[**Selenium Tutorials**](https://www.guru99.com/selenium-tutorial.html)

[**Java programs**](https://docs.google.com/document/d/18vkdcEjOZI0n3jViC9TlNPt3vcJlanVCNuhxKHMhyT8/edit)

* reverse string
* Palindrome
* OddEvenNumber
* Reverse number
* Fibonacci
* Prime number
* random number
* Factorial
* Print pattern
* Ascii
* Sunny number
* Reverse String without inbuilt function
* Swap numbers

Object

Any entity that has state and behaviour is known as an object. For example, a chair, pen, table, keyboard, bike, etc. It can be physical or logical.

Ex: A dog is an object because it has states like color, name, breed, etc. as well as behaviors like wagging the tail, barking, eating, etc.

Class

*Collection of objects* is called class. It is a logical entity. A class can also be defined as a blueprint from which you can create an individual object. Class doesn't consume any space.

### **Abstraction**

Hiding internal details and showing functionality is known as abstraction. For example phone call, we don't know the internal processing.

In Java, we use abstract class and interface to achieve abstraction

### **Encapsulation**

Binding (or wrapping) code and data together into a single unit are known as encapsulation. For example, a capsule, it is wrapped with different medicines.

A java class is the example of encapsulation. Java bean is the fully encapsulated class because all the data members are private here

### **Access Specifier:**

Access specifier or modifier is the access type of the method. It specifies the visibility of the method. Java provides four types of access specifier:

* **Public**: The method is accessible by all classes when we use a public specifier in our application.
* **Private**: When we use a private access specifier, the method is accessible only in the classes in which it is defined.
* **Protected**: When we use protected access specifiers, the method is accessible within the same package or subclasses in a different package.
* **Default**: When we do not use any access specifier in the method declaration, Java uses default access specifier by default. It is visible only from the same package only.

### **Polymorphism**

* Method Overloading

if a [class](https://www.javatpoint.com/object-and-class-in-java) has multiple methods having same name but different in parameters, it is known as Method Overloading

* By changing the no of arguments
* By changing the type of arguments
* By changing the data type
* Method Overriding

If subclass (child class) has the same method as declared in the parent class, it is known as method overriding in Java.

* The method must have the same name as in the parent class
* The method must have the same parameter as in the parent class.
* There must be an IS-A relationship (inheritance)

### **Diamond problem**

### I**nheritance types**

* Single level inheritance where B extends A
* Multi level inheritance where B extends A and class C extends A
* Hierarchical inheritance Where class B extends A, Class C extends class A and Class D extends class A
* Multiple inheritance Class C extends A,B

### **TestNg, structure, annotations**

* @BeforeSuite: The annotated method will be run before all tests in this suite have run.
* @AfterSuite: The annotated method will be run after all tests in this suite have run.
* @BeforeTest: The annotated method will be run before any test method belonging to the classes inside the tag is run.
* @AfterTest: The annotated method will be run after all the test methods belonging to the classes inside the tag have run.
* @BeforeGroups: The list of groups that this configuration method will run before. This method is guaranteed to run shortly before the first test method that belongs to any of these groups is invoked.
* @AfterGroups: The list of groups that this configuration method will run after. This method is guaranteed to run shortly after the last test method that belongs to any of these groups is invoked.
* @BeforeClass: The annotated method will be run before the first test method in the current class is invoked.
* @AfterClass: The annotated method will be run after all the test methods in the current class have been run.
* @BeforeMethod: The annotated method will be run before each test method.
* @AfterMethod: The annotated method will be run after each test method.
* @Test: The annotated method is a part of a test case

public class testNGAnnotations {

@Test

public void testCase1() {

System.out.println("This is the A Normal Test Case");

}

@BeforeMethod

public void beforeMethod() {

System.out.println("This will execute before every Method");

}

@AfterMethod

public void afterMethod() {

System.out.println("This will execute after every Method");

}

@BeforeClass

public void beforeClass() {

System.out.println("This will execute before the Class");

}

@AfterClass

public void afterClass() {

System.out.println("This will execute after the Class");

}

@BeforeTest

public void beforeTest() {

System.out.println("This will execute before the Test");

}

@AfterTest

public void afterTest() {

System.out.println("This will execute after the Test");

}

@BeforeSuite

public void beforeSuite() {

System.out.println("This will execute before the Test Suite");

}

@AfterSuite

public void afterSuite() {

System.out.println("This will execute after the Test Suite");

}

}

This will execute before the Test Suite

This will execute before the Test

This will execute before the Class

This will execute before every Method

This is the A Normal Test Case

This will execute after every Method

This will execute after the Class

This will execute after the Test

PASSED: testCase1

### **Parameters in TestNG**

<https://www.toolsqa.com/testng/testng-parameters/>

<https://www.javatpoint.com/testng-parameters>

### **Pom**

A Project Object Model or POM is the fundamental unit of work in Maven. It is an XML file that contains information about the project and configuration details used by Maven to build the project. It contains default values for most projects. Examples for this is the build directory, which is target; the source directory, which is src/main/java; the test source directory, which is src/test/java; and so on. When executing a task or goal, Maven looks for the POM in the current directory. It reads the POM, gets the needed configuration information, then executes the goal.

