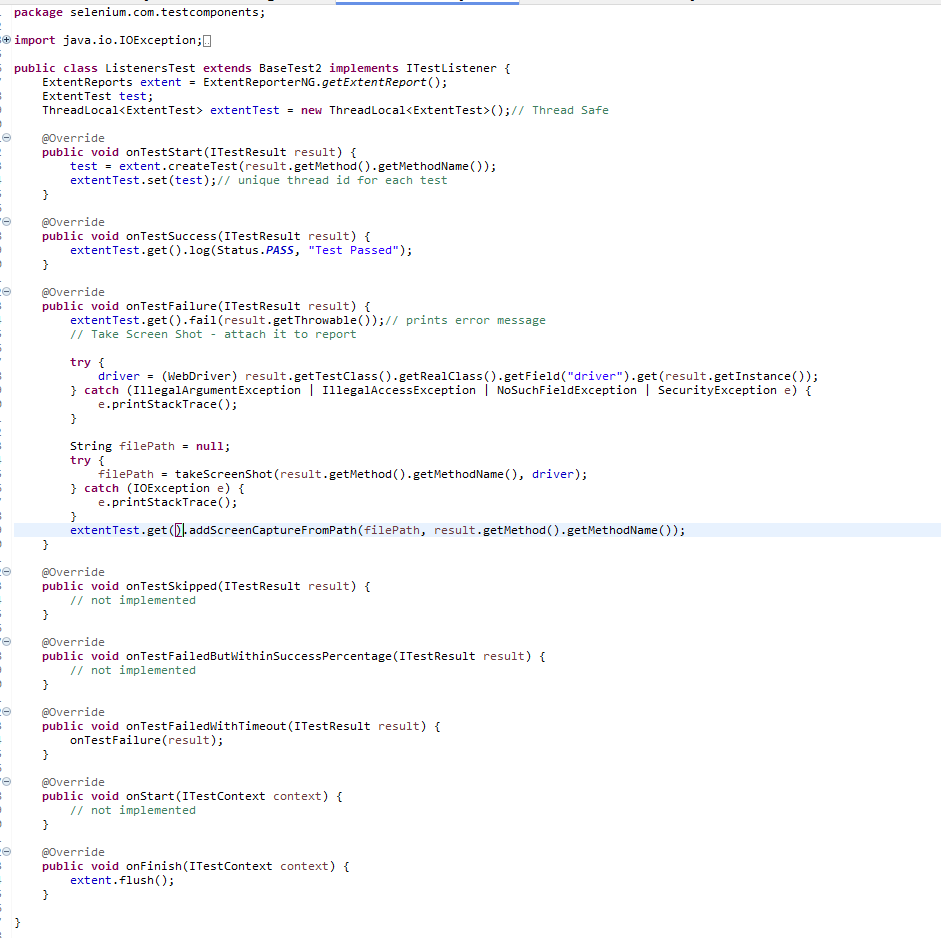
</suite>

**concurrency problem -Implement ThreadLocal class to avoid sync issues in Tests**

To do that we can use ThreadLocal class to make thread safe :synchronization

ThreadLocal<ExtentTest> extentTest = **new** ThreadLocal<ExtentTest>();// Thread Safe

extentTest.set(test);// unique thread id for each test



**Testng.xml**

<suite parallel="tests" name="Suite">

**IRetry Analyzer to rerun the flaky failed Selenium tests in the framework**

Due to some reasons the test will fail without any actual mistake, that time we should re-run the failed test case ,

**Interface-** IRetryAnalyzer

**Retry.java**

**package** selenium.com.testcomponents;

**import** org.testng.IRetryAnalyzer;

**import** org.testng.ITestResult;

**public** **class** Retry **implements** IRetryAnalyzer {

**int** count = 0;

**int** maxTry = 1;

@Override

**public** **boolean** retry(ITestResult result) {

**if** (count < maxTry) {

count++;

**return** **true**;

}

**return** **false**;

}

}

**ErrorValidation.java**

@Test(groups = "ErrorHandling", retryAnalyzer = Retry.**class**)

**public** **void** loginErrorValidation() {

String userName = "gagana@gmail.com";

String password = "Gagana@12345345";

landingPage.login(userName, password);

Assert.*assertEquals*("Incorrect email password.", landingPage.getLoginErrorMessage());

}

**Framework Part 6-Test Execution From Maven & Integration with Jenkins CI/CD**

How to run tests in the framework from terminal using Maven commands

Maven Download:

* Search-> Maven Download -> download Zip file ->extract the zip file .
* Set the system variables->search bar – env-> environment Variable->new – MAVEN\_HOME : paste the path .
* Click on path-> edit-> new -> paste the path upto bin.
* Cmd->mvn -version
* <https://www.qamadness.com/knowledge-base/how-to-install-maven-and-configure-environment-variables/>

Maven Testng plugin:

* Search->Maven Testng integration

1. <plugin>
2. <groupId>org.apache.maven.plugins</groupId>
3. <artifactId>maven-surefire-plugin</artifactId>
4. <version>3.3.0</version>
5. <configuration>
6. <suiteXmlFiles>
7. <suiteXmlFile>testng.xml</suiteXmlFile>
8. </suiteXmlFiles>
9. </configuration>
10. </plugin>

* Add to pom.xml(multiple profile can be created like this)

A computer screen shot of a program code

Description automatically generated

* In pom.xml set compiler.sorce and compiler.target to 1.8



To run tests:

* mvn test -PRegression (P stands for profile followed by profile name)

Set Global Parameters using Maven commands and Update tests at run time

* mvn tetst -PRegression -Dbrowser=FireFox (D stands for maven parameter)

String browser = System.*getProperty*("browser") != **null** ? System.*getProperty*("browser")

: prop.getProperty("browser");

// String browser = prop.getProperty("browser");

**Install Jenkins in the local system for CI/CD:**

* Search Jenkins Download: <https://www.jenkins.io/download/>
* Download : Generic Java Package (.war)
* Open the cmd navigate upto war file is present

**To start Jenkins** :java -jar jenkins.war --httpPort=9090

**If First**:

-Copy the administration password

-Open browser : localhost:9090

-paste the administration password

Set password

Integrate the Selenium framework with Jenkins and Parameterize jenkin job

* New Item -> Selenium Framework (FreeStyle Project) -> ok
* Advanced drop down ->User custom workspace-> paste the path
* Build -> Invoke top-level Maven targets-> provide cmds in goal field->

**test -PRegression -Dbrowser=chrome** (no need of mvn)->save

* Build now

Parameterize:

* Click on configure
* Click on **This project is parameterized?** -> Add parameter-> choice parameter

Name : browserName

Choice: chrome

edge

firefox

* update cmd -> configure->cmd -> **test -PRegression -Dbrowser="$browserName"->** save
* Build with parameters :chrome

Note: many choice parameter can be included

* Click on **This project is parameterized?** -> Add parameter-> choice parameter

Name : Profile

Choice: Regression

Purchase

ErrorValidation

* update cmd -> configure->cmd -> **test -P”$Profile” -Dbrowser="$browserName"->** save
* Build with parameters :chrome

 How to Run tests in headless mode and integrate the parameter in Jenkins

Headless->browser less – invisible

Using **ChromeOptions** class

String browser = System.*getProperty*("browser") != **null** ? System.*getProperty*("browser")

: prop.getProperty("browser");

// String browser = prop.getProperty("browser");

**if** (browser.contains("chrome")) {

ChromeOptions options = **new** ChromeOptions();

**if** (browser.contains("headless")) {

options.addArguments("headless");

}

driver = **new** ChromeDriver(options);

driver.manage().window().setSize(**new** Dimension(1440, 800)); // run in full screen

}

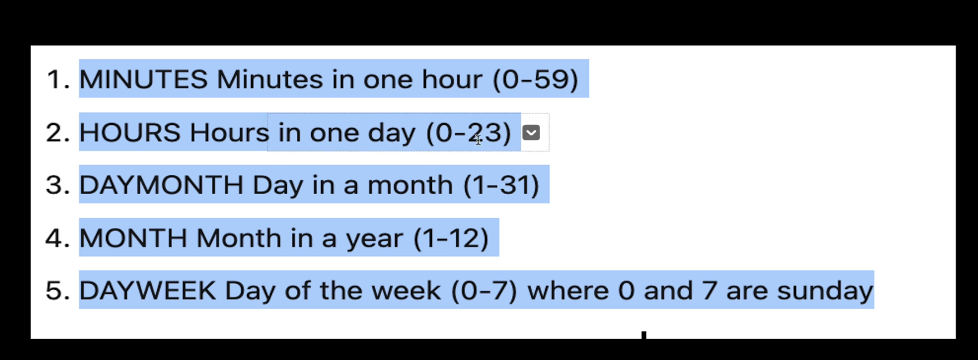
Schedule Jenkin Jobs with regular expression and trigger nightly Automation job

Configuration -> Build triggers->Build Periodically-> \* \* \* \* \*

Second parameter- Hour

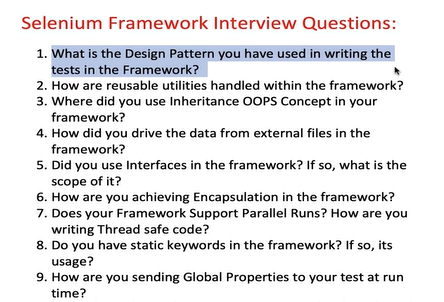
First parameter -minutes

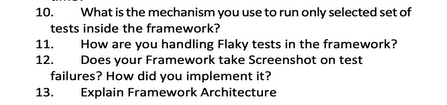
Third parameter – day of the month



Note: H 6 \* \* \* -> any hour at 6 AM

**Framework Part 7- Common Selenium Framework OOPS Interview Questions.**

****

****

1. Page Object -using PageFactory.
2. AbstractClass – with common methods

And all other class extending AbstractClass

In Case of Test we are using BaseTest and all other test class extending BaseTest

Reusable classes =AbstractClass ,BaseTest

1. Related to question 2.
2. With help of DataProvider -using json file and using parameterized concepts.
3. Interfaces Listerners implementing ITestListener, WebDriver Interface.
4. Fields of the class in Page Objects by making private ,WebElements -private.
5. Yes, in testNg Xml file we are setting test to run parallel.

<suite parallel="tests" name="Suite">

Thread Safe -each tests run in different threads.

But in Generating report we came across an issue with this for that we have used **ThreadLocal** class to handle.

1. static keyword is commonly shared across all the threads. If is supports parallel run then I don’t want to see concurrency problem hence I prefer no. but if necessary and not leading to any synchronization problem then can use it.
2. GlobalProperties.properties file, using this file I am reading data in BaseTest using Properties class.

Even through maven commands also can be achieved.

1. Using TestNg groups

<groups>

<run>

<include name="ErrorHandling" />

</run>

</groups>

1. Using IRetryAnalyzer interface.
2. Yes it takes,Using TestNg Listeners when test failes it takes screenshot and attach the screen to report.
3. Architecture

**Framework Part 8 - Integrating Cucumber Wrapper into Selenium Framework.**

**How much Cucumber does this course cover:**

Cucumber -Is one of the exclusive BDT framework which helps to design test automation framework.

1.Understanding of cucumber

2.Cucumber Integration to the existing framework

**Introduction to cucumber and its terminologies with examples:**

Cucumber Terminologies:

What is Gherkin?

It is a language to describe the software behaviour.

Example: Pop up messaged is displayed when buttons are clicked and errors are gone.

**Keywords Used in cucumber**: Scenario, Feature, Feature file, Scenario outline, Step Definition.

**Scenario:**

In cucumber TestCases are represented as Scenario.

Scenario contain steps which are equivalent to test steps and use the following keywords (Gherkin syntax) to denote them: Given ,When, Then, But and And (case Sensitive)

* **Given:** Precondition are mentioned in the Given Keyword.
* **When:** The purpose of the When steps is to describe the user action.
* **Then:** The purpose of Then steps is to observe the expected output. The observations should be related to the business value/benefit of your Feature description.

**Scenario:** Make minimum Due Payment

* **Given** User is on pay credit card page.
* **When** User fills all details and select Minimum amount option**.**
* **And** User clicks on pay button.
* **Then** Credit card confirmation page is displayed.

**Note: And** if reference number is displayed

**But** error message is not displayed.

When we specify a business requirement, sometimes there are multiple pre-conditions, user actions, and expected outcomes.

We are going to add one more Scenario and will use the And and But keywords:

* **And:** This is used for statements that are an addition to the previous steps and represent positive statements.
* **But:** This is used for statements that are an addition to the previous steps and represent negative statements.

**Feature and Feature File:**

Feature represents Business requirement.

Feature File acts as a Test suite which consists of all scenarios.

In Cucumber , Feature files contain Scenarios. We can simply create feature file with .feature extension.

Scenarios belonging to specific area of application will be grouped into one Feature file.

The text that immediately follows the Feature keyword, and is in the same line, is the Title of the Feature file.

Feature file should contain either scenario or scenario Outline. The naming conventions for Feature files should be lowercase with .feature extension.

