NTT Data Selenium Interview Questions with Answers (2026) | Saravana Tech

81 interview questions with explanations, examples, and practical code

Note: Personalize sample answers about your project, role, team, repository, reporting, and tools to match your actual experience. Code examples assume Java with Selenium 4 and TestNG unless otherwise stated.

1. Introduce yourself

Use a 60–90 second structure: years of experience, manual/automation skills, tools, responsibilities, one strength or achievement, and the role you want. Sample: “I have around 13 years of experience in software testing across manual and automation testing. My automation experience includes Selenium with Java, TestNG, Cucumber, API automation, and Playwright. I contribute to framework maintenance, test planning, execution analysis, defect investigation, and coordination with developers and business teams. I’m interested in a role where I can combine hands-on automation with quality ownership.” Personalize this with only tools and responsibilities you can discuss confidently.

2. Explain your framework

Sample hybrid framework: TestNG/Cucumber tests → test classes or step definitions → Page Object Model page classes → reusable utilities/configuration/test data → Selenium WebDriver → browser → assertions, logs, screenshots, reports, and CI. Explain why each layer exists: page objects isolate locators/actions; utilities centralize waits and common functions; data/configuration keep environment-specific values out of test logic; reports help diagnose failures.

```
TestNG tests / Cucumber feature files
               |
               v
Test classes / Step Definitions
               |
               v
Page Objects (locators + page actions)
               |
               v
Utilities + Config + Test Data
               |
               v
Selenium WebDriver -> Browser -> Application
               |
               v
Assertions + Logs + Screenshots + Reports + CI
```

Adjust the diagram to your actual framework; do not claim Cucumber, Jenkins, or a particular reporting tool unless your project uses it.

3. What is your role in your project?

Describe your actual responsibilities in terms of test planning, scenario design, automation development, framework maintenance, code reviews, execution analysis, defect triage, reporting, mentoring, and stakeholder coordination. Separate tasks you own from tasks you support.

4. What is a constructor?

A constructor initializes an object when it is created. It has the same name as the class and no return type. Constructors can be overloaded. If no constructor is declared, Java may provide a default no-argument constructor.

```
class LoginPage {
    private WebDriver driver;
    public LoginPage(WebDriver driver) {
        this.driver = driver;
    }
}
LoginPage page = new LoginPage(driver);
```

5. Difference between final and finally

- final is a keyword used with variables, methods, and classes: a final variable cannot be reassigned; a final method cannot be overridden; a final class cannot be subclassed.
- finally is a block used with try/catch to run cleanup code whether an exception occurs or not, with a few exceptional termination cases.

```
final int timeoutSeconds = 10;
try {
    // risky operation
} catch (RuntimeException e) {
    // handle or log
} finally {
    // cleanup, e.g. close resources
}
```

6. How do you create an object?

Use the new keyword and a constructor for a typical class instance. Other mechanisms exist (factory methods, dependency injection, reflection), but new is the standard interview example.

```
Employee employee = new Employee();
LoginPage page = new LoginPage(driver);
```

7. How do you double-click?

```
WebElement element = driver.findElement(By.id("record"));
new Actions(driver).doubleClick(element).perform();
```

Ensure the element is visible and interactable. Use explicit waits if the UI loads asynchronously.

8. How do you mouse-hover?

```
WebElement menu = driver.findElement(By.id("products"));
new Actions(driver).moveToElement(menu).perform();
WebElement submenu = driver.findElement(By.id("reports"));
submenu.click();
```

9. How do you handle a popup?

Identify the type first: JavaScript alert/confirm/prompt, new browser tab/window, HTML modal, or operating-system dialog. Use Alert API for JavaScript alerts, window handles for new tabs/windows, normal locators for HTML modals, and a native tool only when a real OS dialog cannot be handled through WebDriver.

10. How do you handle alerts?

```
Alert alert = new WebDriverWait(driver, Duration.ofSeconds(5))
    .until(ExpectedConditions.alertIsPresent());
String text = alert.getText();
alert.accept();       // OK
// alert.dismiss();   // Cancel
// alert.sendKeys("input"); // prompt dialog
```

11. How do you handle windows?

```
String parent = driver.getWindowHandle();
driver.findElement(By.id("openWindow")).click();
new WebDriverWait(driver, Duration.ofSeconds(10))
    .until(d -> d.getWindowHandles().size() > 1);

for (String handle : driver.getWindowHandles()) {
    if (!handle.equals(parent)) {
        driver.switchTo().window(handle);
        System.out.println(driver.getTitle());
        break;
    }
}
driver.switchTo().window(parent);
```

Do not assume the popup is always the second handle; compare handles or identify the window by title, URL, or content.