Some of the configuration that can be specified in the POM are the project dependencies, the plugins or goals that can be executed, the build profiles, and so on. Other information such as the project version, description, developers, mailing lists and such can also be specified.

### **relative xpath -** <https://www.guru99.com/using-contains-sbiling-ancestor-to-find-element-in-selenium.html>

### **What is Scrum & Agile**

### **Ceremonies of Scrum**

* Backlog grooming (product backlog refinement)
* Sprint planning
* Daily scrum
* Sprint review
* Sprint retrospective

### **Test Case Design techniques**

* Boundary value analysis
* Equivalence partitioning
* Error guessing
* Decision table
* State transition diagram

### **Different exception handling methods in java**

| try | The "try" keyword is used to specify a block where we should place an exception code. It means we can't use try block alone. The try block must be followed by either catch or finally. |
| --- | --- |
| catch | The "catch" block is used to handle the exception. It must be preceded by try block which means we can't use catch block alone. It can be followed by finally block later. |
| finally | The "finally" block is used to execute the necessary code of the program. It is executed whether an exception is handled or not. |
| throw | The "throw" keyword is used to throw an exception. |
| throws | The "throws" keyword is used to declare exceptions. It specifies that there may occur an exception in the method. It doesn't throw an exception. It is always used with method signatures. |

### **Explain about reporting in framework like extent reporting**

### **How failed cases are handled**

Under Test report folder, we find a xml called testng-failed.xml, by running the file we ensure failed test cases are run again

### **action class, and use of action class**

* Actions builder = new Actions(driver);
* Action mouseOverHome = builder.moveToElement(link\_Home).build();
* clickAndHold()
* contextClick()
* dragAndDrop(source, target)
* moveToElement(toElement)
* release()
* sendKeys(onElement, charsequence)

### **Structure of testNG xml file**

<suite>

<Test>

<Classes>

<class>

</class>

</test>

</suite>

### **Take value from the dropdown**

### **What are collections**

### **String manipulation**

* Concatenation
* String length
* indexOf(char)- starts with 0
* charAt(int num) – starts with 1
* compareTo –receive 0 if string is matched
* contains – true /false
* endswith – true or false
* CompareToIgnoreCase - receive 0 if string is matched
* replace(initial,final)
* toLowerCase
* toUpperCase

### **Write code to read data from Excel**

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

FileInputStream fis = new FileInputStream("C:\\Users\\vinay\\Desktop\\testing.xlsx");

XSSFWorkbook wb = new XSSFWorkbook(fis);

XSSFSheet sheet = wb.getSheetAt(0);

XSSFRow row = sheet.getRow(3);

XSSFCell cell = row.getCell(0);

System.out.println(cell);

XSSFRow row1 = sheet.getRow(2);

XSSFCell cell1 = row1.createCell(2);

cell1.setCellValue("WriteData cell2");

FileOutputStream fos = new FileOutputStream("C:\\Users\\vinay\\Desktop\\testing.xlsx");

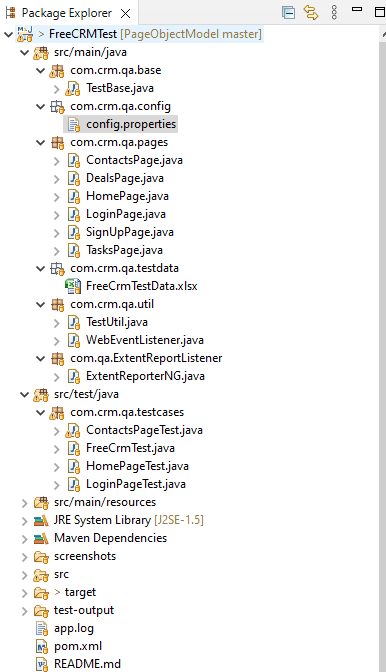
wb.write(fos);

fos.close();

System.out.println(cell1);

}

### **Selenium framework**



Testbase contains initialization

### **assembler & interpreter**

### **uses of Super and this keyword**

### **Encapsulation**

### **what is super class of Java**

The class named **Object** is the super class of every class in Java.

Let’s test it with an example. The java.lang.Class.getSuperclass() returns the Class representing the superclass of the entity (class, interface, primitive type or void) represented by this Class.

