Selenium Java Interview Questions and Answers Part-3

1) How to scroll web page up and down using Selenium WebDriver?

To scroll using Selenium, you can use JavaScriptExecutor interface that helps to execute JavaScript methods through Selenium Webdriver.

```
JavascriptExecutor js = (JavascriptExecutor) driver;
//This will scroll the page till the element is found
js.executeScript("arguments[0].scrollIntoView();", Element);
```

2) How to perform right click (Context Click) action in Selenium WebDriver?

We can use Action class to provide various important methods to simulate user actions

```
//Instantiate Action Class
Actions actions = new Actions(driver);
//Retrieve WebElement to perform right click
WebElement btnElement = driver.findElement(By.id("rightClickBtn"));
//Right Click the button to display Context Menu
actions.contextClick(btnElement).perform();
```

3) How to perform double click action in Selenium WebDriver?

Action class method doubleClick(WebElement) is required to be used to perform this user action.

```
//Instantiate Action Class
Actions actions = new Actions(driver);
//Retrieve WebElement to perform double click WebElement
btnElement = driver.findElement(By.id("doubleClickBtn"));
//Double Click the button
actions.doubleClick(btnElement).perform();
```

4) How to perform drag and drop action in Selenium WebDriver?

```
//Actions class method to drag and drop
Actions builder = new Actions(driver);
WebElement from = driver.findElement(By.id("draggable"));
WebElement to = driver.findElement(By.id("droppable"));
//Perform drag and drop
builder.dragAndDrop(from, to).perform();
```

5) How to highlight elements using Selenium WebDriver?

```
// Create the  JavascriptExecutor object
JavascriptExecutor js=(JavascriptExecutor)driver;
// find element using id attribute
WebElement username= driver.findElement(By.id("email"));
// call the executeScript method
js.executeScript("arguments[0].setAttribute('style,'border: solid 2px red'');", username);
```

6) Have you used any cross browser testing tool to run Selenium Scripts on cloud?

Below tools can be used to run selenium scripts on cloud:

-SauceLabs

-CrossBrowserTesting

7) What are the DesiredCapabitlies in Selenium WebDriver and their use?

The Desired Capabilities Class helps us to tell the webdriver, which environment we are going to use in our test script.

The setCapability method of the DesiredCapabilities Class, can be used in Selenium Grid. It is used to perform a parallel execution on different machine configurations. It is used to set the browser properties (Ex. Chrome, IE), Platform Name (Ex. Linux, Windows) that are used while executing the test cases.

8) What is Continuous Integration?

Continuous Integration (CI) is a development practice that requires developers to integrate code into a shared repository several times a day. Each check-in is then verified by an automated build, allowing teams to detect problems early.

9) How to achieve database testing in Selenium?

```
//Make a connection to the database
Connection con = DriverManager.getConnection(dbUrl,username,password);
//load the JDBC Driver using the code
Class.forName("com.mysql.jdbc.Driver");
//send queries to the database
Statement stmt = con.createStatement();
//Once the statement object is created use the executeQuery method to execute the SQL queri
stmt.executeQuery(select *  from employee;);
//Results from the executed query are stored in the ResultSet Object. While loop to iterate
while (rs.next()){
String myName = rs.getString(1);}
//close the db connection
con.close();
```

10) What is TestNG?

TestNG is a testing framework inspired from JUnit and NUnit, but introducing some new functionalities that make it more powerful and easier to use. TestNG is an open source automated testing framework; where NG means NextGeneration.

11) What are Annotations and what are the different annotations available in TestNG?

Annotations in TestNG are lines of code that can control how the method below them will be executed. They are always preceded by the @ symbol.

Here is the list of annotations that TestNG supports −

-@BeforeSuite: The annotated method will be run only once before all tests in this suite have run.

-@AfterSuite: The annotated method will be run only once after all tests in this suite have run.

-@BeforeClass: The annotated method will be run only once before the first test method in the current class is invoked.

-@AfterClass: The annotated method will be run only once after all the test methods in the current class have run.

-@BeforeTest: The annotated method will be run before any test method belonging to the classes inside the <test> tag is run.

-@AfterTest: The annotated method will be run after all the test methods belonging to the classes inside the <test> 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.

-@BeforeMethod: The annotated method will be run before each test method.

-@AfterMethod: The annotated method will be run after each test method.