12. How do you run your scripts?

Typical Maven/TestNG flow: run from IDE for debugging, run a TestNG XML suite or Maven command locally, then execute through CI such as Jenkins. Example: mvn clean test. Use suite XML, groups, or Maven parameters to select tests. Mention the actual workflow used by your team.

13. Run method B only if method A passes using TestNG

```
@Test
public void methodA() {
    Assert.assertEquals(2 + 2, 4);
}

@Test(dependsOnMethods = "methodA")
public void methodB() {
    System.out.println("Runs only if methodA passes");
}
```

By default, TestNG skips a dependent test if its dependency fails or is skipped. Dependencies should be used when the tests genuinely depend on each other; independent tests are usually more robust.

14. What is Maven?

Maven is a Java build and dependency-management tool. The pom.xml declares dependencies, plugins, project metadata, and build configuration. Common commands include mvn clean, mvn compile, mvn test, and mvn clean test. Maven can run TestNG tests through a configured test plugin.

15. What is Jenkins? Where does it run—client or server?

Jenkins is an automation server used for CI/CD. The controller coordinates jobs and agents/workers execute builds and tests. It typically runs as a server process on a machine, VM, container, or cloud environment; a browser user interface is the client side. A test can run on the controller or, preferably in many setups, on an agent with Java, Maven, browser, and required configuration.

16. Git and GitHub

Git is a distributed version-control system used to track code changes, branches, commits, and merges. GitHub is a hosted platform for Git repositories, pull requests, code review, issues, and collaboration. Git can be used without GitHub; GitHub hosts and adds collaboration workflows.

17. How do you pull and push using Git?

```
git status
git switch -c feature/login-tests
git add .
git commit -m "Add login automation tests"
git pull --rebase origin main
git push -u origin feature/login-tests
```

Typical sequence: inspect status, create/switch branch, stage, commit, pull/rebase latest changes when appropriate, then push. Team policies may require pull requests and approvals. Resolve conflicts, run tests, and re-check status before pushing.

18. Run only 20 out of 100 test cases using TestNG

Common methods: use groups and include the desired group in testng.xml, list only the required classes/methods in the suite, or use a controlled test-selection mechanism. Example group selection:

```
@Test(groups = {"smoke"})
public void loginTest() { }

<suite name="Smoke">
  <test name="SelectedTests">
    <groups>
      <run><include name="smoke"/></run>
    </groups>
    <packages><package name="tests.*"/></packages>
  </test>
</suite>
```

Tag exactly the intended 20 tests as smoke (or another group). Groups select by category, not automatically by count.

19. Agile methodology

Agile delivers working software incrementally, encourages collaboration and feedback, and adapts to changing requirements. In Scrum, the team refines a backlog, plans a sprint, holds daily Scrum, delivers an increment, reviews it, and reflects in a retrospective. QA participates throughout development.

20. Collections

The Java Collections Framework provides structures to store and process groups of objects. List allows duplicates and preserves order; Set stores unique values; Map stores key-value pairs; Queue supports processing in order. Selenium examples include List<WebElement> for table rows, Set<String> for window handles, and Map<String,String> for test data.

21. How to pass values in Selenium other than sendKeys?

Options depend on the control. For normal text inputs, sendKeys() is preferred. Alternatives include JavaScript setting the value (a fallback), selecting an option with Select for native dropdowns, Actions for keyboard input, or application APIs/test-data setup where appropriate.

```
WebElement input = driver.findElement(By.id("email"));
((JavascriptExecutor) driver).executeScript(
    "arguments[0].value = arguments[1];",
    input, "qa@example.com");
```

JavaScript value assignment may not trigger the same input/change events as real typing and can bypass application behavior, so validate the resulting UI state and prefer normal user-like interactions.

22. How do you get text from an alert?

```
Alert alert = driver.switchTo().alert();
String message = alert.getText();
```

If the alert appears asynchronously, wait for ExpectedConditions.alertIsPresent() first.

