Stage 1 - Java Programming

Stage 2 - Web automation (Selenium) & Windows GUI (Auto IT)

Stage 3 - Framework 1 (TestNG, Data Driven Framework, Page Object Model, Keyword Driven Framework)

Stage 4 - Framework 2 - BDD Framework

Stage 5 - Git & Jenkins

java-concept - <https://github.com/balaji-githubstore/java-concept-equiniti-jun-2025.git>

selenium-concept - <https://github.com/balaji-githubstore/selenium-java-concept-equiniti-jun-2025.git>

Framework 1 - Hybrid Framework- <https://github.com/balaji-githubstore/java-hybrid-framework-equiniti-jun-2025.git>

Branch

master - latestcode

b1-no-page-object -- project without page object model

b2-no-base-page-class -- WebDriverKeywords class not implemented here

Framework 2 - BDD Framework - <https://github.com/balaji-githubstore/java-bdd-framework-equiniti-jun-2025.git>

master - latestcode

b1-no-automation -- BDD Framework without automation

—-------------

Selenium - https://www.selenium.dev/

* Web automation
* Open Source
* Language Independence - Java, C#, Python, Javascript, Ruby

Selenium - A suite of tools

1. Selenium IDE
   1. No programming knowledge is required
   2. Plugin - chrome, edge, firefox
   3. Can use it only for internal or exploratory testing
2. Selenium RC - Depreciated
   1. Programming knowledge is required
   2. Architecture

Source code (Java+Selenium RC)--> RC server (Turn ON/OFF) → Browser

1. Selenium WebDriver
   1. Programming knowledge is required
   2. Architecture

Source code (Java+Selenium WebDriver)→ Browser

1. Selenium Grid
   1. If you want to scale by distributing and running tests on several machines, and manage multiple environments from a central point.

### Java Programming

1. Installation
   1. JDK 11 and above

<https://www.oracle.com/java/technologies/javase/jdk17-archive-downloads.html>

* 1. IDE
     1. Eclipse
     2. IntelliJ

1. Architecture

Source code (.java) → Byte code (.class) → O/P

Byte code - Platform-independent code

JVM (Java Virtual machine) → Helps the Operating system to understand the bytecode

Compile time → source code to byte code

Runtime → byte code to O/P

1. UpperCamelCase - MyFirstProject

lowerCamelCase - myFirstProject

1. Eclipse Structure

Workspace -

Project 1 - UpperCamelCase

Package - lowercase (com.companyname.purpose)

Class - UpperCamelCase

Methods and variables - lowerCamelCase

1. Datatypes
   1. Pre-defined
   2. Non-pre-defined - collection of pre-defined
      1. String
      2. Array
         1. Size is fixed
         2. Datatypes of fixed
      3. User defined datatypes
2. Whenever you give whole number, default it considered as int

Whenever you give decimal number, default it considered as double

1. Debugging
   1. Continue
   2. Terminate
   3. Step into
   4. Step over
2. Methods - Building block of the program

* Reusablity
* Maintenance
  1. Static methods
     1. How to create and call it?

//accessmodifier static returntype methodName(arguments)

**To call the static method**

**classname.methodname()**

* 1. Non-static methods
     1. How to create it and call it?
     2. To call non-static method
        1. Create object
        2. Using objref.methodName()

1. Class & Object
2. Access modifier
   1. Private – accessible within the class
   2. Default - accessible within the package
   3. Protected - accessible within the package and also outside the package through inheritance
   4. Public - accessible anywhere
3. Object
   1. Declaration
   2. Instantiation
   3. Initialization
4. Variable
   1. Static variable
   2. Non-static variable
5. Class
   1. A class is a template, blueprint, or type from which objects are created.
6. Object
   1. Object is an instance of a class
   2. Every object has its own state (non-static variable) and behaviour (non-static methods)
7. this keyword
   1. Helps to distinguish between instance and local variable
   2. this → points to the current object
8. Constructor - prerequisite to the object
   1. Constructor name and classname should be same
   2. It is a kind of method without any return type.
   3. It gets called whenever objects are created.
   4. There will always a default constructor that helps to load all the non-static variable with default values
   5. We can override the default constructor, by creating our own constructor
   6. We can create a constructor explicitly, but you need to call it.
      1. Without arguments
      2. With arguments
9. Constructor overloading

The Constructor to be called is resolved during compile time.

By a change in

1. Number of arguments
2. Datatypes of arguments
3. Sequence of arguments
4. Method overloading/ Compile time polymorphism/early binding/static polymorphism

The method to be called is resolved during compile time.

