==========

Accenture QA Automation interview real time que.

- **Can you brief me about yourself?**
  Hi, my name is Pankaj.

  I started my career as a Testing Executive 4.5 years back with Infosys currently I am working as Test Engineer.
  My responsibility is to understand Business Requirement Specification and High-Level scenarios and to convert them into test cases & Automation scripts if required.
  Execution of test cases and reporting of defect to the developer if there any and get them fixed.
  I have experience on Functional, Automation, Regression, Smoke, Sanity, Web accessibility, Web Analytics, Mobile Testing.

  In my previous project I have worked on Automation testing where we have used Selenium with java and TestNG Cucumber framework for BDD approach. We have used Page object model where we have separated our test cases with page objects, and we performed testing on the same. For build management tool we are using Maven for version controlling we are using Git and for automating our jobs for nightly run or any schedule we are using Jenkins,.

  For defect management & test case management we have used JIRA, TEST RAIL & HP ALM.
  I have worked on tools like BrowseStack, DeviceAnywhere, Toadsql,

  I am working on Agile environment we have daily standup call and we have 2-week sprint cycle. I am part of 8-member team out of which we are 3-Tester, 2- dev, 1- manager, 1-scrum master.

- Tell me your Day to Day activities as QA?

  First thing I do after login in my system. I check the active sprint in Jira for our project code. There I can see my assigned open tasks. After that I will check my mail if there is any important mail I need to take action on. Then we have our daily scrum meeting where we used to tell our previous day actions what we did, what we are planning for today and if we have any blocker to discuss. Product owner and scrum master help us to resolve that blocker. After that I need to take the pending task and do needed action whether creating test case, Execution, Defect retesting if any.

- **Do you have created framework from scratch, or you have maintained that?**
  I have not created Framework from scratch by myself but yes, I was part of framework creation and created some part of it.

- **How much you rate yourself in Java out of 10?**
  Out of 10 I will rate myself 6 in java as QA Automation engineer.

- **Can you tell me Oops concepts and relate it with your Framework?**
  We have Polymorphism, Inheritance, Encapsulation and Abstraction in Oops. So, we will start with
  1) **DATA ABSTRACTION**

Data Abstraction means to handle complexity by hiding unnecessary details from the user. In java, abstraction is achieved by interfaces and abstract classes. We can achieve 100% abstraction using interfaces.
In Selenium, WebDriver itself acts as an interface. Consider the below statement:
WebDriver driver = new ChromeDriver();
We initialize the Chrome Browser using Selenium Webdriver. It means we are creating a reference variable (driver) of the interface (WebDriver) and creating an Object. Here WebDriver is an Interface and ChromeDriver is a class.
We can apply Data Abstraction in a Selenium framework by using the Page Object Model design pattern. We define all our locators and their methods in the page class. We can use these locators in our tests but we cannot see the implementation of their underlying methods. So we only show the locators in the tests but hide the implementation. This is a simple example of how we can use Data Abstraction in our Automation Framework.

2) **ENCAPSULATION**
Encapsulation is defined as the wrapping up of data under a single unit. It is the mechanism that binds together code and the data it manipulates. Encapsulation can be achieved by: Declaring all the variables in the class as private and writing public methods in the class to set and get the values of variables.
All the classes in an Automation Framework are an example of Encapsulation. In Page Object Model classes, we declare the data members using @FindBy and initialization of data members will be done using Constructor to utilize those in methods.

3) **INHERITANCE**
Inheritance is the mechanism in java by which one class is allowed to inherit the features (fields and methods) of another class.
We can apply Inheritance in our Automation Framework by creating a Base Class to initialize the WebDriver interface, browsers, waits, reports, logging, etc. and then we can extend this Base Class and its methods in other classes like Tests or Utilities. This is a simple example of how we can apply Inheritance in our framework.

4) POLYMORPHISM
Polymorphism allows us to perform a single action in different ways. In Java polymorphism can be achieved by two ways:
– **Method Overloading**: When there are multiple methods with same name but different parameters then these methods are said to be overloaded. Methods can be overloaded by change in number of arguments or/and change in type of arguments.
In Selenium Automation, Implicit wait is an example of Method Overloading. In Implicit wait we use different time stamps such as SECONDS, MINUTES, HOURS etc.
– **Method Overriding**: It occurs when a derived class has a definition for one of the member functions of the base class. That base function is said to be overridden.
In Selenium Automation, Method Overriding can be achieved by overriding any WebDriver method. For example, we can override the findElement method
In assertion we have used overload because in assertion we used to like asset.true(actual, expected) and second time we can use same assert.true(actual, expected, message).

- **How can you use interface and how it is different from Abstract class?**
  Abstract class may have Abstract and concrete methods, and there is not any compulsion in adding abstract method in abstract class. But in Interface, we do have only abstract methods and we don't need to write abstract keyword in Interface this is by default public and abstract.

- **What do you mean by Static keyword in Java?**
  Static means it is at class level not at instance level, we have static method, static variable & static inner class. When we have any variable as static so it will remain same for all the instance of our classes, and static/Private/Final methods can't be over-ridden like if we have initialized any method as Static so we cannot override it in any child class.

- **How to call static method and variable in java?**

  Direct calling, Calling by class name.

- **Can I access Static method by using object reference?**
  Yes we can, but we got one warning that you need to access it via Direct or By class name.

- **How to call non-static method and variable in java?**
  For calling non static method we need to create object first.

- **Can we overload & override main method?**
  Overload-Yes, Override-No

- **What do you mean by wrapper class and how will you do data conversion?**
  Wrapper class in java are used for data conversion. In data conversion if user wants to convert Int to string, String to int, Boolean, double then we use Wrapper class.
  integer.parseInt(); - To convert string to Integer
  Double.parseDouble(); - To convert string to Double
  Boolean.parse Boolean(); - To convert string to Boolean
  String.valueof(); - To convert Integer to String.

- **Can you convert string a =”110a” in integer?**
  No we got NumberFormatException while converting the above string.

- **What do you mean by Call by Value & Call by Reference in Java?**
  Call by value means suppose we have created one sum method with input parameter int a, int b. So while calling the creating the object and running we provide values that is know as call by value.

- **What do you mean by Exceptions in Java?**
  Exception is like any interruption in our normal flow. Like if we are running anything and we got issues in our script this is we called exception, we have 2 types of exception Run Time & Compile Time. (checked & Unchecked exceptions)

- **Can you tell me about difference between Throw and Throws keyword?**
  Throw is a keyword used inside a body of function. And Throws used while initializing any method. By using Throw we can throw only one exception while for Throws we can declare multiple exceptions which might occur in that particular function. Throws keyword followed by instance name and Throw keyword is followed by class name of that exception.

- **How much you rate yourself in selenium out of 5?**
  Out of 5 I will rate myself 3.5 in selenium.

- **Which locator you are using in your framework and why?**
  Mostly we used ID and Xpath because Id is the fastest and unique one and after that we prefer Xpath. Anyways we have other locators as well like css , class name, tag name, Link text, Partial Link text.