**Feature :** Credit card payment

In order to test Credit Card Payment functionality

As a CC user

I want to complete the payment through online

**Scenario :** Make Minimum Due Payment

**Given** user is on Pay Credit card page

**Then** user fills all details and select minimum amount option

**And** user clicks on Pay button

**Then** Credit card confirmation page is displayed.

**Scenario :** Make Minimum Due Payment

**Given** user is on Pay Credit card page

**Then** user fills all details and select Statement balance option.

**And** user clicks on Pay button

**Then** Credit card confirmation page is displayed.

**Scenario :** Make Minimum Due Payment

**Given** user is on Pay Credit card page

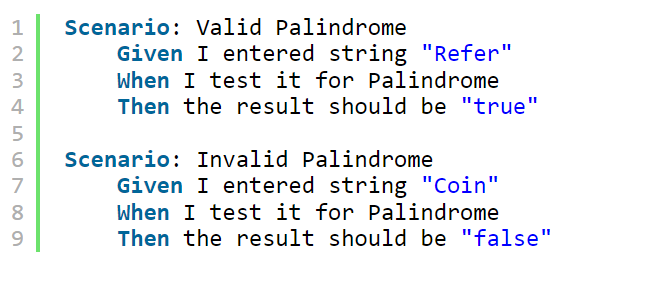
**Then** user fills all details and select other Amount and enter 0

**And** user clicks on Pay button

**Then** Credit card confirmation page is not displayed.

**But** error message is displayed.

**Scenario outline**: <https://javapointers.com/automation/cucumber/cucumber-scenario-outline-example/>



**A computer code with blue text

Description automatically generated**

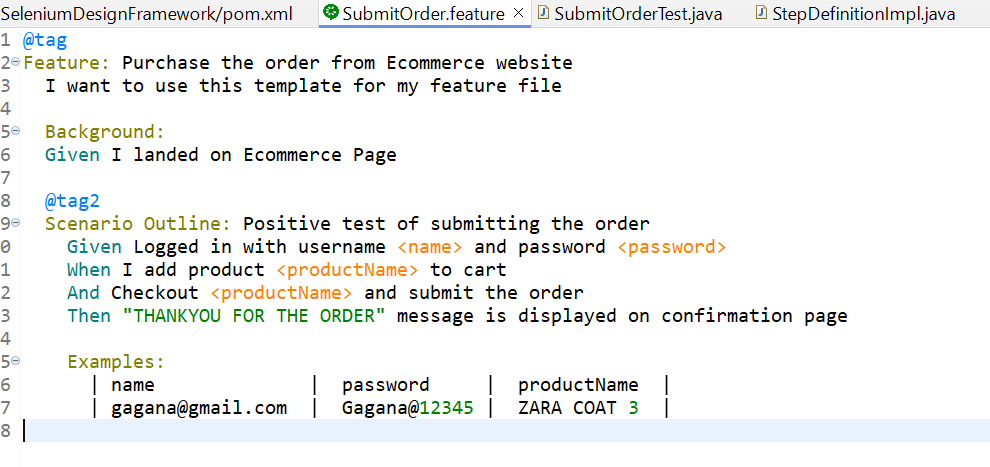
**Setting up cucumber dependencies into Framework and create feature files.**

Go to Maven repo:

Dependencies : Cucumber java , Cucumber TestNg-> add it to pom.xml

Pulgin download :Go to help-> Eclipse marketspace->type cucumber and install >restart the eclipse.

Create Package under src/test/java : and create package and create a feature file with .feature extension.



**Implement step definitions for features and understand regular expression.**

Create one more package for step definition and create one java file.

Note : for the conditions :

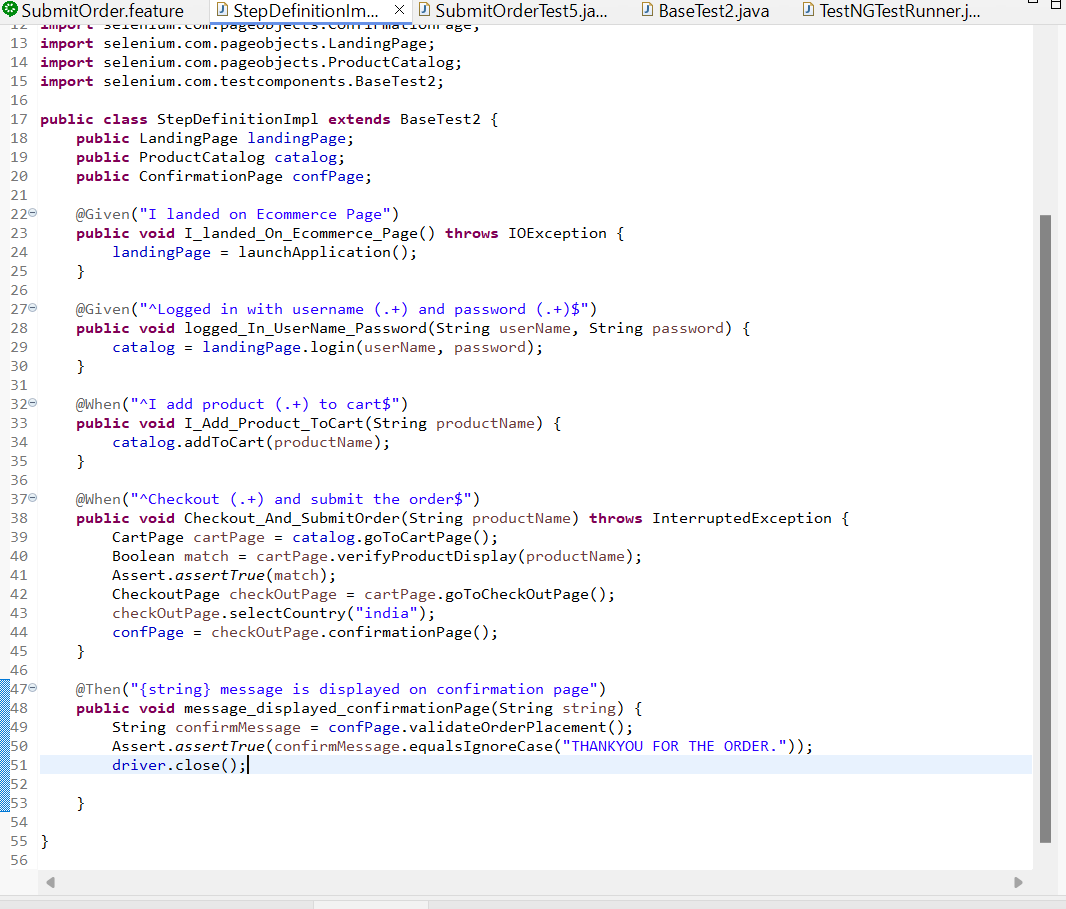
Given Logged in with username <name> and password <password>

we will write step definition like this: @Given("^Logged in with username (.+) and password (.+)$")

because the parameters or data should be in regular expression.For Regular expression it should start with ^ and end with $ symbol

for the conditions :Then "THANKYOU FOR THE ORDER" message is displayed on confirmation page

we will write step definition like this: @Then("{string} message is displayed on confirmation page")



**Inject Selenium code in step definition and introduction to Tidy:**

Search for Tidy plugin in chrome (chrome plugin).

Copy the entire feature file and paste it we will get entire step definitions.

**Introduction to TestNG test runner to run cucumber feature files:**

Create one more class in feature file package :

@CucumberOptions(features = "src/test/java/selenium.com.cucumber", glue = "selenium.com.stepDefinitions", monochrome = **true**, plugin = {

"html:target/cucumber.html" })

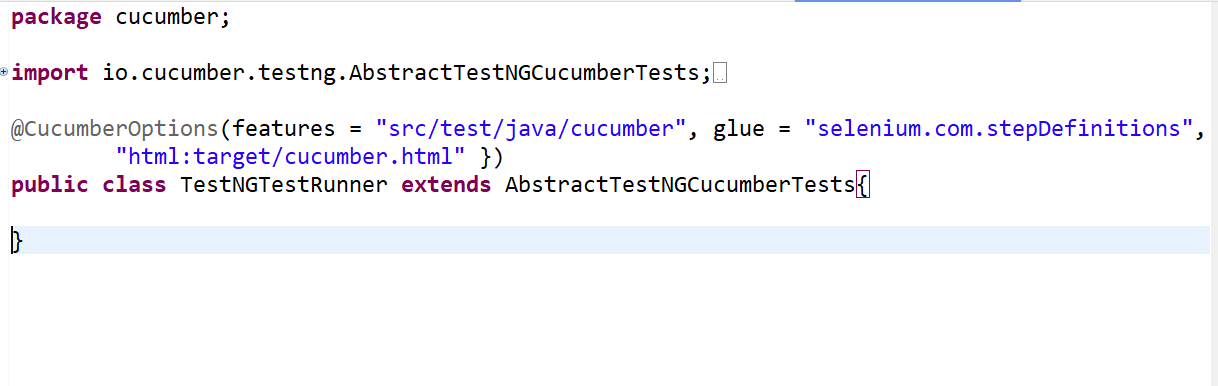
* monochrome: to make the output in readable manner.
* Glue is for step definition
* Features is for feature file

The created class should extends AbstractTestNGCucumberTests

Note: if your using Junit then no need to extend class with AbstractTestNGCucumberTests because cucumber have capability to handle all the wrapper methods

But in case of TestNg it should extend AbstractTestNGCucumberTests class.

* Run the TestNGTestRunner class same as testNg it will run on cucumber.



**Control the cucumber feature files execution with Tags and Background Keywords.**

**SubmitOrder.feature**

@tag

Feature: Purchase the order from Ecommerce website

I want to use this template for my feature file

Background:

Given I landed on Ecommerce Page

@Regression

Scenario Outline: Positive test of submitting the order

Given Logged in with username <name> and password <password>

When I add product <productName> to cart

And Checkout <productName> and submit the order

Then "THANKYOU FOR THE ORDER" message is displayed on confirmation page

Examples:

| name | password | productName |

| gagana@gmail.com | Gagana@12345 | ZARA COAT 3 |

**ErrorValidation.feature**

@tag

Feature: ErrorValidation

I want to use this template for my feature file

@ErrorValidation

Scenario Outline: Login Page error validation

Given I landed on Ecommerce Page

When Logged in with username <name> and password <password>

Then "Incorrect email password" message is displayed

Examples:

| name | password |

| gagana@gmail.com | Gagana@123 |

**TestNgTestRunner.java**

package cucumber;

import io.cucumber.testng.AbstractTestNGCucumberTests;

import io.cucumber.testng.CucumberOptions;

//Cucumber -> TestNG or Junit

@CucumberOptions(features = "src/test/java/cucumber", glue = "selenium.com.stepDefinitions", monochrome = true, tags = "@ErrorValidation", plugin = {

"html:target/cucumber.html" })

public class TestNGTestRunner extends AbstractTestNGCucumberTests {

}

**To run through cmds-> Make changes in pom.xml as**

Add Profile in pom.xml

<profile>

<id>CucumberTests</id>

<build>

<pluginManagement><!-- lock down plugins versions to avoid using

Maven

defaults (may be moved to parent pom) -->

<plugins>

<plugin>

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

<artifactId>maven-surefire-plugin</artifactId>

<version>3.3.0</version>

<configuration>

<includes>

<include>\*\*/TestNGTestRunner.java</include>

</includes>

</configuration>

</plugin>

</plugins>

</pluginManagement>

</build>

</profile>

Cmd to to execute:mvn test PCucumberTests

**GIT- Version Control**

**CI/CD Integration of Selenium Framework with Jenkins & GitHub**

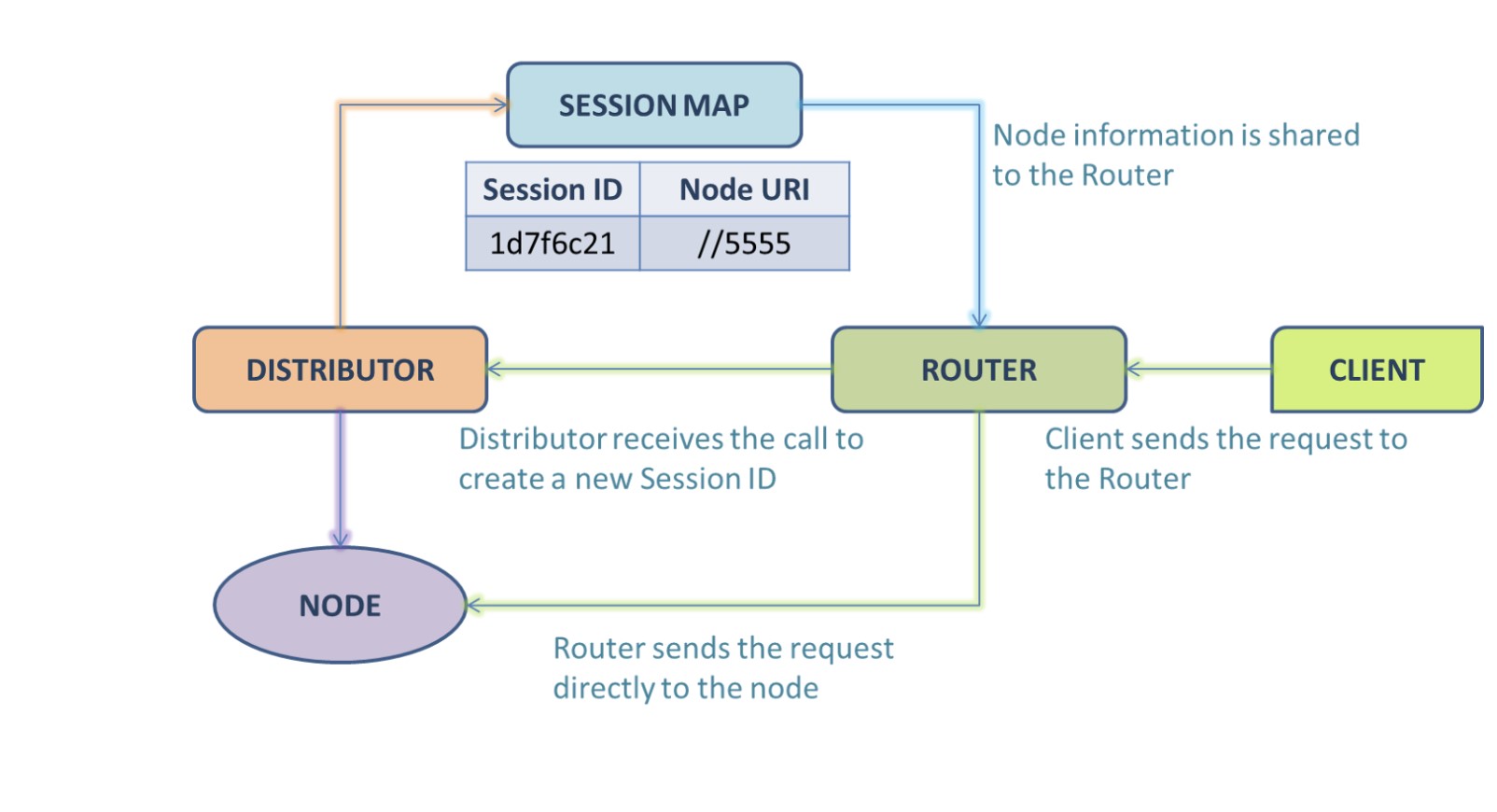
**Cross Browser Testing with Selenium Grid**

**What is Selenium Grid? Its advantages on bringing down execution time.**

What is Selenium Grid?

Selenium Grid is a smart proxy server that makes it easy to run tests in parallel on multiple machines.