Can create multiple methods with the same name. By a change in

1. Number of arguments
2. Datatypes of arguments
3. Sequence of arguments

19. Collections

* Non-generic type
* Generic type
  + ArrayList
  + HashMap

20. Inheritance

* Reusablity
* When we need to reuse methods and variables, we can use the inheritance concepts

### Selenium WebDriver Project

1. Create a Java Project
2. Add the selenium jar
3. Navigate to url
4. Gettitle, getcurrenturl, getPageSource
5. Click, type, select
6. To inspect → tagname, attribute, text or not
7. Basic Locators
   1. Id
   2. Name
   3. className
   4. tagName
   5. linkText
   6. partialLinkText

When there are duplicate locators. then findElement picks the first webelement

1. Advance Locator
   1. Xpath
   2. Css
2. Dropdown
   1. With the select tag
      1. selectByVisibleText(string)
      2. selectByValue(string)
      3. selectByIndex(int)
   2. Without the select tag
      1. click()
3. Synchronization
   1. Unconditional wait (java lib)

Thread.*sleep*(8000); → wait for 8s but not recommended

* 1. Conditional wait (selenium lib)
     1. Implicit wait
        1. Default implicit wait - 0s
        2. Applicable to all findElement and findElements methods.
        3. Example: Implicit wait - 30s
           1. If the element is not present, then it will check for 30s and then throw an exception
           2. If the element is present, it will move on to the next step immediately.
           3. Polling time - 0.5s (how freq it checks for the element)
     2. Explicit wait
        1. Exact condition
        2. Polling time - 0.5s
     3. Fluent wait
        1. Customize the polling time
        2. Ignore exception

1. Click() → The element should be present and visible.
2. Multiple tabs/windows, Frame, Alert → switchTo()
3. Multiple tabs/windows
   1. driver.getWindowHandles() → gets all session id
4. close vs quit
   1. Close → close the current session/tab
   2. Quit → close the current browser/all session and also it kills the process associated with it.
5. Frame
   1. Even though locator is correct, we get .NoSuchElementException:
   2. Check for tagname - frame or iframe
   3. Switch to frame (anyone option)
      1. Using Name
      2. Using index
      3. Using WebElement
6. Alert
   1. Javascript alerts
7. Upload the file
8. Actions - mouse/keyboards
   1. May not throw proper error
   2. May not work in headless mode
   3. Do not disturb the mouse/keyword

Modifier keys - ctrl, shift, alt - keyDown() & keyUp()

Rest of the keys - sendKeys()

1. CSS selector
2. Table
3. Javascript
   1. Click on hidden elements
   2. Type on read-only textbox
   3. Scroll tothe element
   4. Scroll page

Option 1 - click & type using javascript

document.querySelector('#authUser').click()

document.querySelector('#authUser').value='jack'

document.querySelector('#authUser').scrollIntoView()

Option 2 - click and type using javascript and webelement

JavascriptExecutor js=(JavascriptExecutor) driver;

WebElement ele1=driver.findElement(By.*xpath*("//input[@name='dateDeparture']"));

js.executeScript("arguments[0].value='20 Jul 2025'",ele1);

WebElement ele2=driver.findElement(By.*xpath*("//input[@name='dateReturn']"));

js.executeScript("arguments[0].value='30 Jul 2025'",ele2);

1. Chromeoptions
   1. Headless mode
   2. Change download dir
   3. Disable notifications
   4. SSL certificate

—---------------------------------------

### Git - Git is a free and open-source distributed version control system

Architecture

Project (on local machine) → Local repository (on local machine) → remote repository (GitHub, Bitbucket, AWS Code commit)

Git

modified - stage - commit

Steps to push to the remote repo

git init → initialize local repo or existing local repo

git add . → staging

git commit -m "first commit" → update the local repo

git remote add origin <https://github.com/balaji-githubstore/java-hybrid-framework-equiniti-jun-2025.git> - register the remote url with name origin

git push -u origin master - update the remote repo

—-----------------------------------------

### Framework 1 - Hybrid Framework

1. Unit Test Framework - TestNG
2. Data Driven Framework - Separating the test method and test data (excel, csv, json)
3. Page Object Model - Manage the web elements efficiently

Keyword Driven Framework - Create reusable webdriver keywords

Build Management Tool - Maven

pom.xml → Project Object Model

* Manage the jars and also the dependent jars for the project once the dependency is configured.
* Can control the compile version easily
* Can control the project in the command line using goals
* Using the goals pipeline can be configured easily
* Can migrate the project easily to any platform