23. How do you pass a value to an alert?

```
Alert alert = driver.switchTo().alert();
alert.sendKeys("Vajravel");
alert.accept();
```

sendKeys() is supported for JavaScript prompt dialogs, not ordinary alert or confirm dialogs.

24. Absolute vs relative XPath

- Absolute XPath starts from the document root and describes a long path, for example /html/body/div[1]/form/input[1]. It is brittle when the DOM structure changes.
- Relative XPath starts from anywhere using // and usually targets stable attributes or text, for example //input[@id='email']. It is generally preferred for maintainability.

```
driver.findElement(By.xpath("/html/body/div[1]/form/input[1]"));
driver.findElement(By.xpath("//input[@id='email']"));
```

25. How do you handle dynamic objects?

Prefer stable attributes, data-testid attributes, partial matching, parent/child relationships, and explicit waits. Avoid brittle positional locators unless needed.

```
By saveButton = By.xpath(
    "//button[contains(@class,'save') and normalize-space()='Save']");
WebElement save = new WebDriverWait(driver, Duration.ofSeconds(10))
    .until(ExpectedConditions.elementToBeClickable(saveButton));
save.click();
```

For changing IDs, use a stable prefix/suffix only if it is unique, e.g. starts-with(@id,'user_'). Re-check uniqueness with findElements().

26. Assert vs Verify

An assert usually means a hard assertion: failure stops the current test method by throwing an assertion error. “Verify” commonly means a soft assertion or non-blocking check that records failures and continues; it is not a universal Selenium method.

```
Assert.assertEquals(actualTitle, expectedTitle); // hard assertion

SoftAssert soft = new SoftAssert();
soft.assertTrue(isHeaderVisible, "Header missing");
soft.assertEquals(actualText, expectedText, "Text mismatch");
soft.assertAll(); // report collected failures
```

With TestNG SoftAssert, call assertAll() or collected failures may not fail the test.

27. Drag and drop

```
WebElement source = driver.findElement(By.id("drag"));
WebElement target = driver.findElement(By.id("drop"));
new Actions(driver).dragAndDrop(source, target).perform();
```

For custom drag/drop components, a sequence of clickAndHold, moveToElement, and release may work better.

28. TestNG annotations

Common annotations: @BeforeSuite/@AfterSuite, @BeforeTest/@AfterTest, @BeforeClass/@AfterClass, @BeforeMethod/@AfterMethod, @BeforeGroups/@AfterGroups, and @Test. @DataProvider supplies data; @Parameters reads XML parameters; @Listeners registers listeners.

29. Listeners

TestNG listeners receive lifecycle events, such as test start, success, failure, and skip. They are used for reports, screenshots, logging, and custom handling. Implement ITestListener and register it with @Listeners or suite XML.

```
public class TestListener implements ITestListener {
    @Override
    public void onTestFailure(ITestResult result) {
        System.out.println("Failed: " + result.getName());
        // Capture screenshot/logs if driver context is available
    }
}
```

30. JDBC query

JDBC connects Java to a relational database. Keep credentials outside source code and use try-with-resources.

```
String url = "jdbc:postgresql://localhost:5432/testdb";
String user = System.getenv("DB_USER");
String password = System.getenv("DB_PASSWORD");

try (Connection con = DriverManager.getConnection(url, user, password);
     PreparedStatement ps = con.prepareStatement(
         "SELECT status FROM test_runs WHERE run_id = ?")) {
    ps.setInt(1, 1001);
    try (ResultSet rs = ps.executeQuery()) {
        while (rs.next()) {
            System.out.println(rs.getString("status"));
        }
    }
}
```

Add the appropriate JDBC driver dependency and change URL/query for your database. Use PreparedStatement for parameterized values.