- **What is the difference between findelement & findelements?**
  findelement will give the first appearance of that element which matches our locator, whereas findelements will give us list of all the elements which is present over the webpage and matching our locator. And if we don't find the element findelement will give us nosuchelementexception whereas findelements will return NULL/Empty list.

- **Can you tell me how you will handle multiple window in selenium.**
  We have windowhandle & windowhandles function for handling Multiple windows. Windowhandle will give the string value of only the active window that is open whereas windowhandles will give set of all the windows that are open in browser.

- **How you will move from one window to another?**
  First we will check what all windows are open by using driver.getwindowhandles, to get set of opened windows , then I use iterator to iterate over each of the pages and inside for loop will check like Current URL matches with the excepted page, if match then switch to that window by using driver.switchTo(Destination window) -> to return back to main parent window use driver.defaultwindow.

- **Tell me the difference between Implicit & Explicit wait?**
  Implicit wait applies for all the elements and all the tests like if we give 10 sec of implicit wait it will wait for 10 sec for each element before giving nosuchelement exceptions.
  While Explicit wait can be applied for any particular step for which you want extra wait time so we can use explicit wait. We can use mix of both waits to depend on the situation of the step.

- **Can you tell me some exceptions in selenium?**
  NoSuchElementException, NoSuchWindowException NoSuchframeException, StaleElementReferenceException, TimeoutException.

- **Can you tell me about StaleElementReferenceException?**
  Stale means old or decayed, here it sounds like element which was present on that page is no longer there or decayed. To handle this, we can refresh the webpage before pointing to that element. We can write script for waiting via explicit wait by writing expected condition.refresh. Or we can go with page object model in that we can over come this stale element exception.

- **What do you mean by User Defined Exception?**
  User Defined Exception or custom exception is creating your own exception class and throws that exception using 'throw' keyword. This can be done by extending the class Exception. ... The keyword "throw" is used to create a new Exception and throw it to the catch block.

- **Can you tell me what is assert in TestNG?**
  Assert is like verification where we check like expected thing and actual thing are same or not.

- **Which assert you have used in TestNg?**
  We have used Hard assert and Soft assert, while applying Hard assert if we found any glitch in expected and actual then it will through exception and move to next @test while Soft assert it won't give exception and move to next step of that test. And to get all the exceptions in console we need to write at the end assert.all.

- **Can you tell me about the order of TestNG annotations?**
  @BeforeSuite
  @BeforeTest
  @BeforeClass
  @BeforeMethod
  @Test
  @AfterMethod
  @AfterClass
  @AfterTest
  @AfterSuite

- **Do you heard about Priority in TestNg can we set -ve priority?**
  Yes, like priority is 0, -1, TestNg will run -1 then 0 then 1. And if we have any @test which is not having any priority set, then in that case it will search via alphabetic order whichever comes first and execute test respectively.

- **Do you work in cucumber, can you tell me what all files required in cucumber?**
  In cucumber we have Feature file, Step Definition file and Test Runner file.
  In feature file we used to write scenario in gherkin language which is most like in plain English language. Here we use some of the keywords like feature, scenario, scenario outline, given, when, then, and, example, background keywords for writing our test scenarios steps.
  In Step Definition file we write mapping code for all the scenario of feature file.
  In test Runner file we provide the address of the feature file, step definition file, and all-important Tags, Plugin, Listeners in that.

- **What is the difference between scenario & scenario outline?**
  When we have single scenario and we need to run it one time at that place we use Scenario.
  If you want some parametrization or Data Driven testing at that time, we can use scenario outline where we have to use Example keyword like if we are running this scenario for 3 different data set like username & pass. so, it will run 3 times.

- **Can you tell me more about Background Keyword?**
  Background is used when we have some common Given part. Suppose we have pre-condition that we have to check this before each scenario. so in order to avoid rewriting same step we can write it in Background.

- **What is the use of Dry Run in cucumber?**
  Dry run is not running our whole application it will check whether all features are mapped with Step definition.

- **What is hooks in cucumber?**

In cucumber we use hooks for common functionalities, hooks are like we want to run before & after each of the scenario. In hooks we have 2 different @before, @ after which run before and after of each scenario. Also @beforestep, @afterstep which run before and after each step.

- **Can you tell me how you will re-run failed scenario in cucumber?**
  For that we can use re-run attribute in our test runner file. After that we can write one file location. Where all the test cases which failed while execution get stored. So next time while running execution we can give this file location and run the failed TC.

- **You have worked in Cucumber & TestNG according to you which one is best?**
  I will consider Cucumber as it is most likely understood by Laymen people which is English plain language. Because in order to understand the functionality flow no need to go look and script/code. Via Scenario steps lines only we can get clear understanding about the functionality.
  It helps to come all the QA members Dev, Client, Product Owner on same page.

- **Can you explain me TestNG?**
  TestNG is advanced version of Junit only. It is mainly used by Dev/QA for maintain the code easily and for unit testing. It provides lots of benefits to us like we can create a suite and we can write all the required Tc in one go only using that suite. We can group our Tc we can set priority we can run our tc in parallel mode, We can generate good reports via TestNG. We can write functionality depends on methods, depends on group. We can run single tc multiple time with single set of data of multiple set of Data.

- **How to run single method multiple time in TestNG?**
  We have invocation count attribute in @test annotiation. We can write invocation count as 3 if we want to run it 3 times. Apart from that we can write threadpull.size if we want to run that case in multiple thread.

- **Have you used GIT in your project can you explain about it?**
  Yes I have used GIT, It is a version control tool. Where we can maintain our central repo. we used to manage our code via GIT only. We use Git to maintain our project in our local system. So, if someone like to work on that project I need to send complete update copy to him and after that he can work on that. There are chances that single project is handled by multiple teams across the globe. So, it will be difficult if we won't use GIT.

- **Can you give me some GIT commands which you used on daily basis?**
  Git status- which shows status of all the files,if we have some files which is not yet added to our repo so it will give us untracked file.
  After that we can use GIT add command after adding it will added to particular index and we can commit this file using Git Commit-(Message) we can commit this untracked file. Also we have Git Merge, Git Post, Git Pull, Git It in etc.

- **How to solve Merge conflict in GIT?**
  As we are only 2 tester working on this project, if we have any merge conflict I used to pull all the latest file/scripts to my local system. Then I will analyze the difference between that particular file and merge file. After that I will check with my team member whether all his imp things are covered then I will add my steps and push the script to the central repo.

- **You have worked in Jenkins can you tell me how you have created jobs in Jenkins?**

We have separate Dev-Ops Team to create Jenkins jobs at broad level but we also have access to jenkins, so we have created jobs for our internal purpose.
For creating any job we have click on create new job->inside that give name of your job->select freestyle project->then add. Beside that we can provide description of our project and in source code management we can choose Git-> provide repo url ->after that provide some schedule if you want to run the job on any specific schedule time.-> select window batch command-file location-save-click on build now for running. After triggering we can check log in console.