Package

com.eq.test → TestClass and test methods

com.eq.base → Browser and report configuration

com.eq.utilities → Excel, JSON, CSV, properties file reusable code

com.eq.pages → Page class and Page methods

Steps to create a framework

1. Create a Maven project
   1. Add group ID
   2. Add artifact ID
2. Configure the jars required for the project
   1. TestNG - https://mvnrepository.com/artifact/org.testng/testng/7.11.0
   2. Selenium - https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java/4.33.0
3. Create Test package, class and test methods (@Test)
4. Every @Test method should have at least one assertion. Assertion decides whether @Test is pass or fail.
5. Annotations
   1. @BeforeMethod - runs before each @Test method
   2. @AfterMethod - runs after each @Test (even though @Test fails)
6. TestNG generates two reports
   1. Emailable report
   2. Consolidated report - index.html
7. Use @Priority to order the @Test method. @Test method without priority will be given higher preferences
8. Invocation count - @Test(priority = 2,invocationCount = 2)
9. Add browser configuration to the base class - AutomationWrapper
10. DataDriven Activity - @DataProvider
    1. Create a @Test method with proper arguments
    2. Create a method that returns two two-dimensional arrays and also provide @DataProvider annotation
    3. Connect @Test with @DataProvider
11. Read Excel
    1. Add dependencies
       1. poi - https://mvnrepository.com/artifact/org.apache.poi/poi/5.4.1
       2. poi-ooxml - https://mvnrepository.com/artifact/org.apache.poi/poi-ooxml/5.4.1

To read

1. Location - Read/Write
2. Open
3. Sheet
4. Row
5. Cell

.xlsx → XSSFWorkbook

.xls → HSSFWork

1. Connect Excel with @DataProvider
2. Page Object Model - design pattern
   1. Reusablity
   2. Maintenance
   3. Readability

Steps to implement page object model

1. For each web page in the application, we need to create a class - Page class
2. Operations happen through the methods - Page Method
3. Collecting the object repository (locators) at the class level or in different file.
4. Add WebDriverKeywords
5. TestNG XML - Suite file
   1. Parameterization
   2. Control the methods/class to run
   3. Groups
   4. Parallel execution
      1. <test>
      2. <class>

### Framework 2 - BDD

BDD - Behaviour Driven Development - Understand the requirement

ATDD (Modified BDD) - Acceptance Test Driven Development - writing the acceptance testing

1. Tools to work on BDD

* Java and JavaScript - Cucumber
* C# - Specflow and ReqnRoll
* Python - Jbehave

1. Architecture of cucumber

Feature file (.feature) → Step definition (.java)

Steps to create a BDD Framework

1. Create a Maven project
   1. Add group ID
   2. Add artifact ID
2. Configure the jars required for the project
   1. Cucumber-java
   2. Cucumber testng
   3. Selenium - <https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java/4.33.0>
3. Create Feature
   1. Provide feature header
   2. Provide a feature description
      1. Unformatted description
      2. Formatted description

In order to [business goal]

As a [role]

I would like to [visible change in the application]

1. Create Scenario
   1. Add Scenario title
   2. Add Steps
      1. Minimum one given, when, then

Given - Pre-requisite of that scenario

When - Actions/ operation

Then - Assertions

1. Create TestNG runner and configure the feature to run in [com.eq](http://com.eq).runner package
2. Added the package - [com.eq](http://com.eq).steps package
3. Step parameterization
   1. Code Reuse
   2. It reduces duplicate code in the step definition.
4. Scenario outline
   1. One scenario - multiple sets of test data
   2. Reduce duplicates inthe feature file
5. Background
   1. If all scenario in a single feature file follows the same given then we can implement the background; otherwise, ignore it.
6. Cucumber Options
   1. dryRun - helps to find missing step defn
   2. Publish - publish the report to public github
   3. Plugin - generate reports locally
   4. Tags - and, or, not
      1. tags = "@login and (not @invalid)"
      2. <https://cucumber.io/docs/cucumber/api/#tag-expressions>
7. Datatable
8. Datatable with Scenario outline
9. Hooks
   1. @Before - runs before each scenario
   2. @After - runs after each scenario, even scenario fails

Important concepts - Improve maintenance and readability

1. Step parameterization - code resue
2. Scenario outline - one scenario - multiple set of test data
3. Background - repeated given
4. Datatable - tabular data to the single step defn
5. Login to jenkins
6. Configure maven project
   1. Create a free style project
   2. Add SCM
   3. Triggers
   4. Build steps

1 GB - 1027 MB

1 MB - 1027 KB

1 KB - 1027 B

1 B - 8 bits

Exceptions

1. NoSuchDriverExceptions
2. NoSuchElementExceptions
3. ElementClickInterceptedException: - Some other elements hides the target element (like popup)
4. ElementNotIteractableException→ Element is present but not visible
5. org.openqa.selenium.NoSuchWindowException: no such window: target window already closed
6. org.openqa.selenium.NoAlertPresentException: no such alert