So, Create a sample concrete class and lets try to get the name of its super class using this method.

Example

public class Test {

public static void main(String args[]){

Test obj = new Test();

Class cls = obj.getClass().getSuperclass();

System.out.println(cls.getName());

}

}

Output

Since the Object class is the super class of all classes it displays the name of the object class as shown below.

**java.lang.Object**

### **driver.close and driver.quit**

### **How to perform drag and drop**

* Actions builder = new Actions(driver);
* Action dragAndDropHome= builder.dragAndDrop(source, target).build();

### **challenges in automation**

### **Waits**

* Implicitly wait

driver.manage().timeouts().implicitlyWait(15, TimeUnit.SECONDS);

* Explicitly wait

WebDriverWait wait = new WebDriverWait(driver,30);

wait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath("//div[contains(text(),'COMPOSE')]")));

* Fluent Wait

FluentWait wait = new FluentWait(driver);

wait.withTimeout(5000,TimeUnit.MILLISECONDS);

wait.pollingEvery(250,TimeUnit.MILLISECONDS);

wait.ignoring(NoSuchElementException.class)

wait.until(ExpectedConditions.alertIsPresent());

### **How to iterate through array list**

public static void main(String[] args) {

int arr[] = {1,2,3,4,5,6,7,8,9};

for(int i =0;i<arr.length;i++) {

int x = arr[i];

System.out.print(x+ " ");

}

}

### **static method and static class in java**

### **Can we have more than 1 main method**

* Yes we can achieve this using different parameters same like method overloading

class TestOverloading4{

public static void main(String[] args){System.out.println("main with String[]");}

public static void main(String args){System.out.println("main with String");}

public static void main(){System.out.println("main without args");}

‘ }

### **Hashmap**

A HashMap store items in "key/value" pairs, and you can access them by an index of another type like string

HashMap<String, String> capitalCities = new HashMap<String, String>();

capitalCities.put("England", "London");

capitalCities.get("England");

capitalCities.remove("England");

capitalCities.clear(); // to remove all the hashmaps

### **difference between arraylist and hashmap**

| **ArrayList** | **HashMap** |
| --- | --- |
| ArrayList implements the List interface. | HashMap implements the Map interface. |
| ArrayList stores element's value and maintains the indexes for each element. | HashMap stores elements key & value pair. For each value, there must be a key associated with HashMap. |
| ArrayList stores only a single object. | HashMap stores elements in Key and value pairs. |
| We get the element by specifying the index of it in ArrayList. | The elements are being fetched by the corresponding Key in HashMap. |
| The ArrayList maintains the order of the objects they are inserted. | HashMap does not provide a guarantee of the order in which they are inserted. |
| ArrayList allows duplicate elements. | HashMap allows duplicate values but does not allow duplicate keys. |
| ArrayList has any number of null elements. | HashMap allows only one null Key and lots of null values. |
| ArrayList is the index-based data structure supported by the array. | While HashMap is a mapped data structure that works on hashing to obtain stored values. |

### **Difference between hashmap and hashset**

| **Basic** | **HashSet** | **HashMap** |
| --- | --- | --- |
| Implements | Set interface | Map interface |
| Duplicates | No | Yes duplicates values are allowed but no duplicate key is allowed |
| Dummy values | Yes | No |
| Objects required during an add operation | 1 | 2 |
| Adding and storing mechanism | HashMap object | Hashing technique |
| Speed | It is comparatively slower than HashMap | It is comparatively faster than HashSet because of hashing technique has been used here. |
| Null | Have a single null value | Single null key and any number of null values |
| Insertion Method | Add() | Put() |

HashSet<String> hs = new HashSet<String>();

// Adding elements to the HashSet

hs.add("geeks");

hs.add("practice");

hs.add("contribute");

### **Throw and throws**

| Sr. no. | Basis of Differences | throw | throws |
| --- | --- | --- | --- |
| 1. | Definition | Java throw keyword is used throw an exception explicitly in the code, inside the function or the block of code. | Java throws keyword is used in the method signature to declare an exception which might be thrown by the function while the execution of the code. |
| 2. |  | Type of exception Using throw keyword, we can only propagate unchecked exception i.e., the checked exception cannot be propagated using throw only. | Using throws keyword, we can declare both checked and unchecked exceptions. However, the throws keyword can be used to propagate checked exceptions only. |
| 3. | Syntax | The throw keyword is followed by an instance of Exception to be thrown. | The throws keyword is followed by class names of Exceptions to be thrown. |
| 4. | Declaration | throw is used within the method. | throws is used with the method signature. |
| 5. | Internal implementation | We are allowed to throw only one exception at a time i.e. we cannot throw multiple exceptions. | We can declare multiple exceptions using throws keyword that can be thrown by the method. For example, main() throws IOException, SQLException. |

public class TestThrowAndThrows

{

// defining a user-defined method

// which throws ArithmeticException

static void method() **throws** ArithmeticException

{

System.out.println("Inside the method()");