# **Selenium Grid Architecture**



**A diagram of a company

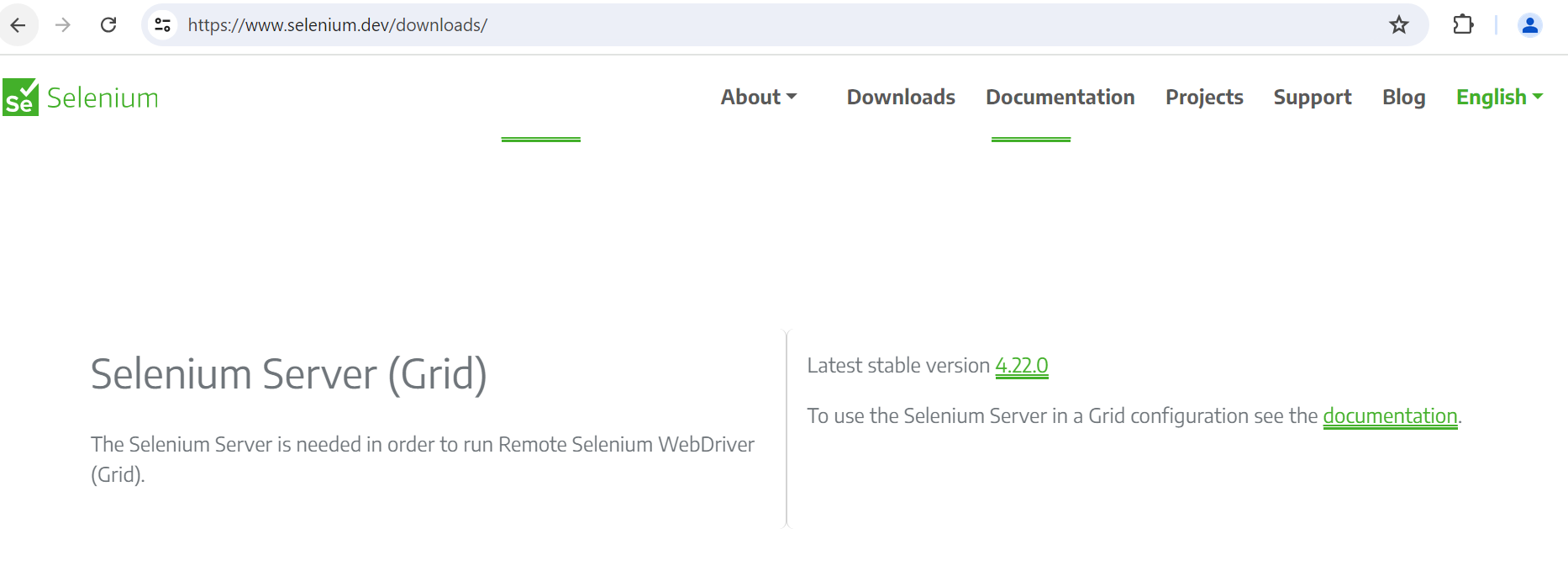
Description automatically generated**

**Getting Started with Grid Infrastructure Setup - Create Components**

**Documentation**: <https://www.selenium.dev/documentation/grid/getting_started/>

Search : <https://www.selenium.dev/downloads/>

-Download Selenium Server (Grid ) jar



**Selenium Grid Setup Instructions-**

**Step 1:**

* Download the Selenium Server

Download Browser drivers and place in the same path where Selenium server is located.

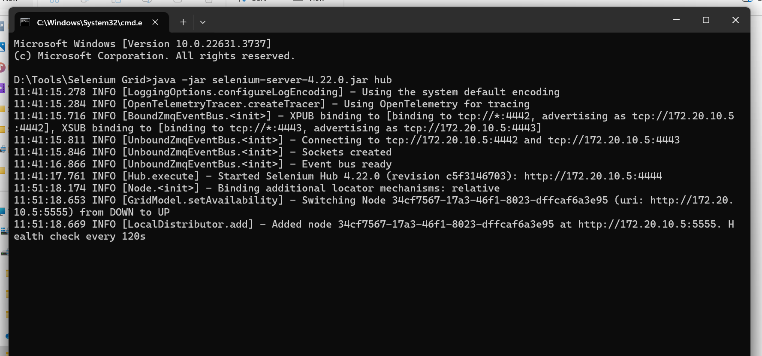
A screenshot of a computer

Description automatically generated

Navigate to that folder in cmd

**Step 2:**

* Start the Hub - which eventually Starts Router, Distributor, Session Map , New Session Queue, Event Bus
* **java -jar <SeleniumJarname> hub** [java -jar selenium-server-4.22.0.jar hub]

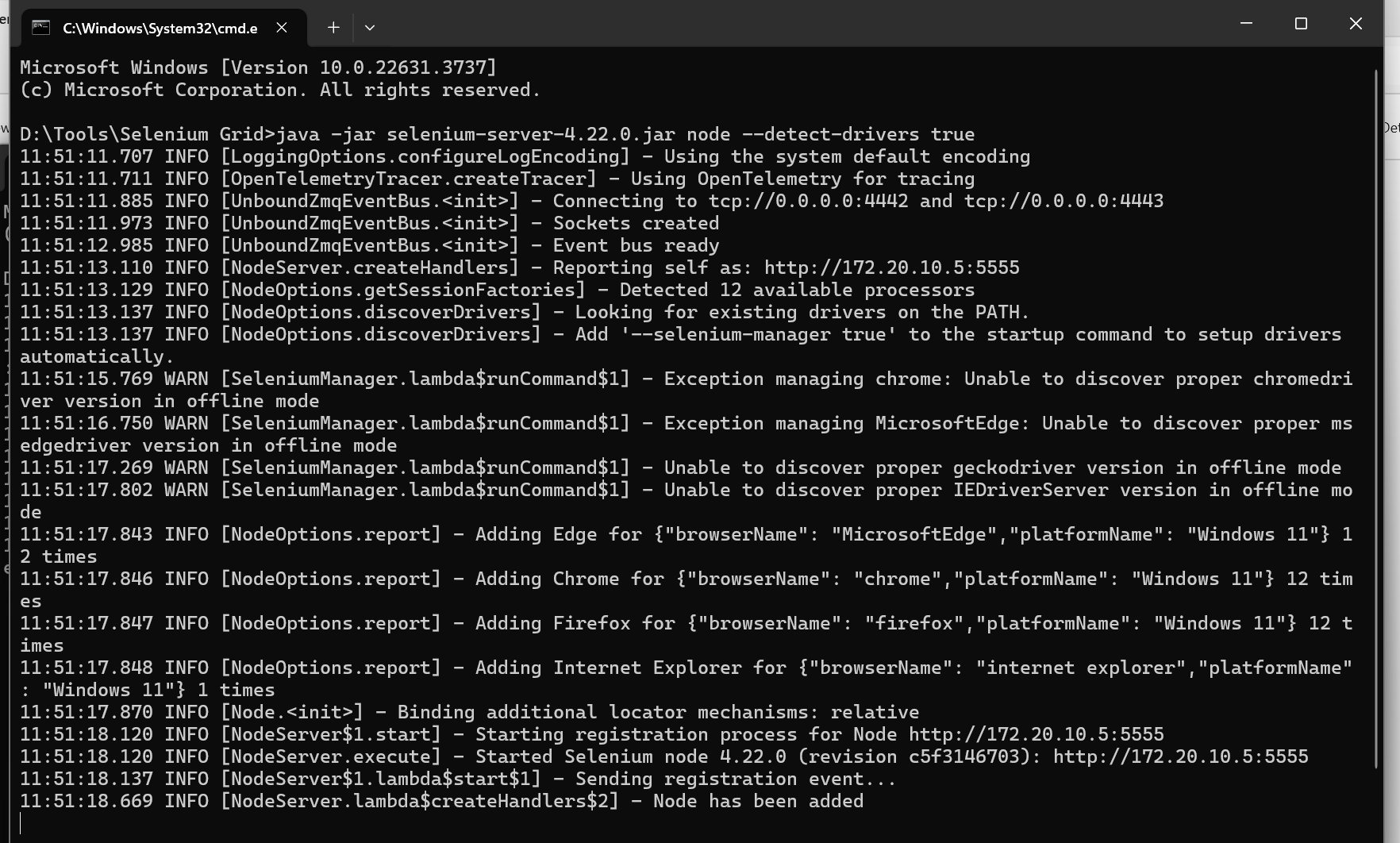


Note: To see the status of (Selenium grid) running hub : localhost:4444 -(chrome)

**Step 3:**

* Start the Node in Same Machine where Hub is running

**java -jar <SeleniumJarname> node --detect-drivers true**

****

**Step 4:**

* Start the Node in different Physical Machine (Team-viewer -for this we need to install browser and do the first step again )

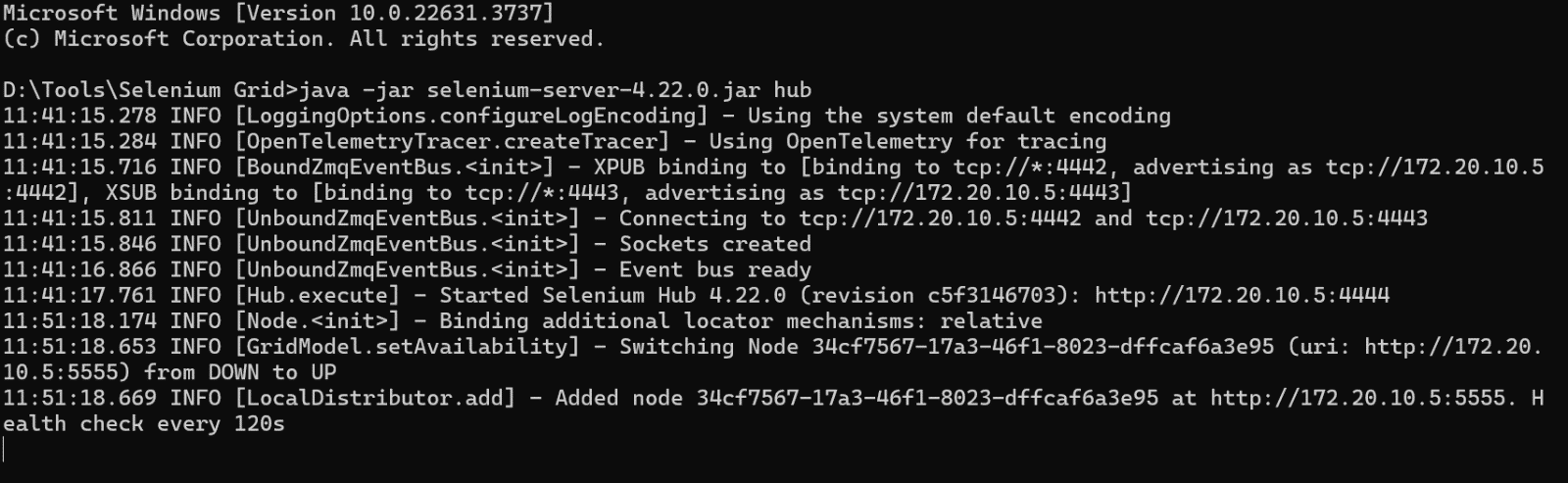
**java -jar <SeleniumJarname> node --detect-drivers true --publish-events tcp://<ipaddressofhub> --subscribe-events tcp:// <ipaddressofhub>**

Note:tcp :transmission control protocol

**Create Selenium TestNG tests with Desired capabilities & remote webdriver class**

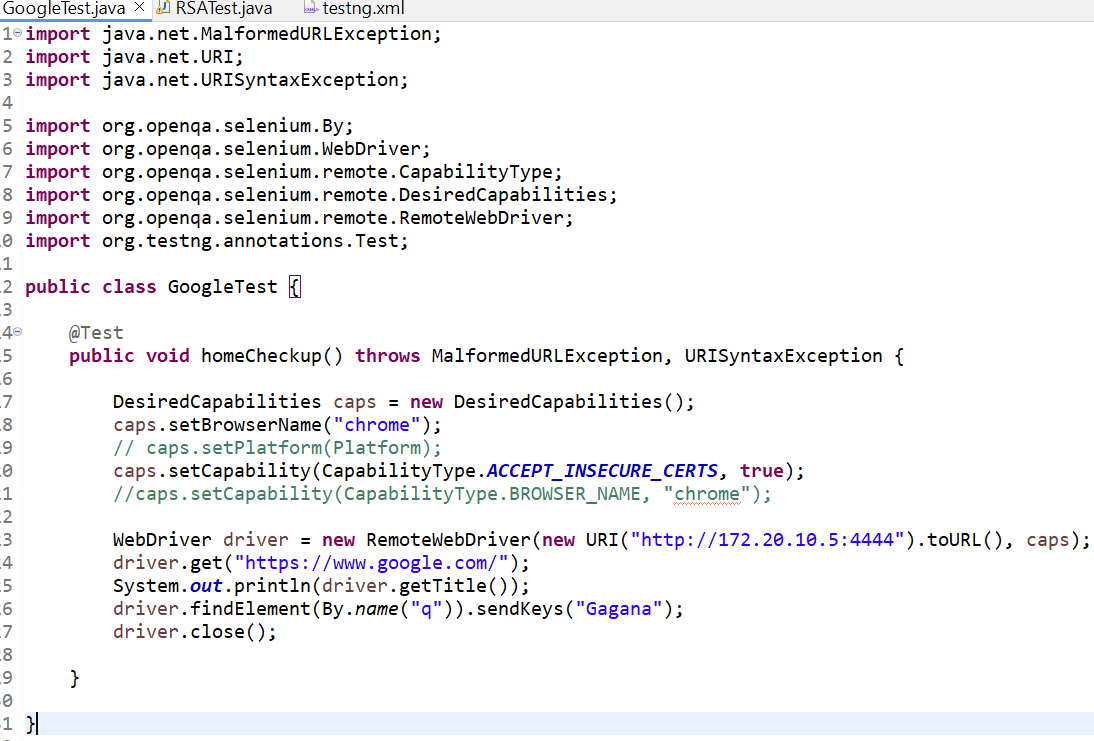
Create Java Project and add Selenium Java jar file and create class.