31. Download/upload files using Selenium

Upload: locate input[type=file] and sendKeys(fullPath). Download: configure the browser download directory through browser options/preferences, trigger the download, and verify the file exists and is complete. Selenium does not provide a universal download-file API for every browser.

```
// Upload
driver.findElement(By.cssSelector("input[type='file']"))
      .sendKeys("C:\\test-data\\sample.pdf");

// Download: configure ChromeOptions prefs before creating ChromeDriver
Map<String, Object> prefs = new HashMap<>();
prefs.put("download.default_directory", "C:\\downloads");
ChromeOptions options = new ChromeOptions();
options.setExperimentalOption("prefs", prefs);
WebDriver driver = new ChromeDriver(options);
```

32. Screenshot

```
File source = ((TakesScreenshot) driver)
    .getScreenshotAs(OutputType.FILE);
Files.copy(source.toPath(),
    Paths.get("target/screenshots/failure.png"),
    StandardCopyOption.REPLACE_EXISTING);
```

Use a unique file name for each test and capture screenshots on failure via a TestNG listener or teardown hook. Import java.nio.file.Files, Paths, StandardCopyOption, and Selenium screenshot classes.

33. Do you write scripts for functionality in the same script?

In a maintainable framework, test logic is separate from page interactions. A test method expresses the scenario; Page Objects contain page locators and actions; utilities handle common waits/configuration/data. Avoid duplicating login, navigation, and common actions in every test. Small one-off checks may be simpler, but regression frameworks benefit from reuse.

34. How do you pass parameters in TestNG?

```
// testng.xml
<suite name="Suite">
  <test name="Login">
    <parameter name="browser" value="chrome"/>
    <classes><class name="tests.LoginTest"/></classes>
  </test>
</suite>

// Java
@Parameters("browser")
@BeforeMethod
public void setup(String browser) {
    if (browser.equalsIgnoreCase("chrome")) driver = new ChromeDriver();
    else if (browser.equalsIgnoreCase("firefox")) driver = new FirefoxDriver();
}
```

For multiple data rows, use @DataProvider instead of XML @Parameters.

35. Synchronization

Synchronization means coordinating test execution with application readiness. Selenium strategies include implicit waits, explicit waits, FluentWait, and application-specific conditions. Prefer explicit waits for dynamic elements.

```
WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(10));
wait.until(ExpectedConditions.visibilityOfElementLocated(By.id("result")));
```

36. Class A has method1 and Class B has method2; how to call method1 from Class A?

If method1 is an instance method, create/use an instance of Class A. If it is static, call it through the class name. Avoid circular dependencies and design classes around responsibilities.

```
class A {
    public void method1() { System.out.println("A.method1"); }
}
class B {
    public void method2() {
        A a = new A();
        a.method1();
    }
}
```

37. How do you get coordinates?

```
WebElement element = driver.findElement(By.id("button"));
Point location = element.getLocation();
Dimension size = element.getSize();
System.out.println("x=" + location.getX() + ", y=" + location.getY());
System.out.println("width=" + size.getWidth() + ", height=" + size.getHeight());
```

Use getRect() to obtain location and size together. Browser viewport coordinates can differ from screen coordinates.

38. Frames

```
driver.switchTo().frame("frameName"); // name or id
// or driver.switchTo().frame(0);
// or driver.switchTo().frame(frameElement);
driver.findElement(By.id("insideFrame")).click();
driver.switchTo().defaultContent();
```

Use parentFrame() to move up one frame level; defaultContent() returns to the top-level document.

39. Navigate back and forward

```
driver.navigate().back();
driver.navigate().forward();
driver.navigate().refresh();
driver.navigate().to("https://example.com");
```

Use explicit waits after navigation if the next action depends on asynchronous page content.

40. Keyword-driven framework

A keyword-driven framework maps readable action keywords (CLICK, TYPE, SELECT, VERIFY_TEXT) to reusable functions. Test steps/data define which keyword to run and with what target/value; a dispatcher calls the corresponding implementation. Keep keywords typed/validated and report failures with context. A data-driven framework instead focuses on running the same test logic with multiple input datasets; a hybrid can combine both.

41. Shared repository or single repository?

Answer based on the project. A shared Git repository is common for a team so everyone branches from a common codebase and contributes through pull requests. A monorepo contains multiple components in one repository; multiple repositories separate projects/services. Explain branch strategy, code review, and how conflicts are handled.

42. What exceptions do you generally use?

Distinguish exceptions you catch from Selenium exceptions you diagnose. Common examples: NoSuchElementException, StaleElementReferenceException, TimeoutException, ElementClickInterceptedException, ElementNotInteractableException, InvalidSelectorException, NoSuchFrameException, and SessionNotCreatedException. Catch exceptions only when you can recover or add useful context; otherwise log and rethrow.