- **What is the difference between Smoke & Sanity Testing?**
  Smoke and Sanity we can are like same thing because both are checking important functionality.
  Smoke testing is done on first stable build from developer to check like whether it is stable enough to move further or not. While Sanity testing is subset of regression test which we perform on stable build and here also we used to check all the imp functionality.

- **What is Agile ceremony?**
  We have 4 Agile ceremony -Sprint planning, Sprint review, Sprint Retrospective, Daily scrum meeting.

- **Why the main method is static?**

  Java **main**() **method** is always **static**, so that compiler can call it without the creation of an object or before the creation of an object of the class. ... **Static method** of a class can be called by using the class name only without creating an object of a class.

- **What is Run time polymorphism**

  Run-**Time Polymorphism**: Whenever an object is bound with the functionality at **run time**, this is known as **runtime polymorphism**. The **runtime polymorphism** can be achieved by method overriding. Java virtual machine determines the proper method to call at the **runtime**, not at the **compile time**.

- **Difference between list and set.**

  The main **difference between List and Set is** that **Set is** unordered and contains different elements, whereas the **list is** ordered and can contain the same elements in it.

- **Method overloading and overriding.**

  Method overriding is used to provide the specific implementation of the **method** that is already provided by its super class. **Method overloading** is performed within class. **Method overriding** occurs in two classes that have IS-A (inheritance) relationship. In case of **method overloading**, parameter must be different.

- **Use of constructor.**

  The purpose of **constructor** is to initialize the object of a class while the purpose of a method is to perform a task by executing java code. **Constructors** cannot be abstract, final, static and synchronised while methods can be. **Constructors** do not have return types while methods do.

- **Difference between static and non-static methods**

  Static method uses complie time binding or early binding. **Non-static method** uses run time binding or dynamic binding. A **static method** cannot be overridden being compile time binding. A **non-static method** can be overridden being dynamic binding.

- **Explain Git workflow.**

Step 1: Set up a Github Organization. ...
Step 2: Fork Organization Repository to Your Personal GitHub. ...
Step 3: Clone the Repository to Your Local Machine. ...
Step 4: Create a Branch for your Working Files. ...
Step 5: Set Remote Repository to the GitHub Organization. ...
Step 6: Get Coding!
Step 7: Pull the Most Recent Files From the Organization Repo
Step 8: Merge the Master Branch Into the Feature Branch
Step 9: Push Your Code to your GitHub Repo
Step 10: Make a Pull Request to the Organization Repo

- **How to set up Jenkins ?**

Step 1 − Go to the Jenkins dashboard and Click on New Item

Step 2 − In the next screen, enter the Item name, in this case we have named it Helloworld. Choose the 'Freestyle project option'

Step 3 − The following screen will come up in which you can specify the details of the job.

Step 4 − We need to specify the location of files which need to be built. In this example, we will assume that a local git repository(E:\Program) has been setup which contains a 'HelloWorld.java' file. Hence scroll down and click on the Git option and enter the URL of the local git repository.

Note − If you repository if hosted on Github, you can also enter the url of that repository here. In addition to this, you would need to click on the Add button for the credentials to add a user name and password to the github repository so that the code can be picked up from the remote repository.

Step 5 − Now go to the Build section and click on Add build step → Execute Windows batch command

Step 6 − In the command window, enter the following commands and then click on the Save button.

```
Javac HelloWorld.java
Java HelloWorld
```

Step 7 − Once saved, you can click on the Build Now option to see if you have successfully defined the job.

Step 8 − Once the build is scheduled, it will run. The following Build history section shows that a build is in progress.

Step 9 − Once the build is completed, a status of the build will show if the build was successful or not. In our case, the following build has been executed successfully. Click on the #1 in the Build history to bring up the details of the build.

Step 10 − Click on the Console Output link to see the details of the build

- **Can we declare many interfaces object class inside the interface class.**

  Yes, **you can** define a **class inside** an **interface**. **In** general, if the methods of the **interface** use this **class** and if **we** are not using it anywhere else **we will declare** a **class** within an **interface**.

- **Types of the assertion.**

  Selenium Assertions can be of three **types**: "**assert**", "verify", and " waitFor". When an "**assert**" fails, the test is aborted. When a "verify" fails, the test will continue execution, logging the failure. A "waitFor" command waits for some condition to become true.

- **Abstraction in java and exp?**
  In Page Object Model design pattern, we write locators (such as id, name, xpath etc.,) in a Page Class. We utilize these locators in tests but we can't see these locators in the tests. Literally we hide the locators from the tests.

- **What is a super keyword in java?**

  The **super keyword** refers to superclass (parent) objects. It is used to call superclass methods, and to access the superclass constructor. The most common use of the **super keyword** is to eliminate the confusion between superclasses and subclasses that have methods with the same name.

- **Difference between break and continue statement.**

  Break statement resumes the control of the program to the end of loop and made executional flow outside that loop. **Continue statement** resumes the control of the program to the next iteration of that loop enclosing '**continue**' and made executional flow inside the loop again

- **Diff between Abstract class & interface?**

  Abstract class can inherit another **class** using extends keyword and implement an **interface**. **Interface** can inherit only an inteface. **Abstract class** can be inherited using extends keyword. **Interface** can only be implemented using implements keyword.

- **What is a static keyword in Java?**

  In the **Java** programming language, the **keyword static** indicates that the particular member belongs to a type itself, rather than to an instance of that type. This means that only one instance of that **static** member is created which is shared across all instances of the class.

- **Have you used the action class and where it is used?**

  Using the **Actions class** in Selenium, we can implement the sendKeys() method to type specific values in the application. That is how **you use** the **actions class** in Selenium with sendKeys() method. ... The perform() method is **used** to perform the series of **actions** that **are** defined.

- **What is the difference between checked and unchecked exceptions?**

There are two types of **exceptions**: **checked exception** and **unchecked exception**. ... The main **difference between checked and unchecked exception** is that the **checked exceptions** are **checked** at compile-time while **unchecked exceptions** are **checked** at runtime

checked exceptions – SQLException,IOException,ClassNotFoundException,InvocationTargetException

unchecked exceptions –
NullPointerException,ArrayIndexOutOfBoundsException,ArithmeticException,IllegalArgumentException

NumberFormatException

- **Apart from sendkeys, are there any different ways, to type content onto the editable field?**

  WebDriver driver = new FirefoxDriver();
  JavascriptExecutor executor = (JavascriptExecutor)driver;
  executor.executeScript("document.getElementById("textbox_id").value='new value';);

- **What is static and non-static?**

  In non-static method, the method can access static data members and static methods as well as non-static members and method of another class or same class. Binding process. Static method uses compile time or early binding. Non-static method uses runtime or dynamic binding. Overriding.

- **DIfference between this and super?**

  this keyword mainly represents the current instance of a class. On other hand **super** keyword represents the current instance of a parent class. this keyword used to call default constructor of the same class.