RemoteDriver() class accepts to arguments :- 1.Selenium starting Hub ip address



2.capabilities : on which browser it should run

DesiredCapabilities()



A screenshot of a computer code

Description automatically generated

Convert to TestNg

A screenshot of a computer program

Description automatically generated

**Understand Excel Data Driven Testing functions with examples.**

**What is Apache POI API & Download Instructions**

**Apache POI API – Api to connect excel to java test case.(open source)**

**Data driven testing from Excel.**

Maven Dependencies Set up:

Poi-ooxml and poi

<dependencies>

<!-- https://mvnrepository.com/artifact/org.apache.poi/poi-ooxml -->

<dependency>

<groupId>org.apache.poi</groupId>

<artifactId>poi-ooxml</artifactId>

<version>5.2.5</version>

</dependency>

<!-- https://mvnrepository.com/artifact/org.apache.poi/poi -->

<dependency>

<groupId>org.apache.poi</groupId>

<artifactId>poi</artifactId>

<version>5.2.5</version>

</dependency>

</dependencies>

**Strategy to Access Excel Data:**

Create object for XSSFWorkbook class – get Hold to excel

Get Access to sheet

Get Access to all rows of sheet

Access to specific row from all rows

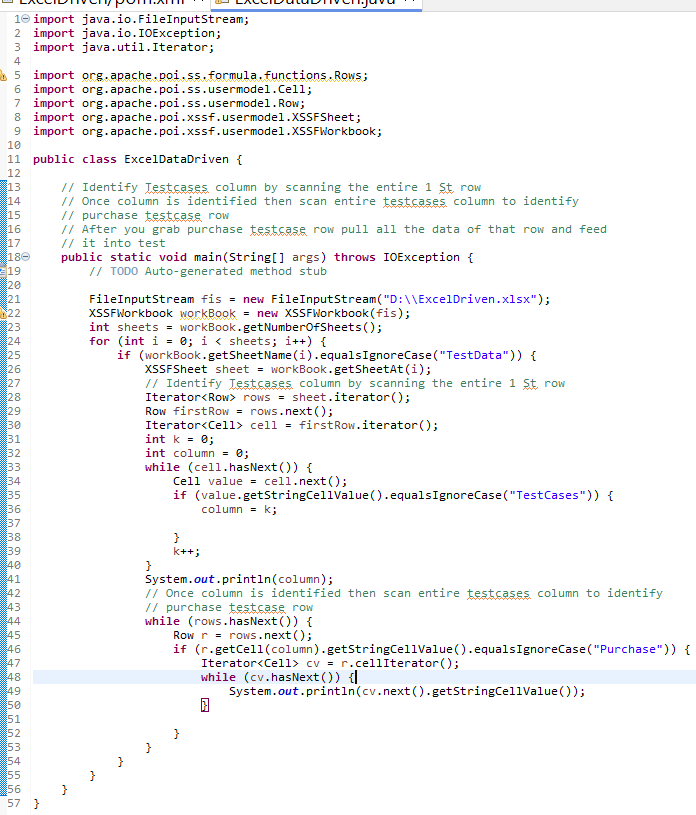
Get Access to all cells of Row

Access the Data from Excel into Arrays.

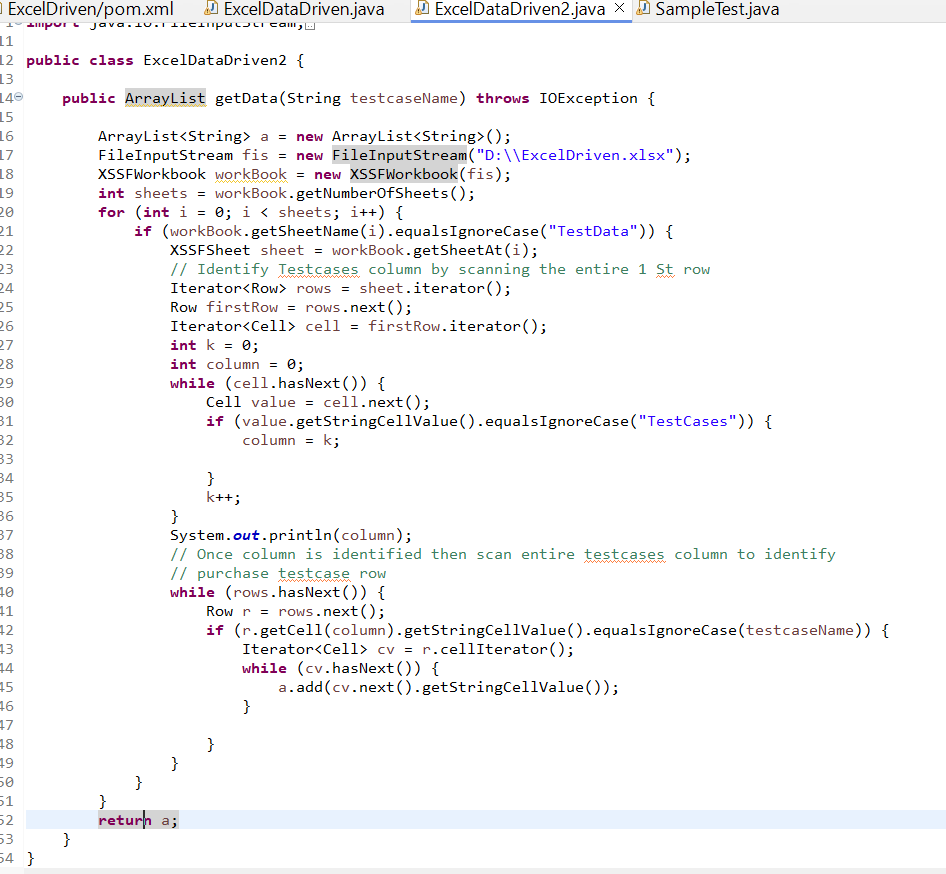
Note: //Identify Testcases column by scanning the entire 1 St row

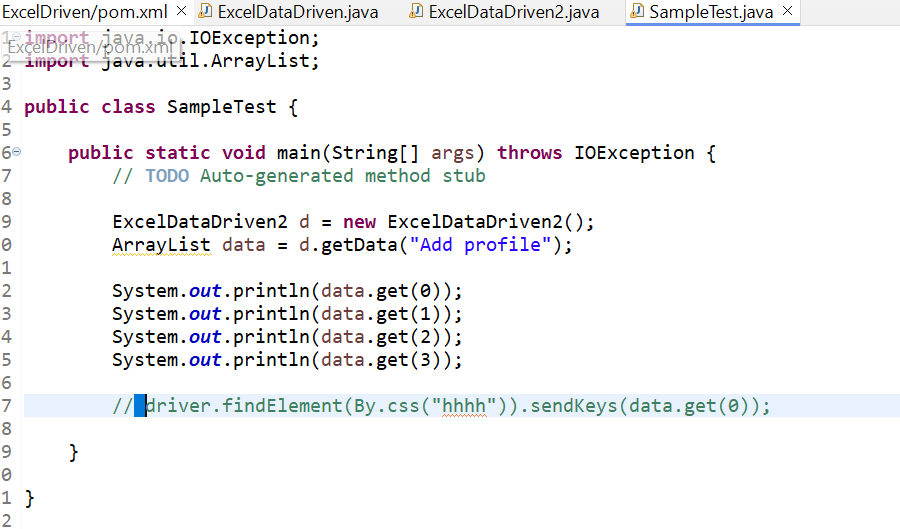
//Once column is identified then scan entire testcases column to identify purchase testcase row

//after you grab purchase testcase row pull all the data of that row and feed it into test

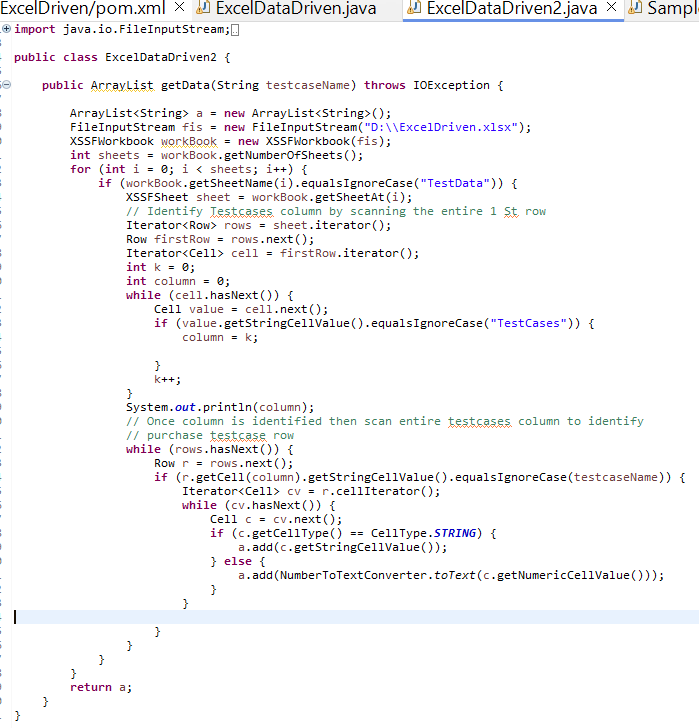
****

To store data in array:





In case of Numeric values in Excel sheet:



**Importance of Data Provider and Excel Integration for better Data driven**

**TestNG Data Provider + Excel Integration**

For each test data the TestNg data provider consider as different test but not incase of Excel because executes inside the method.

**3 data set**

**Data Provider-TestNG**

Result:

Test Case:3

Passed:3

Failure:0

**Excel**

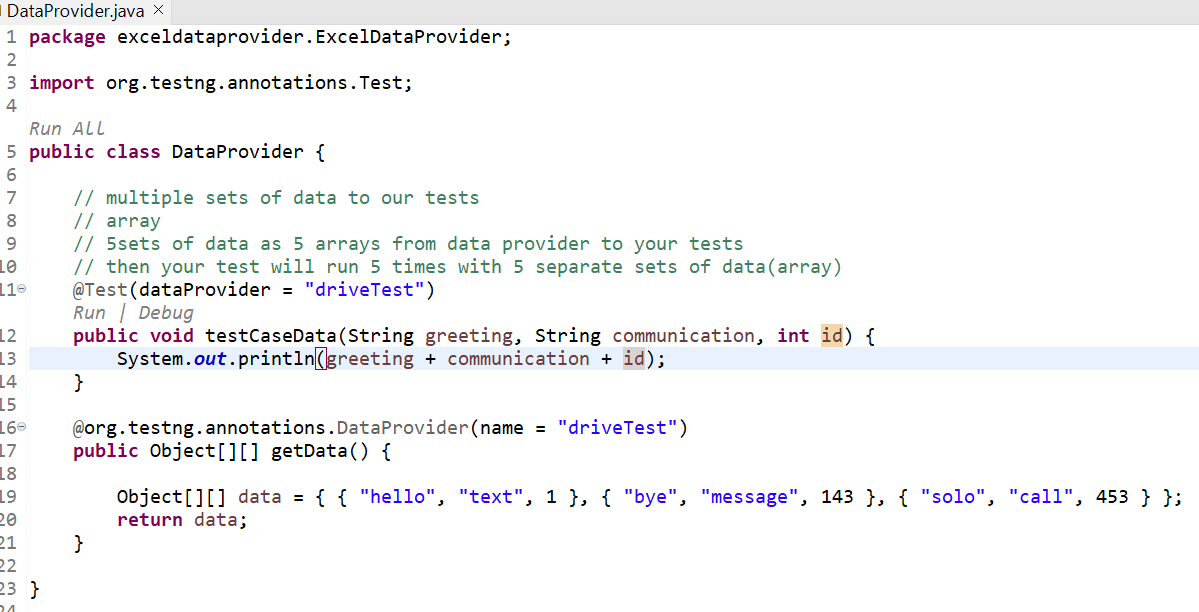
Result:

Test Case:3

Passed:3

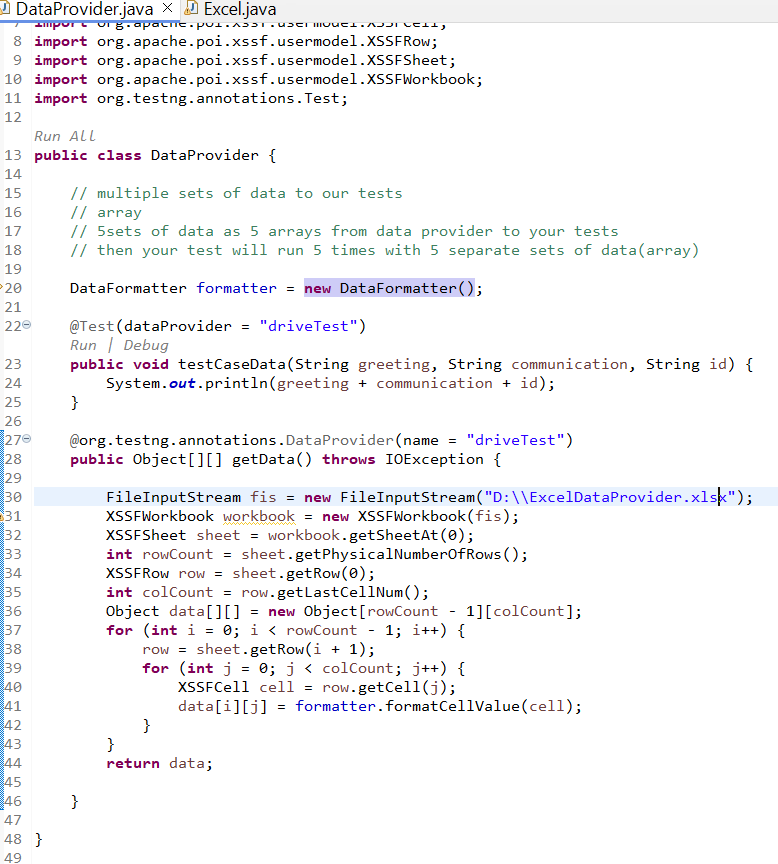
Failure:0

**Understand Dataprovider and how it sends data in Multidimensional array**



**Get Excel Dependencies and connect from java code to excel.**

**Live demo on integrating Excel to Dataprovider to parameterize data**

****

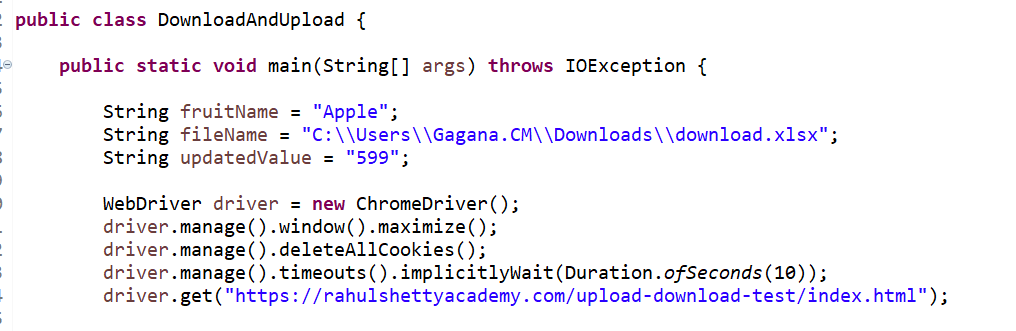
**Upload Download functionalities with Selenium using external excel files.**