-@DataProvider: Marks a method as supplying data for a test method. The annotated method must return an Object[ ][ ], where each Object[ ] can be assigned the parameter list of the test method. The @Test method that wants to receive data from this DataProvider needs to use a dataProvider name equals to the name of this annotation.

-@Factory: Marks a method as a factory that returns objects that will be used by TestNG as Test classes. The method must return Object[ ].

-@Listeners: Defines listeners on a test class.

-@Parameters: Describes how to pass parameters to a @Test method.

-@Test: Marks a class or a method as a part of the test

12) What is TestNG Assert and list out some common assertions supported by TestNG?

Asserts helps us to verify the conditions of the test and decide whether test has failed or passed. A test is considered successful ONLY if it is completed without throwing any exception.

Some of the common assertions are:

-assertEqual

-assertTrue

-assertFalse

13) How to create and run TestNG.xml?

Step 1: Add a new file to the project with name as testng.xml

Step 2: Add below given code in testng.xml

```
<suite name="TestSuite">
<test name="Test1">
<classes>
<class name="TestClass" />
</classes>
</test>
</suite>
```

Step 3: Run the test by right click on the testng xml file and select Run As > TestNG Suite

14) How to set test case priority in TestNG?

We need to use the 'priority' parameter, if we want the methods to be executed in specific order. TestNG will execute the @Test annotation with the lowest priority value up to the largest.

```
@Test(priority = 0)
public void One() {
  System.out.println("This is the Test Case number One");
  }
@Test(priority = 1)
public void Two() {
System.out.println("This is the Test Case number Two");
}
```

15) What is parameterized testing in TestNG?

To pass multiple data to the application at runtime, we need to parameterize our test scripts.

There are two ways by which we can achieve parameterization in TestNG:

With the help of Parameters annotation and TestNG XML file.

```
@Parameters({"name","searchKey"})
```

With the help of DataProvider annotation.

```
@DataProvider(name="SearchProvider")
```

16) How to run a group of test cases using TestNG?

Groups is one more annotation of TestNG which can be used in the execution of multiple tests.

```
public class Grouping{
@Test (groups = { "g1" })
public void test1() {
System.out.println("This is group 1");
}
@Test (groups = { "g2" })
public void test2() {
System.out.println("This is group 2");
}}
```

Create a testing xml file like this:

```
<suite name ="Suite">
<test name = "Grouping">
<groups>
<run>
<include name="g1">
</run>
</groups>
<classes>
<class name="Grouping">
</classes>
</test>
</suite>
```

17) What is the use of @Listener annotation in TestNG?

Listener is defined as interface that modifies the default TestNG's behaviour. As the name suggests Listeners "listen" to the event defined in the selenium script and behave accordingly. It is used in selenium by implementing Listeners Interface. It allows customizing TestNG reports or logs. There are many types of TestNG listeners available:

-IAnnotationTransformer

-IAnnotationTransformer2

-IConfigurable

-IConfigurationListener

-IExecutionListener

-IHookable

-IInvokedMethodListener

-IInvokedMethodListener2

-IMethodInterceptor

-IReporter

-ISuiteListener

-ITestListener

18) How can we create a data driven framework using TestNG?

We can create data driven tests by using the DataProvider feature.

```
public class DataProviderTest {
private static WebDriver driver;
@DataProvider(name = "Authentication")
public static Object[][] credentials() {
return new Object[][] { { "testuser_1", "Test@123" }, { "testuser_2”, "Test@123" }};
}
// Here we are calling the Data Provider object with its Name
@Test(dataProvider = "Authentication")
public void test(String sUsername, String sPassword) {
driver = new FirefoxDriver();
driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
driver.get("http://www.testqa.com");
driver.findElement(By.xpath(".//*[@id='account']/a")).click();
driver.findElement(By.id("log")).sendKeys(sUsername);
driver.findElement(By.id("pwd")).sendKeys(sPassword);
driver.findElement(By.id("login")).click();
driver.findElement(By.xpath(".//*[@id='account_logout']/a")).click();
driver.quit();
}
}
```

19) Where you have applied OOPS in Automation Framework?

Abstraction – 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.

Interface – WebDriver itself is an Interface. So based on the above statement WebDriver driver = new FirefoxDriver(); we are initializing Firefox browser using Selenium WebDriver. It means we are creating a reference variable (driver) of the interface (WebDriver) and creating an Object.