**throw** new ArithmeticException("throwing ArithmeticException");

}

//main method

public static void main(String args[])

{

try

{

method();

}

catch(ArithmeticException e)

{

System.out.println("caught in main() method");

}

}

}

### Hard and soft assertion

### initialise web driver

* System.setProperty("webdriver.chrome.driver", "path of the exe file\\chromedriver.exe");
* WebDriver driver=new ChromeDriver();

OR

* WebDriverManager.chromedriver().setup();
* WebDriverManager.firefoxdriver().setup();
* WebDriverManager.edgedriver().setup();
* WebDriverManager.operadriver().setup();
* WebDriver driver=new ChromeDriver();

### **Defect life cycle**

* New
* Assigned
* Open - Duplicate, Deferred, Won't fix, Invalid
* Fixed
* Retest - Reopen
* Verified

### **SDLC**

* Requirement analysis
* Planning
* Software design such as architectural design
* Software development
* Testing
* Deployment

### **STLC**

* Requirement Analysis
* Test planning
* Test case Development
* Test Environment setup
* Test Execution
* Test cycle Closure

### **FindElement and FindElements**

* WebElement loginLink = driver.findElement(By.linkText("Login"))
* List<WebElement> elements = driver.findElements(By.xpath("//div"));

System.out.println("Number of elements:" +elements.size());

### **Alerts**

* driver.switchTo().alert().dismiss();
* driver.switchTo().alert().accept();
* driver.switchTo().alert().getText();
* driver.switchTo().alert().sendKeys("Text");

### **Tables**

* Static - <https://www.guru99.com/selenium-webtable.html>
* By.xpath("//table/tbody/tr[2]/td[2]")
* Dynamic - <https://demo.guru99.com/test/web-table-element.php>

### **Uses of TestNG**

* **Generate the report in a proper format** including a number of test cases runs, the number of test cases passed, the number of test cases failed, and the number of test cases skipped.
* **Multiple test cases can be grouped more easily by converting them into testng.xml** file. In which you can make priorities which test case should be executed first.
* **The same test case can be executed multiple times without loops** just by using keyword called ‘invocation count.’
* Using testng, **you can execute multiple test cases on multiple browsers**, i.e., cross [browser testing](https://www.guru99.com/top-10-cross-browser-testing-tools.html).
* The TestNG framework **can be easily integrated with tools like Maven, Jenkins, etc**.
* **Annotations used in the testing are very easy to understand** ex: @BeforeMethod, @AfterMethod, @BeforeTest, @AfterTest
* WebDriver has no native mechanism for generating reports. **TestNG can generate the report in a readable format**.
* TestNG simplifies the way the tests are coded. There is no more need for a static main method in our tests. The sequence of actions is regulated by easy-to-understand annotations that do not require methods to be static.

### **bug priority and severity**

### **Locators**

* Id
* Name
* Class
* Css selector
* Xpath
* Linktext
* Partial linktext

### **Why jenkins**

* Running Selenium tests in Jenkins **allows you to run your tests every time your software changes** and deploy the software to a new environment when the tests pass.
* **Jenkins can schedule your tests to run at specific time**.
* **You can save the execution history and Test Reports**.
* **Jenkins supports Maven for building and Testing a project in continuous integration**.

### **how to schedule a job in jenkins**

The letter H, representing the word Hash can be inserted instead of any of the values. It will calculate the parameter based on the hash code of you project name.

This is so that if you are building several projects on your build machine at the same time, let’s say midnight each day, they do not all start their build execution at the same time. Each project starts its execution at a different minute depending on its hash code.

You can also specify the value to be between numbers, i.e. H(0,30) will return the hash code of the project where the possible hashes are 0-30.

Examples:

1. Start build daily at 08:30 in the morning, Monday - Friday: 30 08 \* \* 1-5
2. Weekday daily build twice a day, at lunchtime 12:00 and midnight 00:00, Sunday to Thursday: 00 0,12 \* \* 0-4
3. Start build daily in the late afternoon between 4:00 p.m. - 4:59 p.m. or 16:00 -16:59 depending on the projects hash: H 16 \* \* 1-5
4. Start build at midnight: @midnight or start build at midnight, every Saturday: 59 23 \* \* 6
5. Every first of every month between 2:00 a.m. - 02:30 a.m.: H(0,30) 02 01 \* \*

### **Why Maven**

Selenium WebDriver is great for browser automation. But, when using it for testing and building a test framework, it feels underpowered. Integrating Maven with Selenium provides following benefits

* **Apache Maven provides support for managing the full lifecycle of a test project.**
* **Maven is used to define project structure, dependencies, build, and test management**.
* **Using pom.xml(Maven) you can configure dependencies needed for building testing and running code.**
* **Maven automatically downloads the necessary files from the repository while building the project**.