- **What is the difference between length and length() in Java?**

  The **length** is an instance variable of an array in **Java** whereas **length()** is a method of String class

- **What is an abstract class?**

  Abstract Classes and Methods **Abstract class**: is a restricted **class** that cannot be used to create objects (to access it, it must be inherited from another **class**). **Abstract** method: can only be used in an **abstract class**, and it does not have a body. The body is provided by the subclass (inherited from).

- **Difference between Actions and Action?**

  Actions is a class that **is** based on a builder design pattern. This **is** a user-facing API for emulating complex user gestures. Whereas **Action is** an Interface which represents a single user-interaction **action.**

- **How do you handle keystrokes in Selenium?**

  Using Actions Class: Actions action = new Actions(driver); action. keyDown(**Keys**. ...
  Using SendKeys Chord: driver. findElement(By. ...
  Using Robot Class: // Create Robot class Robot rb = new Robot(); // Press control keyboard key rb.

- **What is dry run in Cucumber?**

  Dry-run is used to compile feature files and step definitions in **cucumber**. It is specially used in the stage when you will have to see if there are any compilation errors, to check that you can use **dry-run**. **Dry-run** options can either set as true or false.

- **Annotations in Cucumber**

  Total 11 Annotations -Feature, Scenario, Background, given, when , then, and, but, example, scenario outline, scenario template.

- **What are hashmap and HashSet? Explain?**

  HashMap and HashSet both are one of the most important classes of Java Collection framework. ... **HashMap** Stores elements in form of key-value pair i.e each element has its corresponding key which is required for its retrieval during iteration. **HashSet** stores only objects no such key value pairs maintained.

- **Where do you use a hashmap?**

Maps are **used** for when you want to associate a key with a value and Lists are an ordered collection. Map is an interface in the Java Collection Framework and a **HashMap** is one implementation of the Map interface. **HashMap** are efficient for locating a value based on a key and inserting and deleting values based on a key. HashMap<String, Integer> map = new HashMap<>();

```
// Add elements to the map
map.put("vishal", 10);
map.put("sachin", 30);
map.put("vaibhav", 20);

// Print size and content
System.out.println("Size of map is:- "
                   + map.size());
System.out.println(map);

// Check if a key is present and if
// present, print value
if (map.containsKey("vishal")) {
    Integer a = map.get("vishal");
    System.out.println("value for key"
                       + " \"vishal\" is:- " + a);
```

- **How do you handle if XPath is changing dynamically?**

  Option 1: Look for any other attribute which Is not changing every time In that div node like name, class etc. So If this div node has class attribute then we can write xpath as bellow.

  //div[@class='post-body entry-content']/div[1]/form[1]/input[1]

  Option 2: We can use absolute xpath (full xpath) where you do not need to give any attribute names In xpath.
  /html/body/div[3]/div[2]/div[2]/div[2]/div[2]/div[2]/div[2]/div/div[4]/div[1]/div/div/div/div[1]/div/div/div/div[1]/div[2]/div[1]/form[1]/input[1]

  Option 3: We can use starts-with function. In this xpath's ID attribute, "post-body-" part remains same every time. //div[starts-with(@id,'post-body-')]/div[1]/form[1]/input[1]

  Option 4: We can use contains function. Same way you can use contains function as bellow.div[contains(@id,'post-body-')]/div[1]/form[1]/input[1]

- **Does Jenkins require a local system for CI?**

  It is a server-based application and requires a web server like Apache Tomcat

- **What is a singleton class?**

  The Singleton's purpose is to control object creation, limiting the number of objects to only one. Since there is only one Singleton instance, any instance fields of a Singleton will occur only once per class, just like static fields. Singletons often control access to resources, such as database connections or sockets. For example, if you have a license for only one connection for your database or your JDBC driver has trouble with multithreading, the Singleton makes sure that only one connection is made or that only one thread can access the connection at a time.

- **When finally block get executed?**

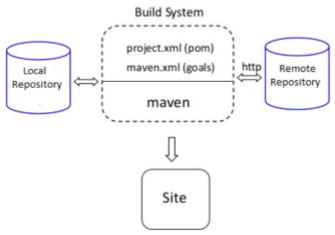
  The **finally block** always **executes** when the try **block** exits. This ensures that the **finally block** is **executed** even if an unexpected exception occurs.

- **How many times you can write catch block?**

  maximum **one catch block will** be executed. No, **we can write multiple catch block** but only **one** is executed at a **time.**

- **What Maven Architecture and explain pom.xml?**

Maven architecture diagram:



POM is an acronym for Project Object Model. The **pom**. **xml** file contains information of project and configuration information for the **maven** to build the project such as dependencies, build directory, source directory, test source directory, plugin, goals etc. **Maven** reads the **pom**.

Real time selenium + Java interview questions.

- **How many types of WebDriver API 's are available in selenium?**
  Chrome, Geko, Chromium, Edge, html, android,
- **How can you make sure that page is loaded via wed driver or selenium?**
  Via first apply wait , is element present, then get text.
- **How can we launch a batch file in selenium webdriver project?**
  Take path of batch file->process.batchjob= runtime.get(runtime.executable"Path"));
- **How do you run selenium webdriver test from the command line?**
  For that go to cmd-> java-class path(of the selenium project) ->hit enter
- **What are the different exception you faced in selenium webdriver?**
  Webdriver exc, noalertpresent exc, nosuchwindow exc, nosuchelement exc, timeout exc.
- **How do you scroll down a page using javascript in selenium?**
  Windows.scrollby function
- **How do you scroll down to a particular element?**
  Windows.scroll.intoview function
- **Which all files can be used as a data source for different frameworks?**
  .csv, .xml, .text etc
- **What are listeners in selenium?**

Listeners actually is an interface, that modifies the behavior of the system. It is used for customization of reports. 2 types webdriver listeners, TestNg Listeners.