**Download and uploading file using file attribute sendKeys with Selenium**

**Handle Synchronized toast messages with explicit wait for appear & disappear**

**Write Smart xpath to find table row column cell based on the conditions.**

**End to end solution for updating excel and uploading the file with validations.**

****

**Download Excel**

**Edit Excel**

**Upload Excel**

**Wait for success message and Wait to message to get disappear**

**Verify the result**

****

**Edit excel :**

Get rowNumber

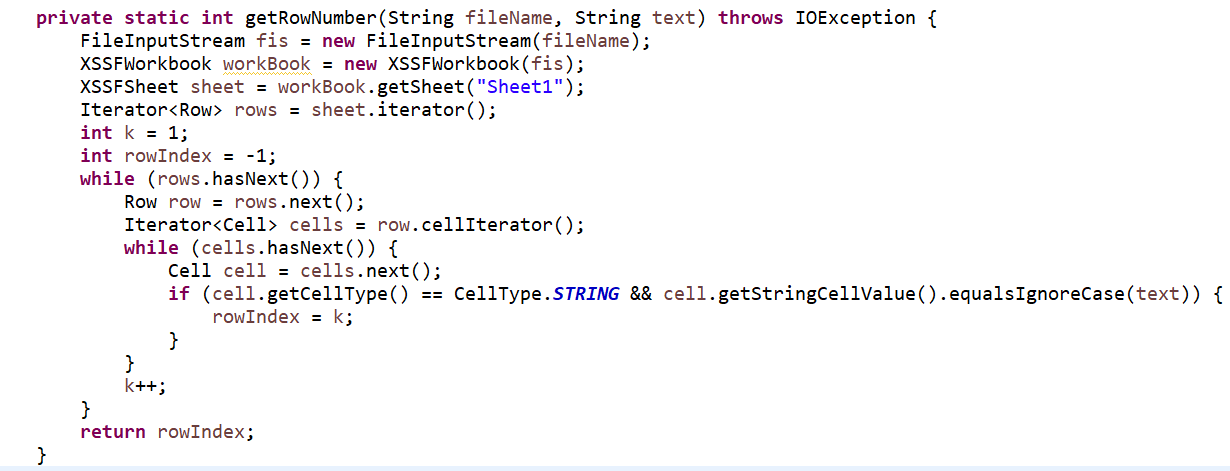
Get ColumnNumber

Update ****

**int** col = *getColumnNumber*(fileName, "price");

**int** row = *getRowNumber*(fileName, "Apple");

Assert.*assertTrue*(*updateCell*(fileName, row, col, updatedValue));



**A screen shot of a computer code

Description automatically generated**

**A screen shot of a computer code

Description automatically generated**

**Selenium 4 Chrome Dev tools Protocol (CDP) Integration Concepts**

**What are Chrome Dev tools? Why do we need this for Selenium testing?**

**What are ChromeDev Tools?**

Chrome DevTools is a set of web developer tools built directly into the Google Chrome Browser.

With Chrome DevTools, developers have deeper access to the applications which render on browser.

**What is Chrome DevTools Protocol (CDP)?**

The Chrome DevTools Protocol provide tools to instrument, inspect, debug and profile Chromium, Chrome and other Blink-based browsers.

Selenium 4 introduces powerful commands which are wrapper around the CDP Domains to grants access to Chrome DevTools directly from your automated tests.

With this CDP Support Selenium opens up the possibilities of out of Box Testing with the complete access and control to the browser features within the Test.

Examples:

* Capture, Monitor and stub the Network requests and responses.
* Inject Session Cookies and Perform basic Auth.
* Mock Device Coordinates for Mobile/Tabs view.
* Check and monitor the site’s performance.
* Mock geolocations of the user.
* Block the Network requests.
* Mock faster/solwer network speeds.
* Execute and debug JavaScript
* View console logs.

**Understand Device metrics override function to simulate browser as mobile**.

**Code Snippet to initialize ChromeDevTools connection with selenium.**

Chromium Driver Class has predefined methods to access the Dev Tools.

Chrome Driver and Edge Driver are inherited from Chromium Driver. So, we can access to DevTools with Chrome and Edge browsers.

**Step 1:**

Initiate Chromium Driver.

**Step 2:**

Create the object for Chrome Dev tools with getDevTools() method.

DevTools devTools = driver.getDevTools();

Returns the new DevTools object which allows you to send() the built-in selenium commands for CDP.

**Step 3:**

Initiate Dev Tools Session to send the commands from Selenium

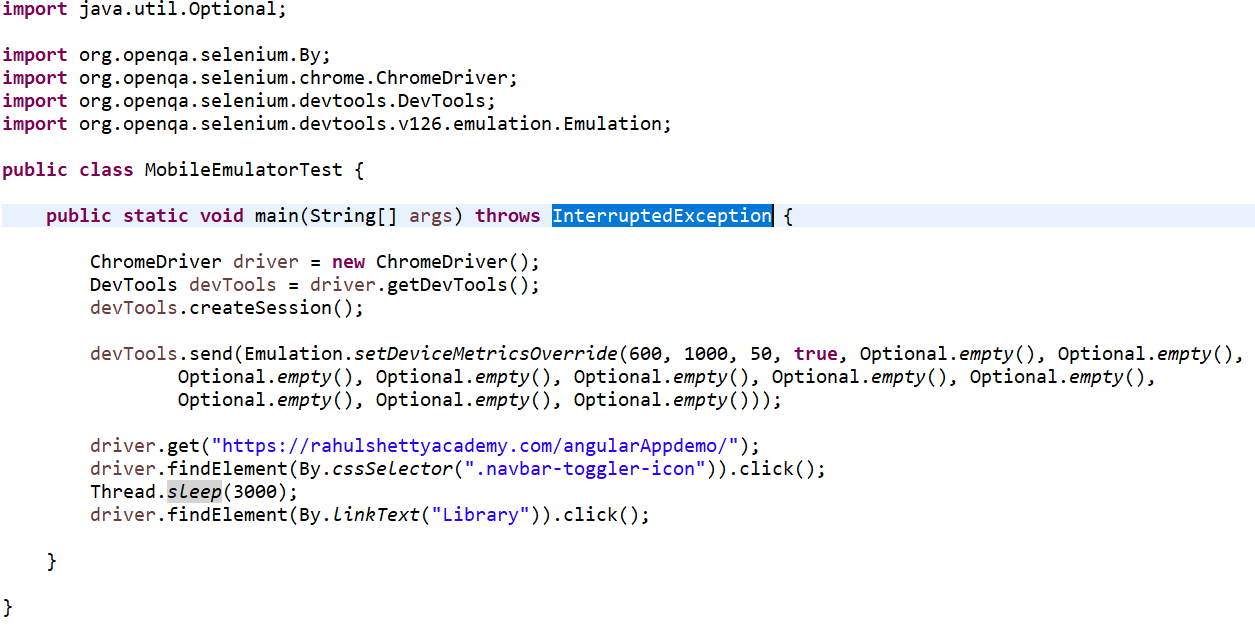
devTools.createSession()

Note: <https://chromedevtools.github.io/devtools-protocol/>

Under Emulation Domain

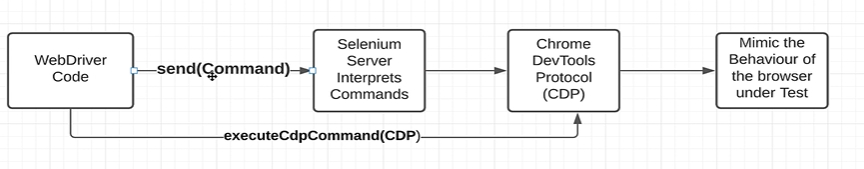
**Step 4:**

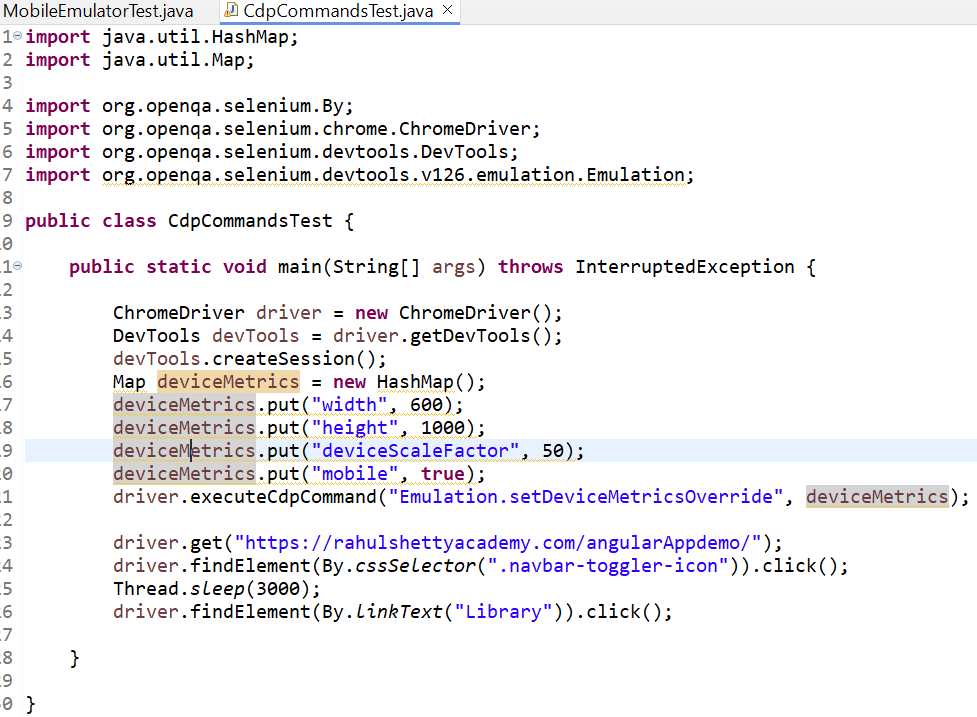
Create custom commands.



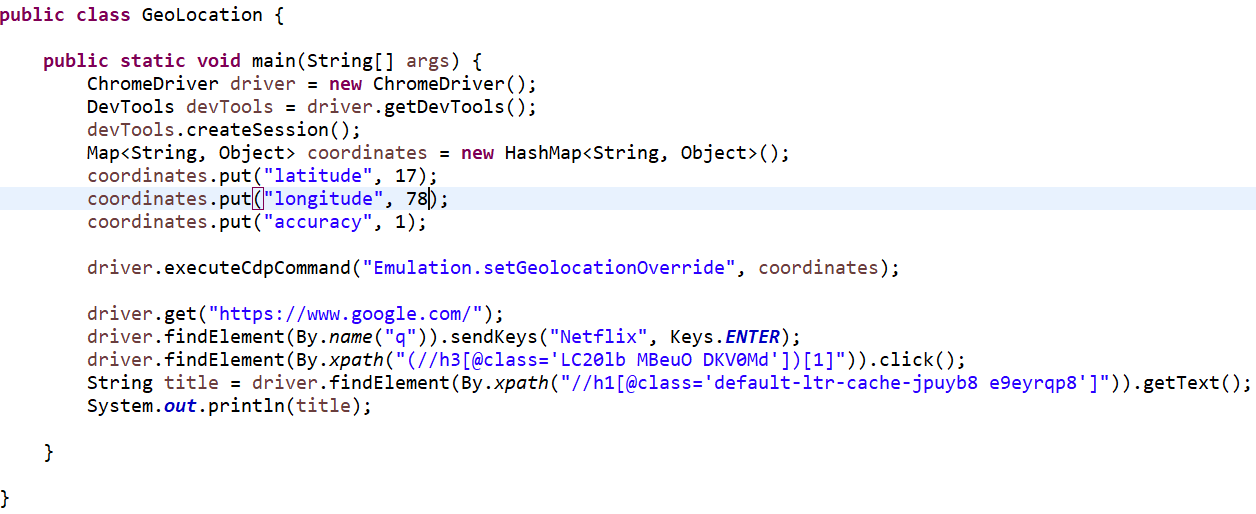
**Importance of executeCDP Command to construct the own CDP functions.**

Or instead of Selenium custom comands we can use executeCpdcommands (CDP)