### **Linked list vs arraylist**

* List<String> al=new ArrayList<String>();
* List<String> al=new LinkedList<String>();

### **Exceptions in Selenium**

* **ElementNotSelectableException**: An element is disabled (can not be clicked/selected) in spite of being present in the DOM
* **ElementNotInteractableException**: An element is not in a state, where it can be interacted with (can not be clicked or able to send keys) in spite of it being present in the DOM
* **ElementNotVisibleException**: In spite of the element being present in the DOM, it is not visible (can not be interactive). For example, elements defined in HTML with type =”hidden”. It is a subclass of the ElementNotInteractableException
* **NoSuchElementException**: Webdriver is not able to determine the elements during runtime, i.e., the FindBy method cannot find a particular component
* **NoSuchFrameException**: Webdriver attempts to switch to an invalid frame, which is unavailable
* **NoAlertPresentException**: Webdriver is trying to switch to an invalid alert, which is unavailable
* **NoSuchWindowException**: Webdriver is trying to switch to an invalid window, which is unavailable
* **StaleElementReferenceException**: The referenced element is no longer present on the DOM page (a reference to a component is now Stale). For example, the item belongs to a different frame than the current one or the user has navigated away to another page
* **SessionNotFoundException**: Webdriver is acting immediately after ‘quitting’ the browser
* **TimeoutException**: The command did not complete in the specified time. For example, the element didn’t display at the specified time. This is especially encountered when working with waits
* **WebDriverException**: Webdriver is acting immediately after ‘closing’ the browser

### **How to handle exceptions in Selenium**

* Try -catch block
* Multiple catch blocks
* Nested try catch blocks
* Throw /throws
* Multiple exceptions like throws and try catch
* Finally
* One can also use the following methods to display Exception Information:

printStackTrace(): It prints the stack trace, name of the exception, and other useful description

toString(): It returns a text message describing the exception name and description

getMessage(): It displays the description of the exception

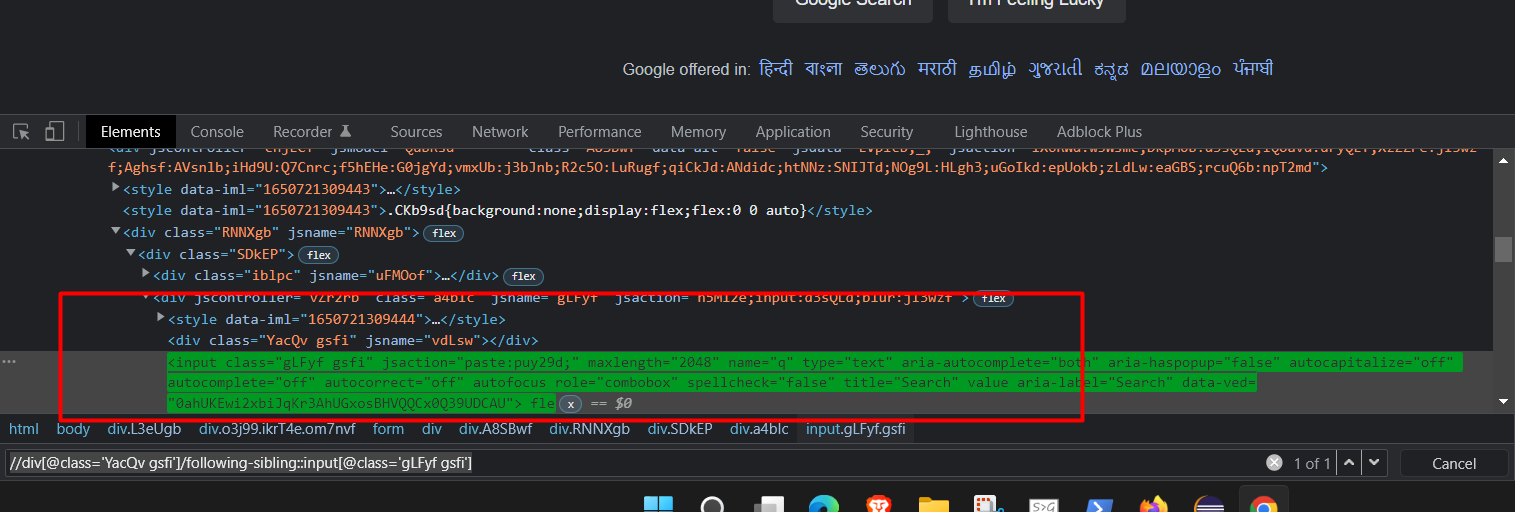
### **Different kinds of Xpaths**

* Contains() [Example from Google.co.in]