- **How do you take screenshots in selenium webdriver?**
  takescreenshot function
- **What do you mean by assertions in selenium?**
  Assert, verify, wait for
- **How many phases are there in maven build cycle?**
  6 validate-compile-test-package-install-deploy
- **How will you handle keyboard and mouse related action using selenium?**
  By action class, robot class, venium driver
- **What do you mean by WebDriver?**
  Webdriver is an interface which is used to automate api of browser for testing.
- **How do you handle drag and drop option?** Using action classes
- **How you handle java pop-ups in selenium?** Using alert, switch to alert, accept, dismiss, get text.
- **What does means Public static void main(variable,value)**
  Public/private/protected/default-Access specifier
  Static- modifier
  Void- return type
  Main-class name
- **How to input text into a text box without Sendkeys?**
  JavascriptExecuter executor = (JavascriptExecutor)driver;
  executor.excuteScript("document.getElementById("<<inputbox_id>>").value='new value');
- **What are the open source frameworks supported by selenium webdriver?**
  TestNG, Junit, Cucumber, Robot Framework, Appium, Protractor.
- **How to handle hidden elements in selenium webdriver?**
  JavascriptExecuter js = (JavascriptExecutor)driver;
  js.excuteScript("document.getElementById("<<displayed_text>>").value='Hiddentext);

- **How to handle iframes in selenium webdriver?**
  driver.switchTo().frames(via index value, name, webelement);
  driver.findeElement(by.id("value")).getText();
  driver.switchTo().defaultContent();-To get back from iframe

- **How you handle dropdown values?**
  From select class, via visible text, value, index
- **How to get color of webelement using selenium webdriver?**
  First get the locator of webElement , then get
  String color= object.getCssValue("background-color")
  String HexbackColor= color.fromString(color).asHex();
  It will give you RGb codes , you need to convert them into back color using HEX function
- **How you handle alert in selenium webdriver?**
  Simple alert(one option), Confirm Alert(Y/N), Prompt alert(enter any value)
  Alert a= driver.switchTo().alert();
  a.getText();
  a.accept(), a.dismiss(), a.sendKeys("name");
- **How you handle multiple windows tabs in selenium webdriver?**
  String PID=driver.getWindowHandle();
  Set<String> allWindowHandle= driver.getWindowHandles();

Apply for loop on allWindowHandle -> switchTo().window(Id); ->if(!(Id.equals(PID)) ->driver.close();

JAVA ONE LINE CONCEPTS

- java file mai ek hi public class hoti hai, uske alava aur classes v ho sakti hai par public ek hi.
- If we need to create one variable for multiple values, we need to use Array concept.
- Int marks[] = new int[5]
- Array can store only homogenous data, int for int array, string for string,
- If we need to add heterogeneous data in array, we need to create object array.
- Object a[] = new Object[5]; now we can add different data type objects.
- Array is fixed in size, which we define while creating.
- If we try to access index value >= given index value, we got arrayOutOfBundException.
- Arrays are not defined by any data layer structure so we can't run readymade methods on it.
- To overcome this, we have collection framework under which there are ArrayList, List, HashMap, HashTable, Tree, Stack.
- We can add new elements in run time under collections while in array we cannot.
- Collection is a group of objects. To represent this we need certain interfaces and classes.
- Common operations we generally do on collections are **adding objects, removing objects & finding object.**
- **Collection (I)** is called collection interface having methods which are common throughout all collections.
- **Collections** is basically a class from java.util package which contains some methods which we can use for collection objects.
- Collection – 1. List, 2. Set, 3. Queue
- **List (I)** is child of collection(I). In list Insertion order is preserved and duplicates are allowed.
- **ArrayList, LinkedList, Vector** these are different classes which implements **List Interface**.
- **Set(I)** is child of collection(I). Insertion order is not preserved & duplicates not allowed.
- **HashSet, Linked Hashset** these are different classes which implements **Set Interface**.
- **Queue(I)** is child of collection(I). We used it when we need prior to processing means first in first out concept.
- **priorityQueue** is class which implements **Set Interface.**
- There is one independent interface known as **Map(I).** In Map(I) objects are created with **key and value** pair. Key cannot be duplicate, but Value can be.
- **Hashmap, Linked Hashmap, Hash Table** these are different classes which implements **Map Interface.**
- Whatever methods present in Collection(I) are also present in their child interface i.e List, Set, Queue.
- **add(object o), addAll(Collection c), remove(Object o), removeAll(Collection c), retainAll(Collection c)** these are some methods of Collection Interface.
- **clear(), isEmpty(), size(), contains(), conatinsAll(), toArray()** are also some methods.
- In List **index** play an important role because with the help of index only we can find duplicates elements.

- **add(index , object), get(index), set(index, object)** are methods of List Interface.
- **ArrayList al= new ArrayList(),** it allows heterogenous objects also.
- **ArrayList<Str> al= new <str>(),** now it can store objects of string only.
- **Collections.Sort(al) , Collections.Shuffle(al)** This will sort & shuffle the objects of arraylist.
- We can read the data with for loop, for each loop, iterator () method.
- JVM have 2 types of memories **Static pool -static data, heaps-Non static data,**
-