**Localization Testing with Selenium 4 using ChromeDevTools Protocols**

****

**How to extract Network Responses and status codes with Selenium CDP Listeners**

**Domain: Network**

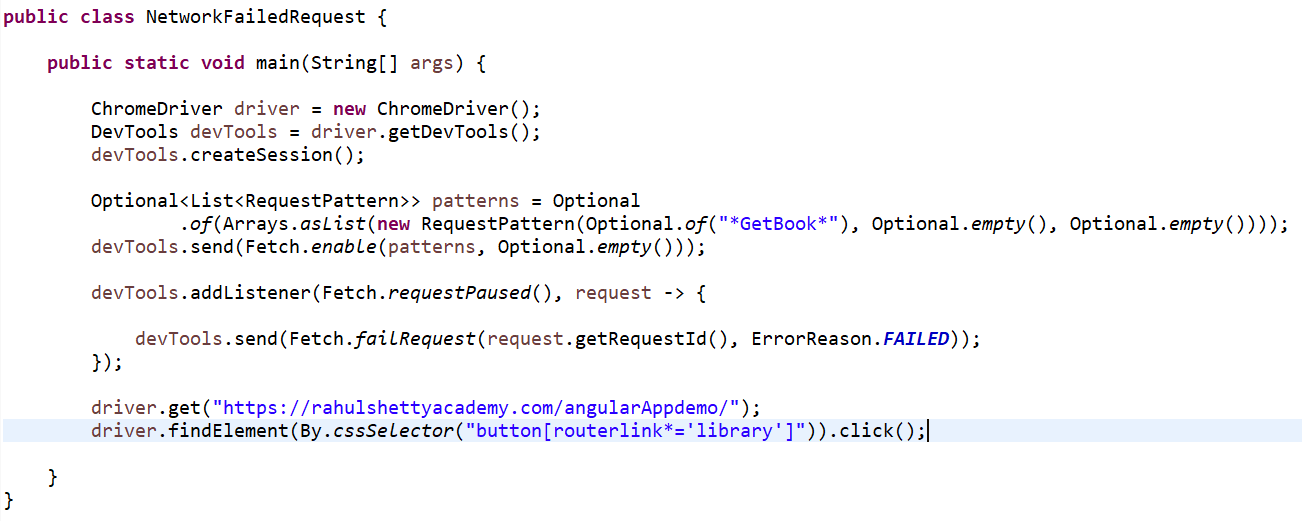
****

**Intercept Network /API Responses with Selenium Chrome dev tools**

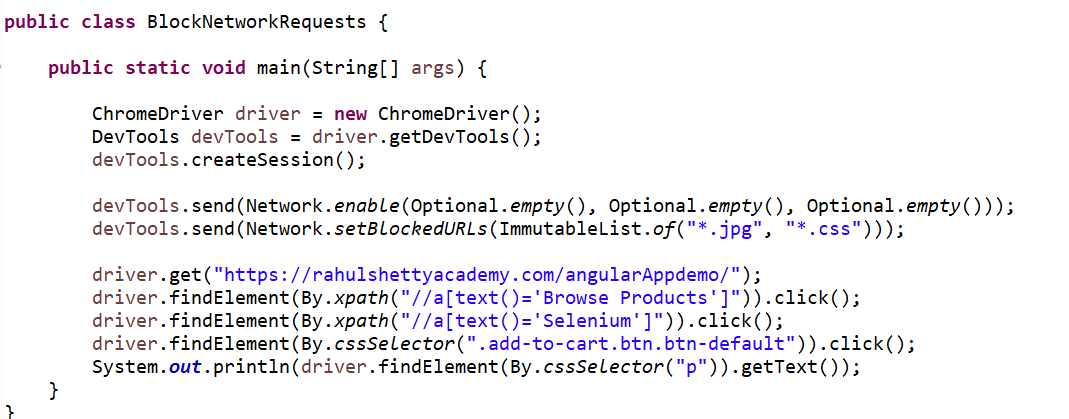
**Domain:Fetch**

****

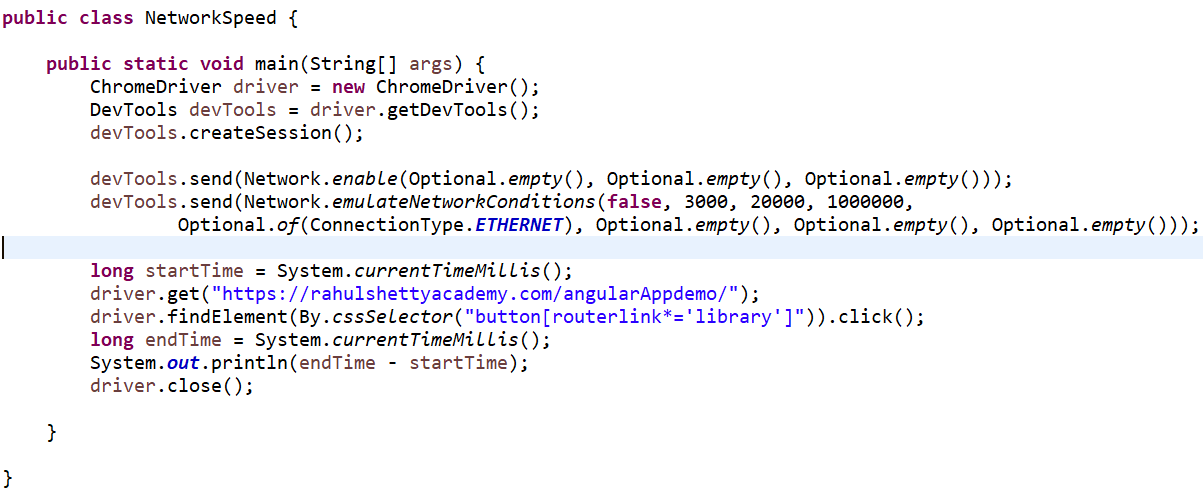
**How to test failed Network request calls with Selenium CDP Commands**

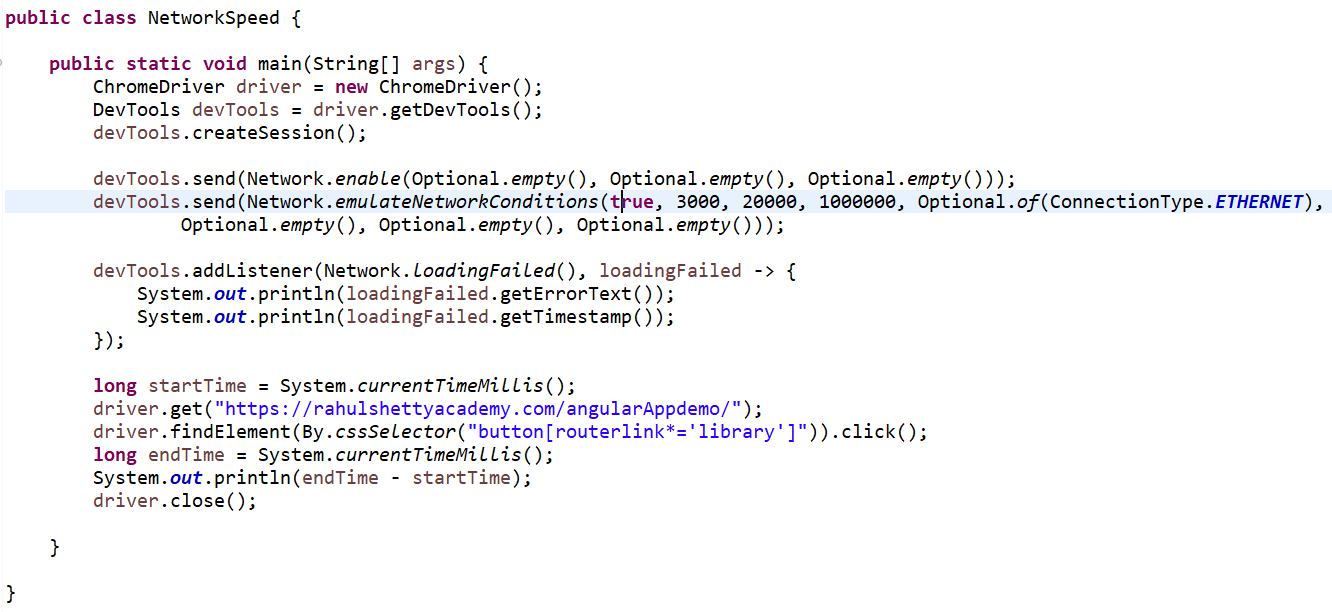
****

**Blocking unwanted Network request calls to speed up the execution with Selenium.**

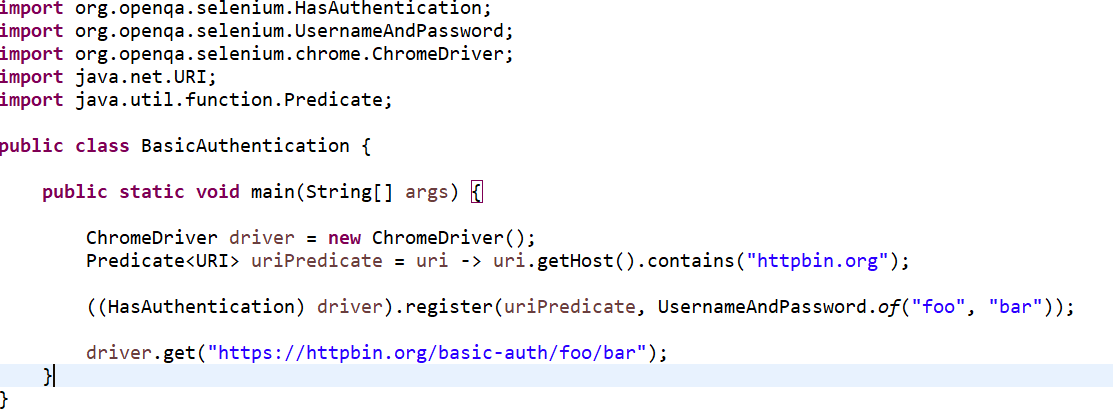
****

**How to emulate network speed with Selenium Chromedevtools Integration.**

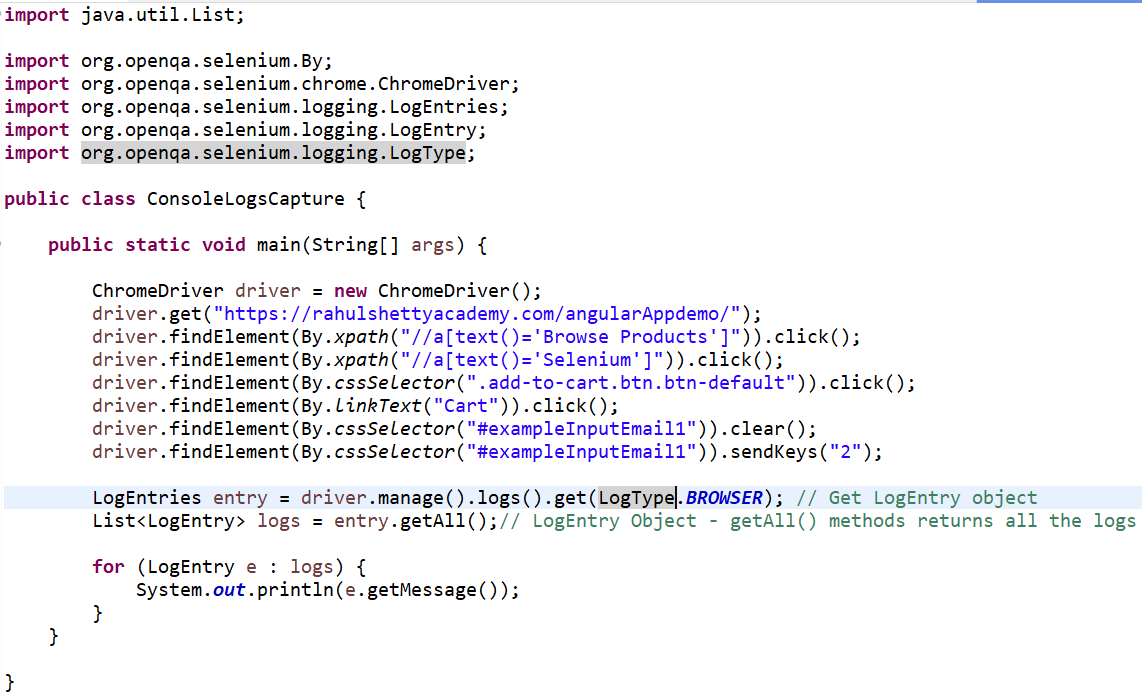
****

****

**Working with Basic Authentication using Selenium uriPredicate function.**

****

**How to log javascript errors from Selenium Script to console for debugging**

****

**DataBase connection to Selenium Testcases**

**Steps to connect Database to Selenium Testcases**

Selenium/Appium scripting with Database Validation

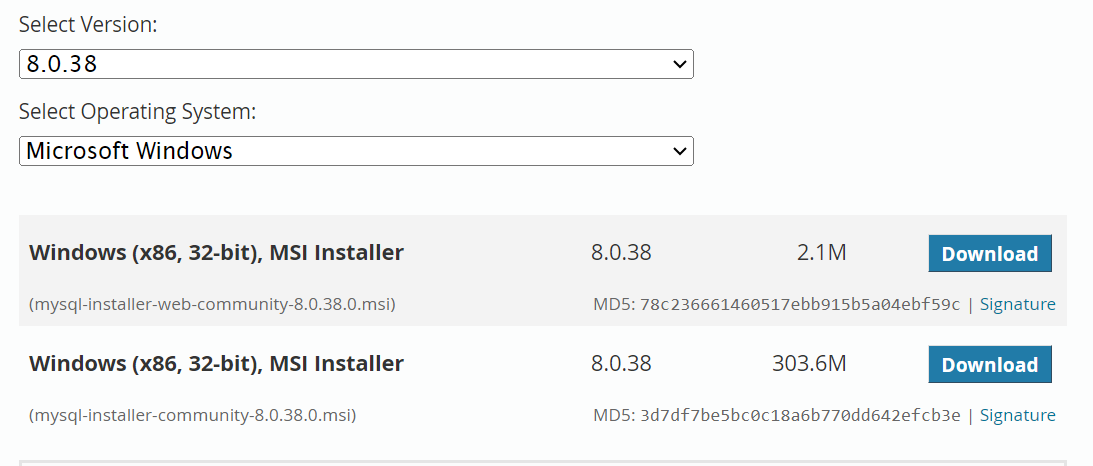
**Problem Explantion** – Client A🡪 Consultant B

Cerdit card Banking🡪 to verify the home page of 0 balanace account.

Zero Balanace account 🡪 password keeps on changed by Client B which is not aware by Consultant B, In order to over come this database connection to selenium is necessary.

**mysql download instructions /mySQL server connection procedure**

<https://dev.mysql.com/downloads/windows/installer/>



Download second one

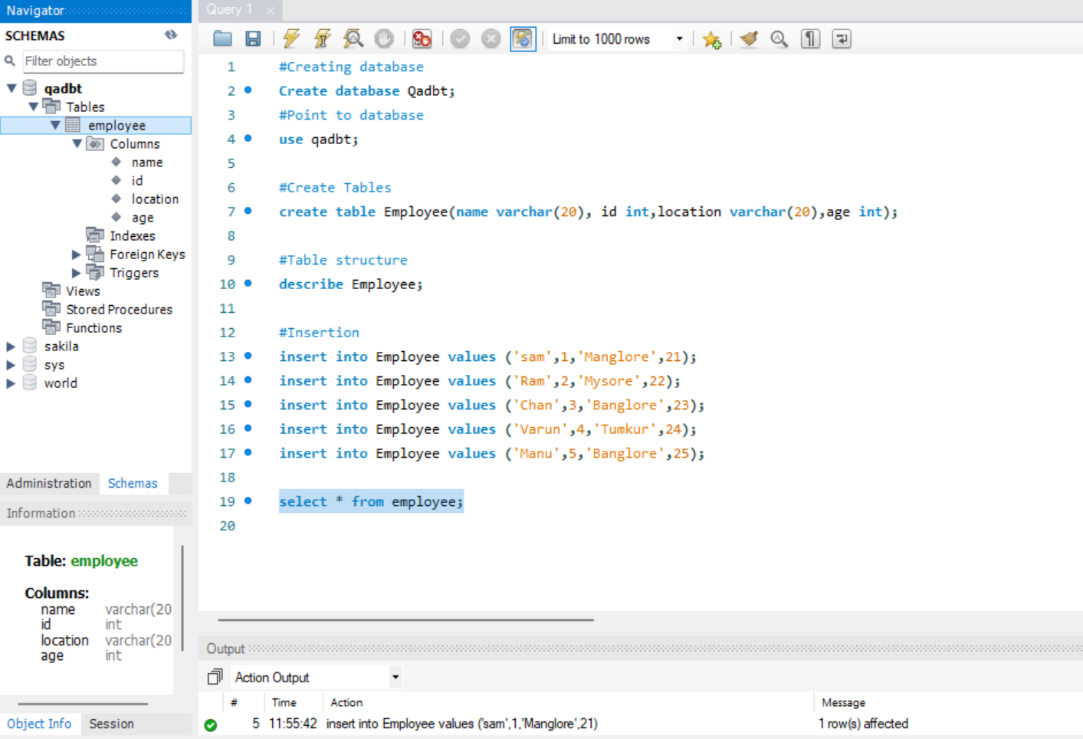
And set all the server and install the workbench.