//div[contains(text() , 'Google offered in: ')]

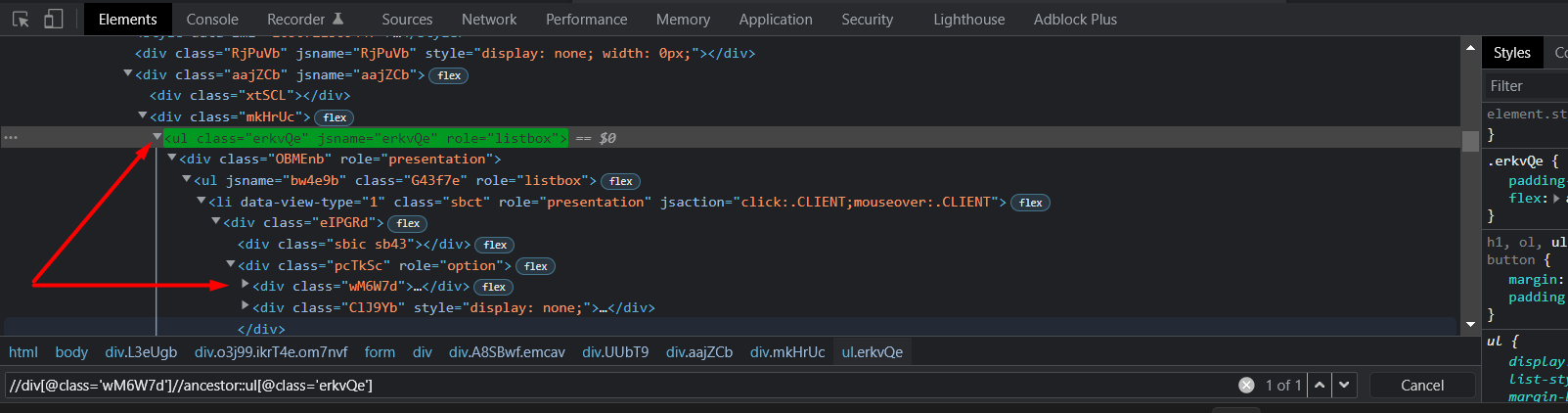
* following-sibling [Example from Google.co.in]

//div[@class='YacQv gsfi']/following-sibling::input[@class='gLFyf gsfi']



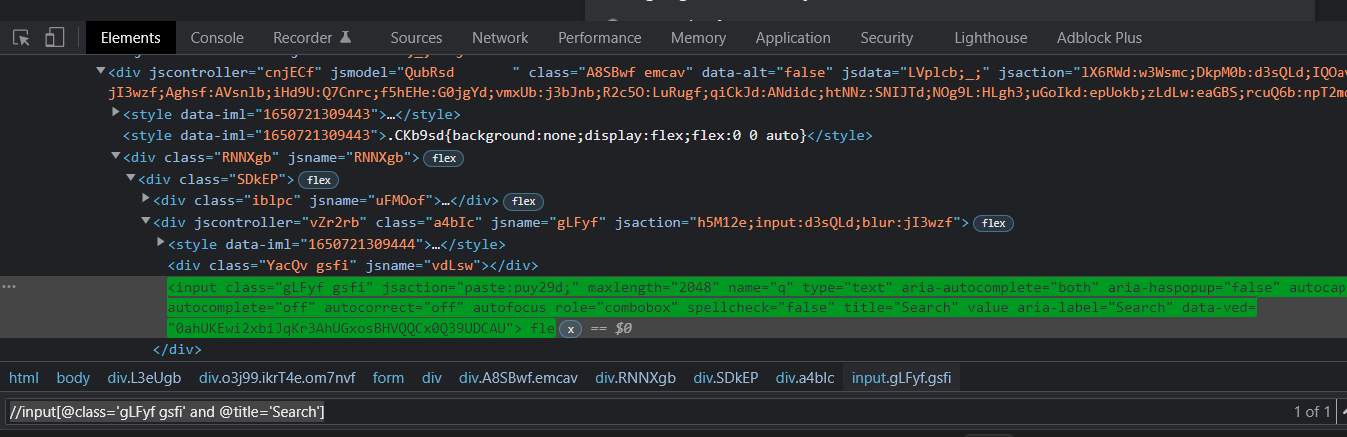
* ancestor [Example from Google.co.in]

//div[@class='wM6W7d']//ancestor::ul[@class='erkvQe']