==========

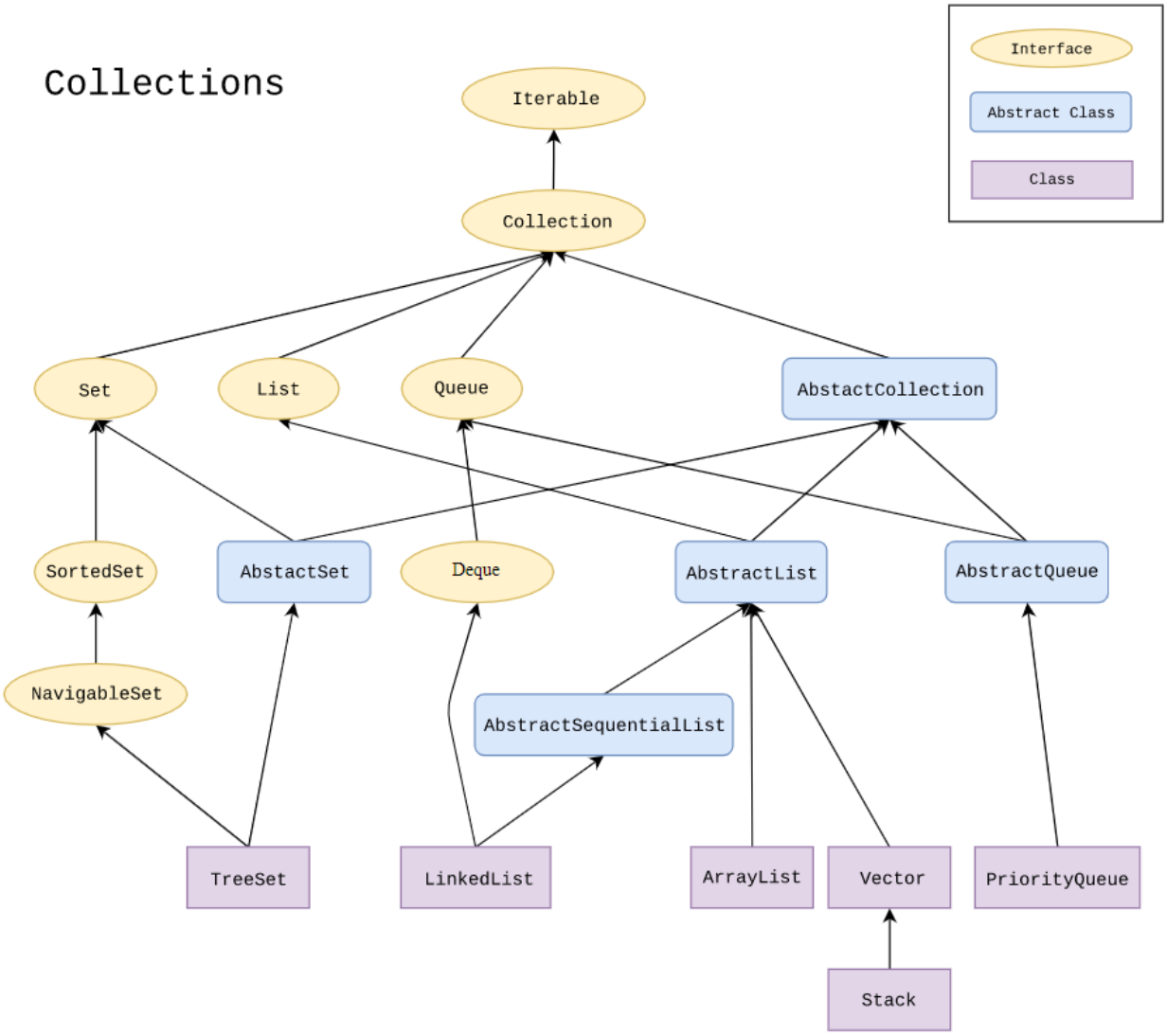
OOPS ONE LINE CONCEPTS

- We can create multiple classes inside one main class.
- Class name cannot be a keyword.
- **Object.** with . we can access behavior and properties of that class or any other public class.
- In **Polymorphism**, we can create many methods with same name, differentiating with input parameter. Void Walk(){}, void walk(int steps){} . We cannot create duplicate methods or one method into another.
- **Compile time polymorphism**, tell which method is called before running it.
- By applying **static** it means now it became class property not object one and it applicable to all objects throughout the class.
- **Static methods** are accessed by class, **Non Static methods** are accessed by objects.
- **Constructor** in java is to make objects, we can create constructor but there is one default constructor in java, like **students a = new students();** highlighted is the default constructor.
- **Constructor** are non static methods. Constructor can be **parameterized**
- We can create our own constructor like **students b= new students(int rollNo ,String name){}**
- **this()** keyword is used to call one constructor by another.
- **this()** keyword also used in agar 2 alag alag methods mai same parameter name hai , to conflict na ho isliye **this.name**, **this.rollNo** use krte h. this is known as method overloading bcs you are using one method parameter in second one.
- **Inheritance** ka matlab hai ki Parent class ki property ko hum child class mai bhi use kar sake.
- e.g **class Sci extends students{},** extends keyword ki help se hum students class ki property ko Sci class mai inherit kr rahe h.
- **super()** keyword parent class ke constructor ko child mai le ata hai , means ab agar hum Sci class ka object baneyenge to students class ka v ek object banega.
- **this()** keyword call current class property, **super()** keyword call parent class property.
- Java ke andar jitne v by default objects hote hai unke ek parents class hoti hai **object.**
- **Run time polymorphism** tell which method is called while running it.
- Public method ka matlab hai hum us method ko ab jis class mai vo bna hai uske bhar bhi access kr sakte hai uski class ka object bna ke.
- Aur isi point par **Encapsulation** ka concept v ata hai , hum kisi v method ko kisi v class mai use kr sakte hai agar vo public hua , par agar hamne usko public na kar to ab vo method apni

class mai Encapsulate ho gya ab usko koi aur use nahi kar skta, ise hi Encapsulation bolte hai.

- Public, private, protected inko access modifier bolte hai.
- Agar hamne kisi method ko Private bnaya hai to vo bas usi class mai hi use ho skta hai aur kahi nahi.
- Agar hum chate hai ki parent class ki property bas uske child class ko mile to un properties/methods ko hum Protected kar dete hai.
- **Encapsulation** isliye v karte hai bcs hame same kind of variables and method jo kisi ek kam ke liye specially bne ho unko ek sath le aye. Is se data hiding mai help milti hai and security v increase hoti hai.
- Aur yahi par **getter** and **setter** ka funda ata hai.
- **Abstraction** iska matlba hai ki hum unnecessary info ko user se hide kre taki usko main functionality par focus rahe na k internal logic pr.
- e.g **ATM** withdraw, user ko bas apna pin no cahiye aur kitna paise nikalana hai , uske piche kya kya process hota hai uska use koi lene dene nahi hai. To jis process ka user ko kuch lena dena nahi use background mai rakhna ise **Abstraction** kehte h.
- **abstract** keyword agar kisi class ke sath lagaya hai to hum uska objects nahi bna sakte.
- Ha bas abstract class ko extend kake hum uske children class mai objects bna skate hai only override karke.
- Aur agar hum kisi class ko abstract banate hai to hame usme kuch logic dene ki jaroort nahi hai. Aur ye ek imp property hai abstract method ki.
- Abstract method bannna hai to uski class ko bhi abstract bannna padega.
- **Interface methods** are by default public and abstract.
- **Abstract method** ke andar hum concrete functions bna sakte hai , par **interface** ke andar kuch nahi bna sakte.
- **Interface** hame complete **Abstraction** provide karata hai.
- **Interface** ko use karne ke liye hum **implements** keyword use karte h.
- Class-**extends**-class, class-**implements**-interface, interface-**extends**-interface
- **Java** mai ek class ke multiple parent nahi ho sakte bcs java multiple inheritance support ni krta. Iske liye hum **interface** ka use karte hai.
- **Tokens** in java, Token is known as the smallest unit of your program, the whole java code you have written is a token.
- **Literals** means are the string values, integer values, comments, keyword.

COLLECTION FRAMEWORK



- A collection represents a group of objects.
- Java collections provide classes and Interfaces for us to be able to write code.
- We need collections for efficient storage and better manipulation of data in java.
- Collection reduces programming effort, provide in-build methods and classes.
- **ArrayList** -> For variables size collections
- **Set** -> For distinct collection
- **Stack**(queue)-> A LIFO (Last In First Out) data structure
- **HashMap** -> For strong key - value pairs
- **Iterator**- To iterate the element from collection.

- Collections class is available in java util package collection class also provides static methods for sorting, searching.
- **Common methods** available in Collection are add(), addAll(), remove(), removeAll(), size(), clear(), contains(), containsAll(), retain(), retainAll()
- **Common exception** is collections are NullPointerException, ClassCastException, IllegalargumentException, IllegalStateException, UnsupportedOperationException
- 

Collection	Ordering	Random Access	Key-Value	Duplicate Elements	Null Element	Thread Safety
ArrayList	✓	✓	✗	✓	✓	✗
LinkedList	✓	✗	✗	✓	✓	✗
HashSet	✗	✗	✗	✗	✓	✗
TreeSet	✓	✗	✗	✗	✗	✗
HashMap	✗	✓	✓	✗	✓	✗
TreeMap	✓	✓	✓	✗	✗	✗
Vector	✓	✓	✗	✓	✓	✓
Hashtable	✗	✓	✓	✗	✗	✓
Properties	✗	✓	✓	✗	✗	✓
Stack	✓	✗	✗	✓	✓	✓
CopyOnWriteArrayList	✓	✓	✗	✓	✓	✓
ConcurrentHashMap	✗	✓	✓	✗	✗	✓
CopyOnWriteArraySet	✗	✗	✗	✗	✓	✓

- **Thread Safety**-When multiple threads are working on same data, and the value of our data is changing, that scenario is not thread-safe and we will get inconsistent results. When a thread is already working on an object and prevent another thread on working on the same object is known as thread safety. We can achieve Thread safety via Synchronization, Volatile Keyword, Atomic variable, Final Keyword.