Connect to the server by entering credentials.

**Creating Database in mysql server**

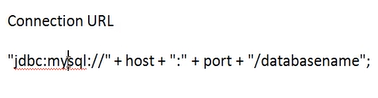
**Creating Tables in Databases**

**Inserting records into table**

****

**Integration of Database with JDBC API**

Jar 🡪 JDBCConnector jar file





**Steps to connect Database info to Selenium – 2**

****

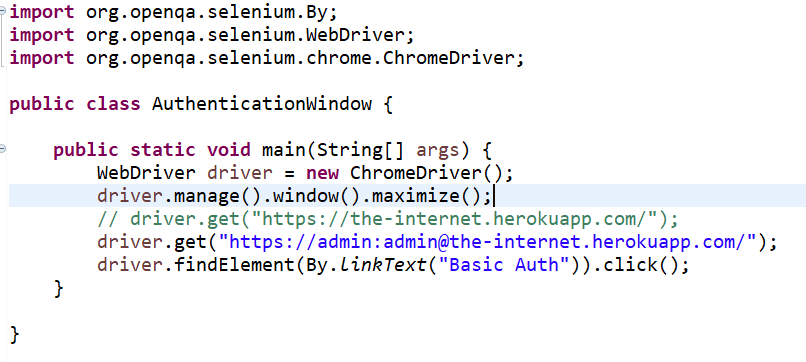
**File Uploading (AUTO IT) & Downloading with Selenium**

**Handling Window Authentication pop ups with Selenium**

* Handling Window Authentication Pop up.

http://Username:Password@SiteURL

**Examples on handling pop ups with modified webdriver url**

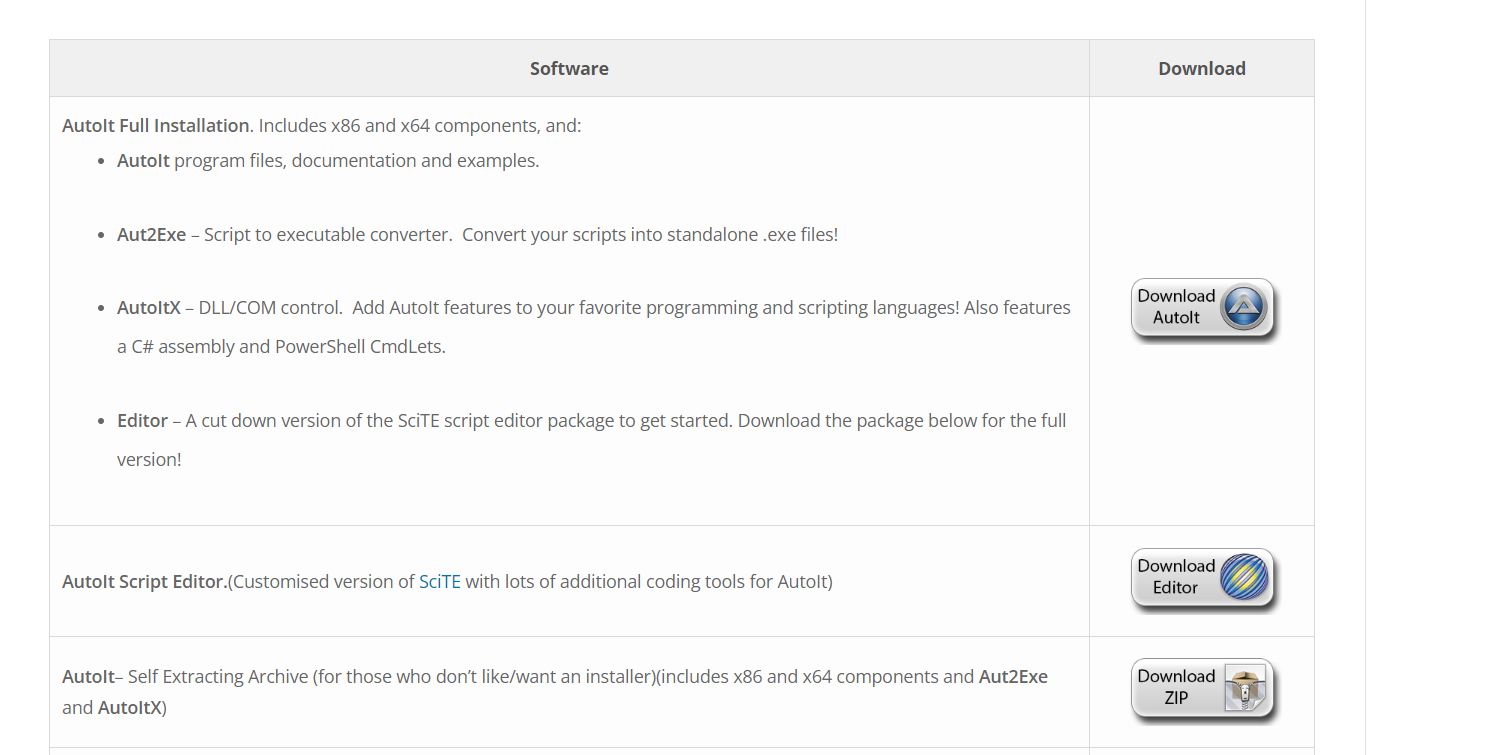


**What is AutoIT? Installation details**

Handling File upload from Windows using AutoIT.

Search : AutoIt Download

<https://www.autoitscript.com/site/autoit/downloads/>

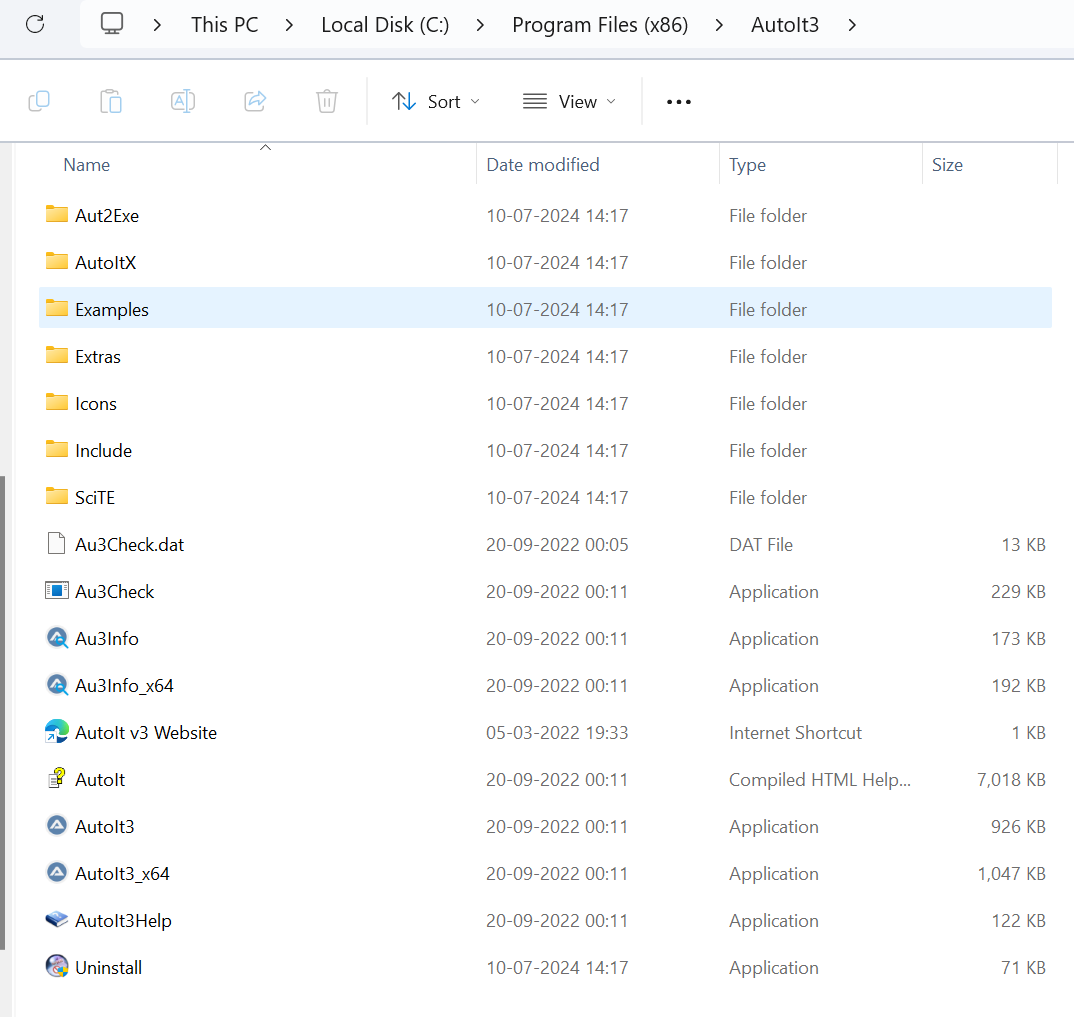


* Shift Focus to the file upload window.
* Set test/ path into file name edit box.
* Click open to upload file.

**Inspecting the window objects and converting into AutoIT code**

**End to End example on uploading File with AutoIT Selenium**

Open recording tool

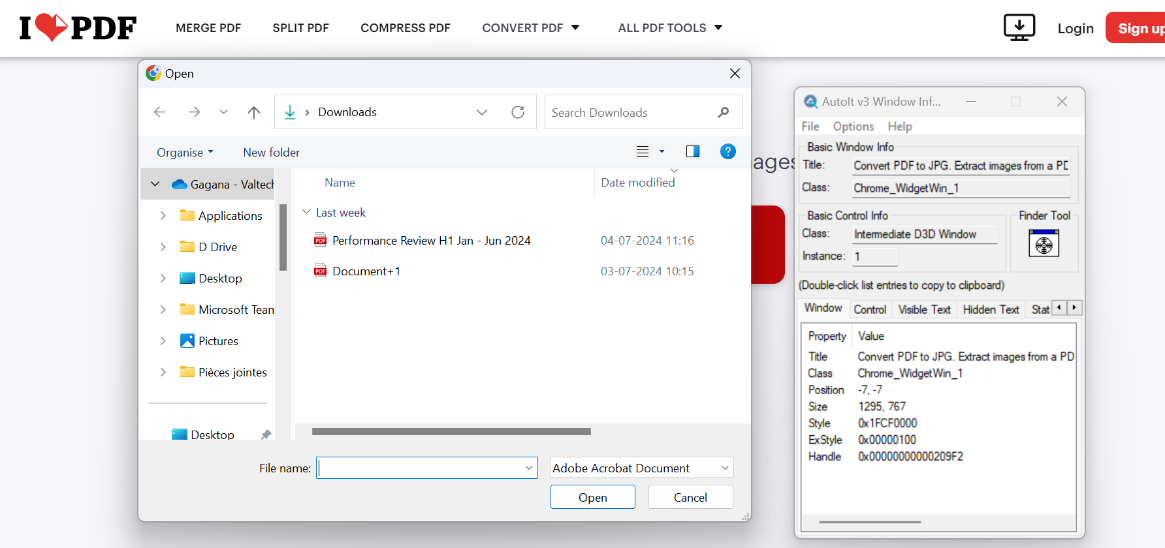


Open: Au3Info\_x64

A screenshot of a computer

Description automatically generated

Drag and drop the Finder Tool to file name edit box



Write scripts based on properties of Edit box

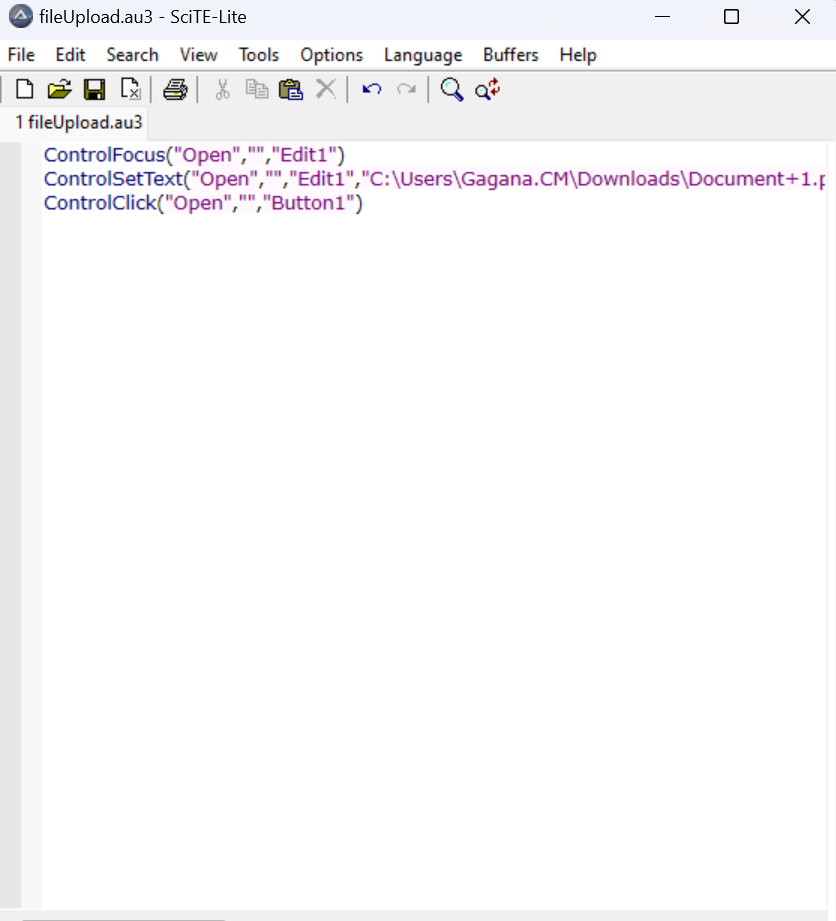
--Open **SciTE folder** under AutoIt3 folder.