****

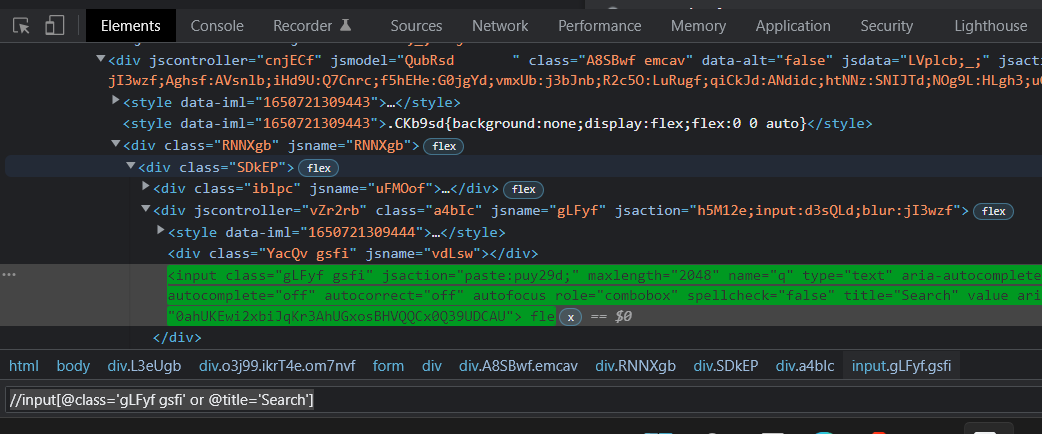
* and [Example from Google.co.in]

//input[@class='gLFyf gsfi' and @title='Search']



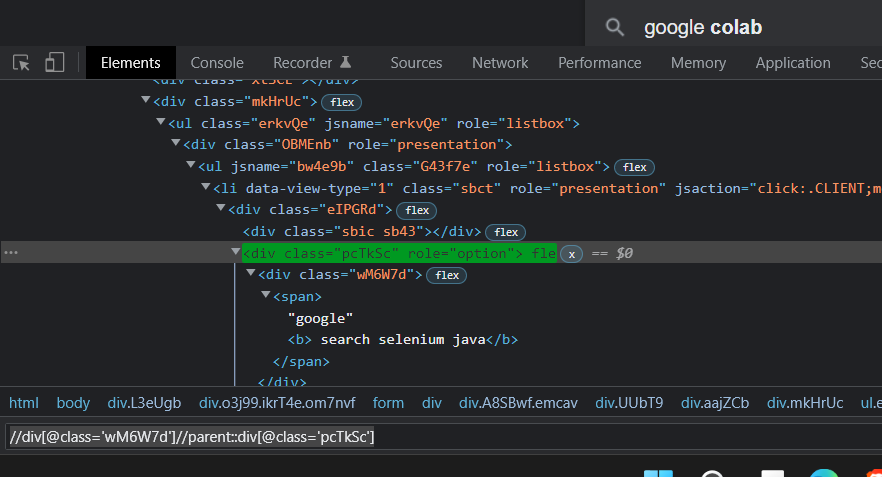
* or [Example from Google.co.in]

//input[@class='gLFyf gsfi' or @title='Search']



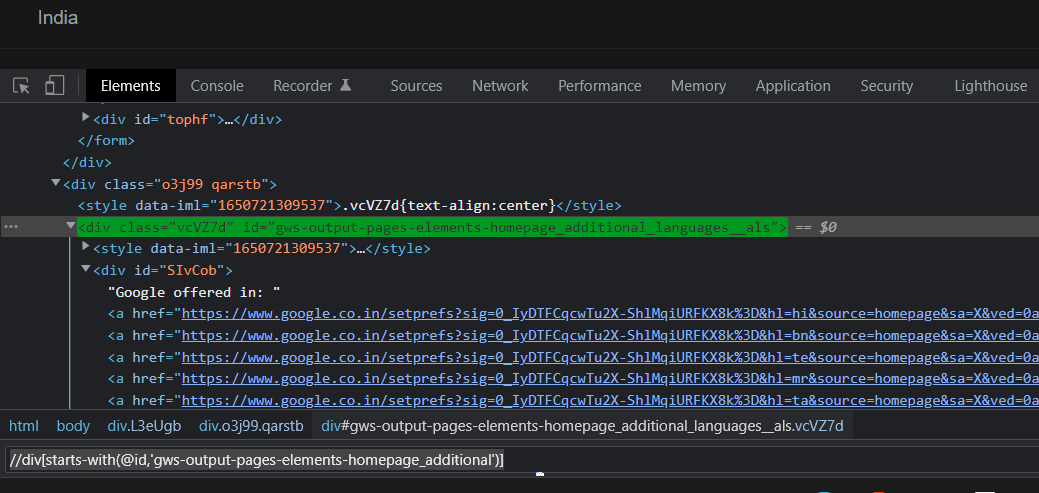
* parent [Example from Google.co.in]