Array List:

- ArrayList<Object Type> ar = new ArrayList<Object Type>();
- ArrayList is Dynamic in nature.
- Virtual Capacity of ArrayList by default is 10 but Physical capacity if we did not add any object is 0. Once we start adding Physical objects Virtual Capacity got decreased by same.

Hashmap:

- Hashmap<String, String>capitalmap = new Hashmap<String, String>();
capitalmap.put("India", "New Delhi");

Java HashMap Interview Questions:

1.How does put() method of HashMap works in Java?

On hashing principle of key value pair

2. What is the requirement for an object to be used as key or value in HashMap?
3. What will happen if you try to store a key which is already present in HashMap?

4. Can you store a null key in Java HashMap?
5. Can you store a null value inside HashMap in Java?
6. How does HashMap handle collisions in Java?
7. Which data structure HashMap represents?
8. Which data structure is used to implement HashMap in Java?
9. Can you store a duplicate key in HashMap? (answer)
10. Can you store the duplicate value in Java HashMap? (answer)
11. Is HashMap thread-safe in Java?
12. What will happen if you use HashMap in a multithreaded Java application?
13. What are the different ways to iterate over HashMap in Java?
14. How do you remove a mapping while iterating over HashMap in Java?
15. In which order mappings are stored in HashMap?
16. Can you sort HashMap in Java? (answer)
17. What is the load factor in HashMap? A load factor is a number that controls the resizing of HashMap when a number of elements in the HashMap cross the load factor as if the load factor is 0.75 and when becoming more than 75% full then resizing trigger which involves array copy.
18. How does resizing happens in HashMap? (answer)
19. How many entries you can store in HashMap? What is the maximum limit?
20. What is the difference between the capacity and size of HashMap in Java?
21. What will happen if two different keys of HashMap return the same hashcode()?

==========

String one line que.

java.lang.String class is used to create a string object.
Different String methods:

- **compareTo** - The Java String compareTo() method is used for comparing two strings lexicographically.
- **boolean equals()** - The java string equals() method compares the two given strings based on the content of the string (case sensitive)
- **String concat()** – concat two strings
- **boolean equalsIgnoreCase()** - The java string equals() method compares the two given strings based on the content of the string (not casesensitive)
- **char charAt()** – index position - The java string charAt() method returns a char value at the given index number.
- **boolean contains()**
- **toUpperCase()** – convert to upper case
- **toLowerCase()** – convert to lower case
- **trim()** – remove spaces from both sides of string
- **substring()** -- returns part of string
- **boolean endsWith()**
- **boolean startWith()** – ends with specified suffix or not

- **int length()**
- **replace()**
- int num = Integer.parseInt(str);- **Convert String to int using Integer.parseInt(String)**
- int num = Integer.valueOf(str);- **Convert String to int using Integer.valueOf(String)**
- **Convert int to String using String.valueOf()**

```
String int ivar = 123;
      String str = String.valueOf(ivar);
      System.out.println("String is: "+str);
      System.out.println(555+str);
```

- **Convert int to String using Integer.toString()**

```
 int ivar = 123;
     String str = Integer.toString(ivar);
     System.out.println("String is: "+str);
     System.out.println(555+str);
```

- In java, string objects are immutable. Immutable simply means unmodified or unchangeable.Once string object is created its data or state can't be changed but a new string object is created.

==========

Array one line que.

- **Write down syntax of iterator function?**

```
Iterator<String> it = studentList.iterator();
while(it.hasNext()){
System.out.println(it.next());
}
```

- **How to find max min of an unsorted array?**

MAX

```
public class maxmin {
public static void main(String[] args) {
int [] arr = {1, 45, 67, 98, 455, 678};
int max = Integer.MIN_VALUE;
for ( int element : arr){
      if(element>max){
       max=element;
      }
}
System.out.println("Max element is " + max);
}}
```

MIN

```
public class maxmin {
public static void main(String[] args) {
int [] arr = {1, 45, 67, 98, 455, 678,-6};
int min = Integer.MAX_VALUE;
```

```
for ( int element : arr){
      if(element<min){
       min=element;
      }
}
System.out.println("Min element is " + min);
}}
```

==========

- **How to reverse any array?**

```
public class reverse array {
public static void main(String[] args) {
int [] arr = {1, 45, 67, 98, 455, 678};
int l = arr.length;
int n = Math.floorDiv(l,2);
int temp;
   for(int i=0; i<n;i++){
   temp= arr[i];
   arr[i]= arr[l-i-1];
   arr[l-i-1]= temp;
}
for(int element:arr)
{
System.out.print( element + " ");
}
}}
```

==========

```
public class reverse array {
public static void main(String[] args) {
int [] Array ={7,8,9,3,4,6,11,67,98};
int k=Array.length-1;
for(k=Array.length-1;k>=0;k--){
   System.out.print( Array[k] + " ");
}
   }}
```

==========

- **How to remove duplicate elements from ArrayList?**

  we can handle this scenario via **LikedHashSet**

```
ArrayList<Integer> numbers = new ArrayList<Integer>(Arrays.asList(1,2,2,4,6,7,2,3,5,4,3,8,2,8));
LinkedHashSet<Integer> linkedHashSet = new LinkedHashSet<Integer>(numbers);
ArrayList<Integer> numbersListWithoutDuplicate = new ArrayList<Integer>(LinkedHashSet));
System.out.println(numbersListWithoutDuplicate);
```

  Also we can handle this via stream

```
ArrayList<Integer> marksList = new ArrayList<Integer>(Arrays.asList(1,2,2,4,6,7,2,3,5,4,3,8,2,8));
List<Integer> marksListUnique= marksList.stream().distnict().collect(Collectors.toList());
System.out.println(marksListUnique);
```

- **How to compare two array list?**
  Via Collection.sort(); and equal

- **How to find additional element in list while comparing 2 List?**
  If we have 2 list l1 & l2 , first we remove all element of l2
  L1.removeAll(l2):
  Sysout(L1) – you will get additional element.

- **How to find common element in list while comparing 2 List?**
  L1.retainAll(L2);
  Sysout(L1) – you will get common element.