Inheritance – We create a Base Class in the Framework to initialize WebDriver interface, WebDriver waits, Property files, Excels, etc. We extend the Base Class in other classes such as Tests and Utility Class. Extending one class into other class is known as Inheritance.

Polymorphism – We use implicit wait in Selenium. Implicit wait is an example of overloading. In Implicit wait we use different time stamps such as SECONDS, MINUTES, HOURS etc.

Encapsulation – All the classes in a framework are an example of Encapsulation. In POM classes, we declare the data members using @FindBy and initialization of data members will be done using Constructor to utilize those in methods.

20) How to handle Chrome Browser notifications in Selenium?

```
// Create object of HashMap Class
Map<String, Object> prefs = new HashMap<String, Object>();
// Set the notification setting it will override the default setting
prefs.put("profile.default_content_setting_values.notifications", 2);
// Create object of ChromeOption class
ChromeOptions options = new ChromeOptions();
// Set the experimental option
options.setExperimentalOption("prefs", prefs);
// pass the options object in Chrome driver
WebDriver driver = new ChromeDriver(options);
```

21) Explain any Test Automation Framework?

Testing frameworks are an essential part of any successful automated testing process. They can reduce maintenance costs and testing efforts and will provide a higher return on investment (ROI) for QA teams looking to optimize their agile processes. A testing framework is a set of guidelines or rules used for creating and designing test cases. A framework is comprised of a combination of practices and tools that are designed to help QA professionals test more efficiently. These guidelines could include coding standards, test-data handling methods, object repositories, processes for storing test results, or information on how to access external resources.

22) Tell some popular Test Automation Frameworks?

Some of the popular test automation frameworks are:

-DataDriven

-KeywordDriven

-Hybrid

-Page Object Model

23) Why Framework?

Below are advantages of using an automation framework:

-Ease of scripting: With multiple Testers in a team, having an automation framework in place ensures consistent coding and that best practices are followed to a certain level. Standard scripting will result in team consistency during test library design and prevent individuals from following their own coding standards, thus avoiding duplicate coding.

-Scalable: Whether multiple web pages are being added or Objects or data, a good automation framework design is scalable when the need arises. A framework should be much easier to extend to larger projects.

-Modularity: Modularity allows testers to re-use common modules in different scripts to avoid unnecessary & redundant tasks.

-Easy to understand: Having an automation framework in place it is quick to transition (or understand) the overall architecture & bring people up-to-speed.

-Reusability: Common library files can be reused when required, no need to develop them every time.

-Cost & Maintenance: A well designed automation framework helps in maintaining the code in light of common changes like Test data, Page Objects, Reporting structure, etc.

-Maximum Coverage: A framework allows us to maintain a good range of Test data, i.e. coverage in turn.

-Better error handling: A good automation framework helps us catch different recovery scenarios and handle them properly.

-Minimal manual intervention: You need not input test data or run test scripts manually and then keep monitoring the execution.

-Easy Reporting: The reporting module within framework can handle all the report requirements.

-Segregation: A framework helps segregate the test script logic and the corresponding test data. The Test data can be stored into an external database like property files, xml files, excel files, text files, CSV files, ODBC repositories etc.

-Test configuration: Test suites covering different application modules can be configured easily using an automation framework.

-Continuous integration: An automation framework helps in integration of automation code with different CI tools.

-Debugging: Easy to debug errors

24) Which Test Automation Framework you are using and why?

Cucumber Selenium Framework has now a days become very popular test automation framework in the industry and many companies are using it because its easy to involve business stakeholders and easy to maintain.

25) Mention the name of the Framework which you are using currently in your project, explain it in details along with its benefits?

Framework consists of the following tools:

Selenium, Eclipse IDE,Junit, Maven, Cucumber

File Formats Used in the Framework:

-Properties file: We use properties file to store and retrieve different application and framework related configuration

-Excel files: Excel files are used to pass multiple sets of data to the application.

Following are the key components of the framework:

-PageObject : It consists of all different page classes with their objects and methods

-TestData: It stores the data files, Script reads test data from external data sources and executes test based on it

-Features: It consists of functional test cases in the form of cucumber feature files written in gherkin format

-StepDefinitions: It consists of different methods to implement each step of your feature files

-TestRunner: It is the starting point for Junit to start executing your tests

-Utilities: It consists of different reusable framework methods to perform different operations

-Reports: It consists of different test reports in different formats along with screenshots

-Pom xml: It consists of all different project dependencies and plugins