--Open ScitE file to write configuration.

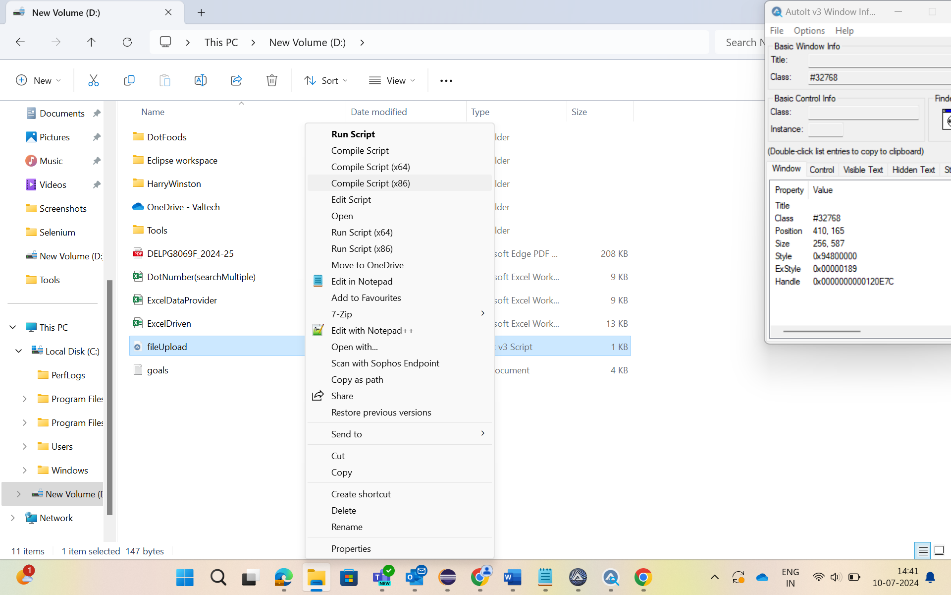
A screenshot of a computer

Description automatically generated

And write the script inside the editor



And save the file and compile it



Click on Compile Script (x86) 🡪 fileupload.exe file will be created 🡪 invoke that in selenium



* Au3info – record Window component objects
* Build script -scite.exe
* Save it - .au3 extenstion
* Convert file into .exe by compiling .au3 File
* Call .exec file with RunTime class in java into your selenium class

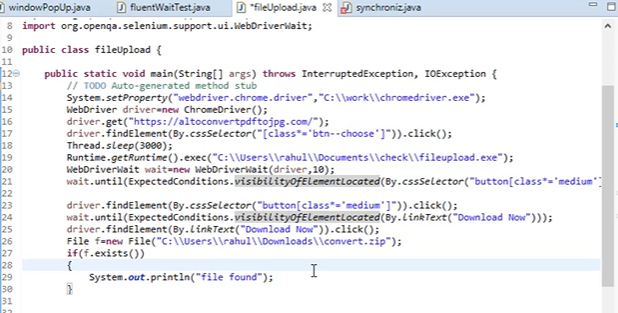
**Steps to complete the flow to download file from Application with Selenium**

Code to Download the file -selenium script.

Verify if the file is successfully downloaded.

Java File package.

Delete the file from your system.



**Chrome driver options to configure download path of browser**

import java.io.File;

import java.io.IOException;

import java.util.HashMap;

import org.testng.Assert;

import org.testng.annotations.Test;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

import org.openqa.selenium.chrome.ChromeOptions;

import org.openqa.selenium.support.ui.ExpectedConditions;

import org.openqa.selenium.support.ui.WebDriverWait;

public class fileUpload {

public static void main(String[] args) throws InterruptedException, IOException {

// TODO Auto-generated method stub

String downloadPath=System.getProperty("user.dir");

System.setProperty("webdriver.chrome.driver","C:\\work\\chromedriver.exe");

HashMap<String, Object> chromePrefs = new HashMap<String, Object>();

chromePrefs.put("profile.default\_content\_settings.popups", 0);

chromePrefs.put("download.default\_directory", downloadPath);

ChromeOptions options=new ChromeOptions();

options.setExperimentalOption("prefs", chromePrefs);

WebDriver driver=new ChromeDriver(options);

driver.get("https://altoconvertpdftojpg.com/");

driver.findElement(By.cssSelector("[class\*='btn--choose']")).click();

Thread.sleep(3000);

Runtime.getRuntime().exec("C:\\Users\\rahul\\Documents\\check\\fileupload.exe");

WebDriverWait wait=new WebDriverWait(driver,10);

wait.until(ExpectedConditions.visibilityOfElementLocated(By.cssSelector("button[class\*='medium']")));

driver.findElement(By.cssSelector("button[class\*='medium']")).click();

wait.until(ExpectedConditions.visibilityOfElementLocated(By.linkText("Download Now")));

driver.findElement(By.linkText("Download Now")).click();

Thread.sleep(5000);

File f=new File(downloadPath+"/converted.zip");

if(f.exists())

{

Assert.assertTrue(f.exists());

if(f.delete())

System.out.println("file deleted");

}

}

}

**ExtentSpark Report and Extent Spark Report PDF For Cucumber + TestNG**

**Dependency:**

<dependency>

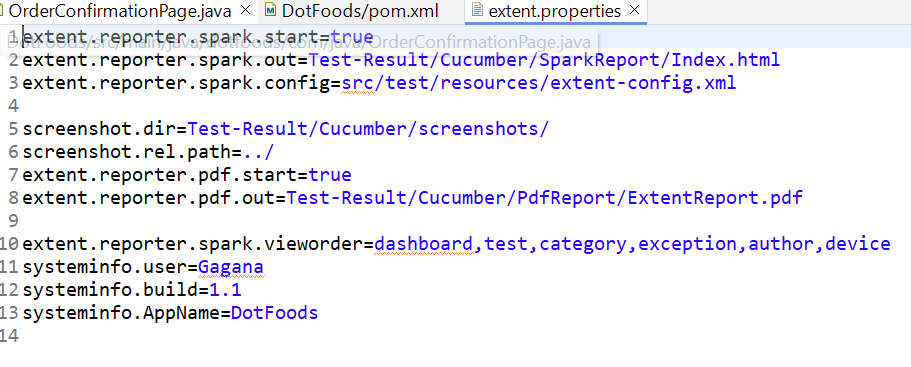
<groupId>tech.grasshopper</groupId>

<artifactId>extentreports-cucumber7-adapter</artifactId>

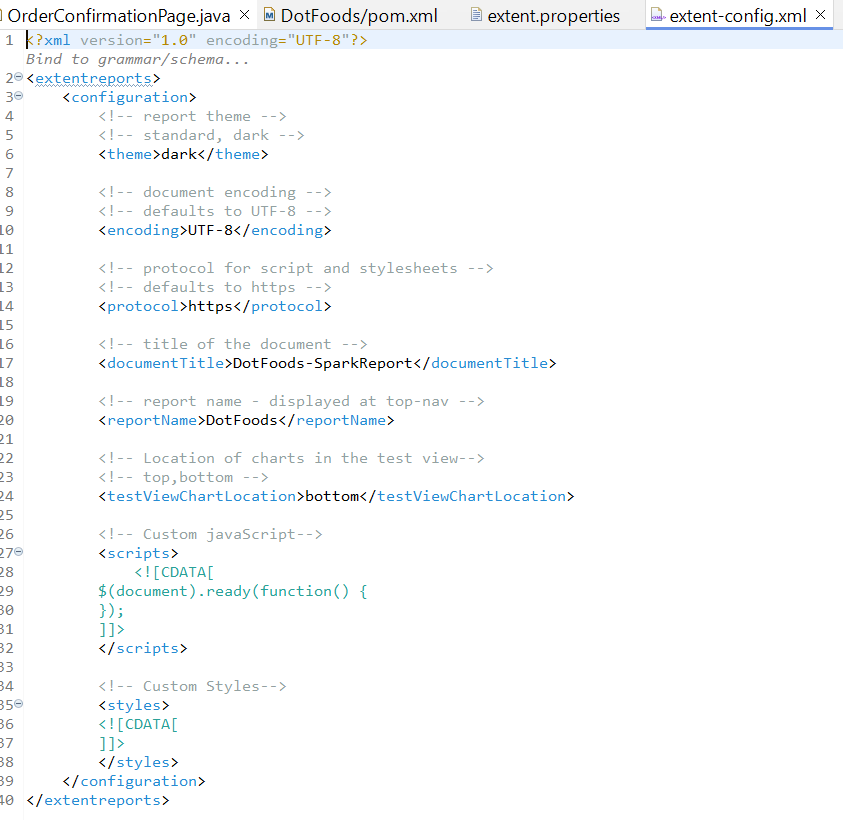
<version>1.14.0</version>

</dependency>

**Extent.Properties File:**



**Extent-congig.xml file:**

****

**TestNgRunner class:**

**A computer screen shot of a code

Description automatically generated**

**Allure Report**

Allure Report is a flexible multi-language test report to show you a detailed representation of what has been tested and extract maximum of useful information from everyday execution pf tests.

1. **Maven Dependency**

TestNg, Selenium Java, WebDriver manager, Allure-TestNG

WebDriver manager is optional.

1. **Download Allure binaries(Zip) and set the path :**

https://github.com/allure-framework/allure2/releases

**Alternatively, allure can also be installed from Powershell using below commands,**

Iex(new-object net.webclient).downloadstring(‘https://get.scoop.sh’)

Scoop install allure

1. **Run Tests using below commands(Ensure maven is installed in your system)**

mvn clean test

1. **Now Let us run below command to generate allure reports**

allure serve

**Note:**

* Run the test case
* Refresh the project
* Open Cmd upto pom.xml of Project
* Run commad – allure serve

**Logger Implementation using Log4j2:**

**Dependencies:**

<dependency>

<groupId>org.apache.logging.log4j</groupId>

<artifactId>log4j-core</artifactId>

<version>2.23.1</version>

</dependency>

<dependency>

<groupId>org.apache.logging.log4j</groupId>

<artifactId>log4j-api</artifactId>

<version>2.23.1</version>

</dependency>

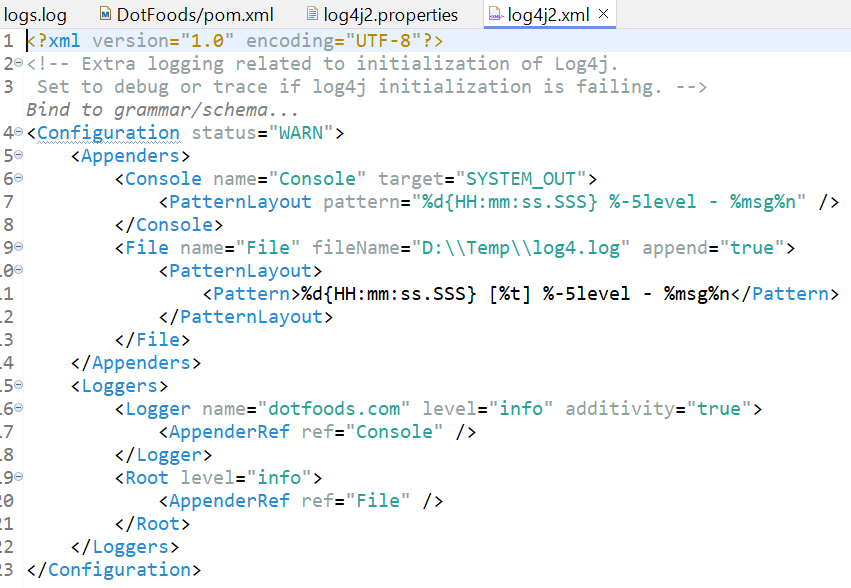
**We can implement using Xml file or properties file :**

**Log4j2.properties**

**A screenshot of a computer program

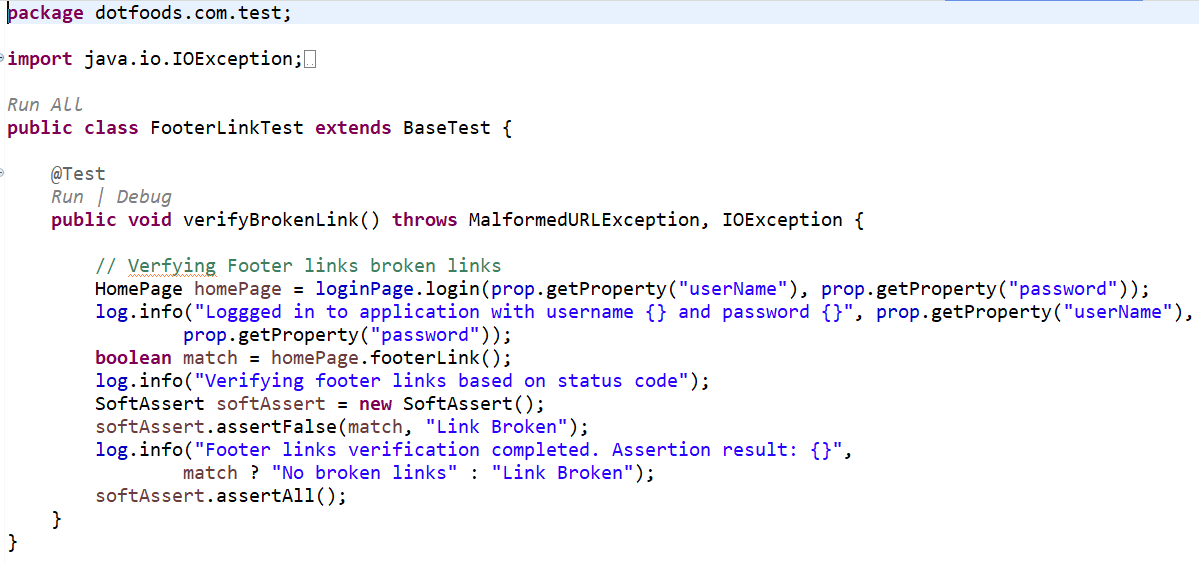
Description automatically generated**

**Log4j2.xml**

****

**A computer screen shot of a computer code

Description automatically generated**

****