- **How will you print length of string without using length method.**
  String str = “Pankaj”
  Sysout(str.toCharArray().length);
  Sysout(str.lastIndexOf(“”));

- **How to find missing element in integer array?**

- **How to reverse a string?**
  1.
  String str = “Pankaj”;
  int len = str.length();
  String rev = ” ”
  for(int i<len-1 , i>=0, i--){
  rev = rev + str.charAt(i);
  }
  Sysout(rev);

  2.
  Create a string-> create new stringBuffer and here you can apply reverse fuction.
  String str = “Pankaj”;
  StringBuffer sf = new StringBuffer(s);
  Sysout(sf.reverse());

- **How will you remove special/junk char from string?**
  We have to use regular expression [a-z, 0-9, A-Z]

  String str = “Y^%^*%&*^*(&*(Pankaj”;
  Str= Str.replaceAll([^a-z, 0-9, A-Z], “”);

Sysout(str);

- **How to reverse an Integer?**

```
int num = 12345;
int rev = 0;
while(num !=0){
rev =rev *10+ num % 10;
num = num/10;
}
Sysout (rev)
}
```

- **How to find duplicate char using hashmap?**
- **Find the count of char using hashmap?**
- Top 7 Selenium Commands with Details
- **#1) get() Methods**
- driver.get("https://google.com");
-
- driver.getClass();
-
- driver.getCurrentUrl();
-
- driver.getPageSource();
-
- driver.getTitle();
-
- driver.getText();
- driver.findElement(By.id("findID")).
  getAttribute("value");
-
- driver.getWindowHandle();
-
-
- ***#2) Locating links by linkText() and partialLinkText()***
- *driver.findElement(By.linkText("Google")).click();*
-
  driver.findElement(By.partialLinkText("abode")).click();
-
- ***#3) Selecting multiple items in a drop dropdown***
- // select the multiple values from a dropdown
- Select selectByValue = new Select(driver.findElement(By.id("SelectID_One")));
- selectByValue.selectByValue("greenvalue"); - By Value
- selectByValue.selectByVisibleText("Red"); - By Visible Text
- selectByValue.selectByIndex(2); - By Index

- ***#4) Submitting a form***
- // submit the form
- driver.findElement(By.id("submit")).submit();
- ***#5) Handling iframes***
- **Select iframe by id**
  driver.switchTo().frame("ID of the frame");
-
- **Locating iframe using tagName**
- *driver.switchTo().frame(driver.findElements(By.tagName("iframe").get(0));*
-
- **Locating iframe using the index:**
- **a) frame(index)**
  driver.switchTo().frame(0);
- **b) frame(Name of Frame)**
  driver.switchTo().frame("name of the frame");
- **c) frame(WebElement element)**
  Select Parent Window
  driver.switchTo().defaultContent();
-
- ***#6) close() and quit() methods***
- *driver.close();* - closes only a single window that is being accessed by the WebDriver instance currently
-
- *driver.quit();*- closes all the windows that were opened by the WebDriver instance
-
- ***#7) Exception Handling***
- WebElement saveButton = driver.findElement(By.id("Save"));
- try{
- if(saveButton.isDisplayed()){
- saveButton.click();
- }
- }
- catch(NoSuchElementException e){
- e.printStackTrace();
- }
- ***#4) isEnabled()***

- **isEnabled()** to Check Whether the Element is Enabled Or Disabled in the Selenium WebDriver.
-
- **findElement(By, by) with sendKeys() to type in the form fields.**
-
- **findElement(By, by) with getText() to store value of targeted web element.**
-
- **Submit()** to submit a web form.
-
- **findElements(By, by)** to get the list of web elements**.**
- List<WebElement> allChoices = dropDown.findElements(By.xpath(".//fruitoption"));
- **findElements(By, by) with size()** to verify if an element is present**.**
- Boolean checkIfElementPresent=
  driver.findElements(By.xpath("//input[@id='checkbox2']")).size()!= 0;
- **pageLoadTimeout(time,unit)** to set the time for a page to load
- driver.manage().timeouts().pageLoadTimeout(500, SECONDS);

- **implicitlyWait()** to set a wait time before searching and locating a web element.
- driver.manage().timeouts().implicitlyWait(1000, TimeUnit.SECONDS);
- **untill() from WebdriverWait and visibilityOfElementLocated()** from ExpectedConditions to wait explicitly till an element is visible in the webpage.
- WebDriverWait wait = new WebDriverWait(driver, 10);
- WebElement element = wait.until(ExpectedConditions.visibilityOfElementLocated (By.xpath("//input[@id='name']")));
- **untill() from WebdriverWait and alertIsPresent() from ExpectedConditions to wait explicitly till an alert appears.**
-
- WebDriverWait wait = new WebDriverWait(driver, 10);
- WebElement element = wait.until(ExpectedConditions.alertIsPresent()
-);
- **Select class for selecting and deselecting values from the drop-down in Selenium WebDriver.**
- WebElement mySelectedElement = driver.findElement(By.id("select"));
- Select dropdown= new Select(mySelectedElement);dropdown.selectByVisibleText("Apple");
- **navigate()** to navigate between the URLs**.**
- driver.navigate().to("https://www.softwaretestinghelp.com");
- driver.navigate().back();
- driver.navigate().forward();
- **getScreenshotAs()** to Capture the entire page screenshot in Selenium WebDriver.
- File shot = ((TakesScreenshot)driver).getScreenshotAs(OutputType.FILE);
- FileUtils.copyFile(shot, new File("D:\\ shot1.jpg"));
- **moveToElement()** from the Actions class to simulate mouse hover effect.
- Actions actions = new Actions(driver);
- WebElement mouseHover = driver.findElement(By.xpath("//div[@id='mainmenu1']/div"));
- actions.moveToElement(mouseHover);
- actions.perform();
- **dragAndDrop()** from Actions class to drag an element and drop it on another element.
-
- WebElement sourceLocator = driver.findElement(By.xpath("//*[@id='image1']/a"));
- WebElement destinationLocator = driver.findElement(By.xpath("//*[@id='stage']/li"));
- Actions actions=new Actions(driver);
- actions.dragAndDrop(sourceLocator, destinationLocator).build().perform();
- **switchTo() and accept(), dismiss() and sendKeys()** methods from Alert class to switch to popup alerts and handle them.
-
- Alert alert = driver.switchTo().alert();
- alert.sendKeys("This Is Softwaretestinghelp");
- alert.accept()
- **getWindowHandle() and getWindowHandles()** to handle Multiple Windows in Selenium WebDriver.
-
- String handle= driver.getWindowHandle();
- Set<String> handle= driver.getWindowHandles();
- for (String handle : driver.getWindowHandles()){
- driver.switchTo().window(handle);
- }
- **getConnection() from DriverManager to start Database Connection.**
- DriverManager.getConnection(URL, "username", "password")
- **POI to read from the excel files.**
- Workbook workbook = WorkbookFactory.create(new FileInputStream(file));

- Sheet sheet = workbook.getSheetAt(0);
- **Asserts using assertEquals(),assertNotEquals(), assertTrue() and assertFalse() to compare the results.**
- Assert.assertEquals(message, “This text”);
- Assert.assertNotEquals(message, “This text”);
- Assert.assertTrue(result<0);
- Assert.assertFalse(result<0);