//div[@class='wM6W7d']//parent::div[@class='pcTkSc']



* starts-with [Example from Google.co.in]

//div[starts-with(@id,'gws-output-pages-elements-homepage\_additional')]



### **Deloitte questions**

* Roles and responsibilities in the current project
* How to iterate in hashmap
* Difference between hashmap and hashtable
* How to check database value using selenium
* JDBC connection string code
* How to find Prime number using Java code
* Who will take the final call for priority bugs
* Entry criteria for testing
* Advantages of using maven and Jenkins
* How to schedule a job in Jenkins
* Advantage of pom.xml file
* In the maven project, what is the use of the settings.xml file?
* In the system, where the jar files will be located while using pom.xml
* How to execute the failed test cases again
* How you used listeners in your project
* What are all OS you worked on?
* How to use parameters using TestNG
* Difference between string buffer and string builder
* Difference between Ant and Maven
* When to stop testing
* Difference between interface and abstract
* How to set up the selenium grid and how to check whether the server is running or not
* What is the page factory?
* How to calculate defect density and test coverage
* Tell me about yourself.
* What tools you have used in your project so far?
* Explain the bug life cycle. What priorities do you give while raising a bug?
* What is a defect triage call?
* If in a sprint two user stories have some conflict, what will you do?
* Have you worked on the BDD framework? Why BDD is preferred over other typical frameworks.
* Write a feature file for the login scenario.
* Are you working on the creation of a framework from scratch or you just modify the existing framework?
* What is a test runner file?
* What is a POM.xml file? Why it is used?
* What is a testNG.xml file? What is the benefit of using a testNG framework?
* What is a hybrid framework? What are the components of a framework?
* What are the benefits of creating a framework? Why POM approach is preferred?
* Have you worked on the data-driven framework? WAP to fetch data from an excel file, just tell me the code.
* From what type of files can we obtain test data?
* How to find broken links in selenium?
* What are test listeners in selenium? What are the different types of test listeners?
* What is the frame is selenium? Let’s say I have 3 frames, then how I can go to the 3rd frame from the 1st frame? And how to come back to the 1st frame?
* Have you worked on database testing? How many types of joins are there? What is the use of joins in SQL?
* What is the difference between cross join and inner join?
* Suppose I alter the table, then I dropped the table, then I did a rollback, what can be the output?
* What is a maven in selenium? How maven is useful?
* What is a build life cycle in Maven?
* What are the qualities of a good build too? On what basis you will select a build tool?
* What is a wrapper class in java?
* What is the difference between array and collections in java?
* What is the difference between final, finally, finalize?
* What is the difference between SOAP and REST?
* What are the different methods used in web services?
* Have you worked on GIT? What are the commands in GIT?
* What is the overall check-in checkout procedure in GIT?
* What are the exceptions that you have encountered in selenium?
* What are the different types of testing?
* Tell me the annotations in testNG by order of execution?
* What are the pop-ups in selenium? How to handle windows based pop-ups?
* Can we handle user sessions in selenium?
* How to capture screenshots in selenium? Tell me the code.
* Do you know any scripting language like VB script? Javascript? Can you automate web services using rest assured?
* How do you decide which test cases to automate?
* Tell me the test case for atm – debit card transactions
* What exactly Webdriver driver?
* When you log into a banking application which test case do you automate? Why don’t you automate transaction test cases as you said you automate only login test cases?
* Tell me about the defect life cycle.
* Tell me about your project.
* What are the entry criteria for start testing? What are the exit criteria for testing?
* Do you mark test cases as passed or failed in QC?
* Where do you execute your test case?
* Explain the Automation Framework architecture.
* What are priority and severity? Who decides to give priority?
* In the current project, what’s your role? Manual or Automation or both?
* What are the details you give while logging a defect?  
  Ans: severity, detected by, assigned to, description, artifacts like screenshots.
* What is Maven
* Who will give priority? Does priority affect the application?  
  Ans: Not always.
* When we do Regression Testing?
* When do you do smoke testing?
* Give me the minimal test scenario for a text field with a range of 0-100 digits range?  
  Ans: Boundary Value Analysis.
* Do you know SQL? Tell me about the types of joins.
* What is an inner join?
* When we use the having clause?
* How do you set up an eclipse?
* What does the POM class have? What is it’s used?
* What does POM? Does XML contain?
* How to read data from excel? Write the code
* What are HSSF and XSSF? When do you use them w.r.t xlsx and xlx
* How do you read more rows in excel?
* What is the jar file used to get data from Excel?
* How do you set a path for Chrome, Firefox, and IE driver