43. Method signature

In Java, a method signature is the method name plus the parameter types (and their order). The return type is not part of the signature, and parameter names are not part of it.

```
void login(String user) { }
void login(String user, String password) { } // overload
// Cannot overload only by changing return type:
 // int login(String user) { } // conflicts with first method
```

44. Introduce yourself

Same core answer as Question 1. Avoid memorized claims; align the skills and achievements with the job description and be ready with a real example for every key skill.

45. Explain Agile-Scrum process

Backlog refinement clarifies stories and acceptance criteria; sprint planning selects work and defines a goal; Daily Scrum coordinates progress and blockers; development and testing occur during the sprint; Sprint Review demonstrates the increment; Retrospective identifies process improvements. The Product Owner manages product backlog priorities; the Scrum Master supports Scrum effectiveness; Developers deliver the increment.

46. How many people attend the Scrum call?

The Daily Scrum is for the Developers of the Scrum Team; the Product Owner and Scrum Master participate as Developers when actively working on sprint work. Others may be invited when useful, but the meeting should remain focused. State your team's actual practice and approximate size.

47. SDLC / STLC / Bug life cycle

- SDLC: requirements, planning, design, development, testing, deployment, maintenance.
- STLC: requirement analysis, test planning, test design/data setup, environment readiness, test execution, defect reporting/retesting, closure.
- Defect life cycle: New → Assigned → In Progress → Fixed/Resolved → Retest → Closed; if the fix fails, reopen. Workflow labels vary by organization.

48. Models in testing

Common software development models include Waterfall, V-model, Iterative, Incremental, Spiral, and Agile. The V-model maps development phases to corresponding test levels (for example, requirements to acceptance testing and design to integration/system testing). Choose a model based on risk, requirements stability, feedback needs, and regulatory context.

49. What is OOP?

Object-oriented programming organizes software around objects that combine state and behavior. Four commonly taught principles are encapsulation, abstraction, inheritance, and polymorphism. Selenium frameworks use classes such as LoginPage, methods such as login(), and interfaces such as WebDriver.

50. Encapsulation, polymorphism, inheritance

- Encapsulation: keep state/implementation details private and expose controlled methods.
- Inheritance: a subclass extends a superclass to reuse or specialize behavior.
- Polymorphism: one interface/base type can refer to different implementations; overridden instance methods dispatch at runtime.

```
WebDriver driver = new ChromeDriver(); // interface reference
driver.get("https://example.com");
```

51. Multiple vs multilevel inheritance

Java classes support single inheritance, not multiple inheritance from multiple classes. Multilevel inheritance is supported: A → B → C. Multiple inheritance of type can be achieved through implementing multiple interfaces.

```
class A { }
class B extends A { }
class C extends B { } // multilevel

interface Readable { void read(); }
interface Writable { void write(); }
class FileHandler implements Readable, Writable {
    public void read() { }
    public void write() { }
}
```

52. Interface vs class

A class can hold state, constructors, and concrete methods; a class can extend one class. An interface defines a contract and can declare abstract methods plus default/static methods; a class can implement multiple interfaces. Use interfaces for interchangeable behavior and classes for concrete state/implementation.

53. Define public static void main

public means accessible to the JVM entry point; static means it can be called without creating an object; void means it returns no value; main is the entry method name; String[] args receives command-line arguments.

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

54. What is your folder structure?

A common Maven structure is:

```
project/
  pom.xml
  src/
    main/java/        # reusable framework code (optional)
    test/java/
      tests/          # TestNG tests
      pages/          # Page Objects
      utilities/      # waits, screenshots, helpers
      listeners/      # TestNG listeners
    test/resources/
      testng.xml
      config.properties
      testdata/       # test data files
  target/             # build output and reports
```

Match the explanation to the actual repository; some projects put all framework code under src/test/java or use Cucumber feature files under src/test/resources.

55. Display links using linkText

```
List<WebElement> links = driver.findElements(By.tagName("a"));
for (WebElement link : links) {
    System.out.println(link.getText());
}
// Find a link by exact visible text:
driver.findElement(By.linkText("Contact")).click();
```

By.linkText() locates a specific anchor by exact visible text. It is not used to retrieve all links; use findElements(By.tagName("a")) for that.

56. Program to handle alert popup

```
WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(5));
Alert alert = wait.until(ExpectedConditions.alertIsPresent());
System.out.println(alert.getText());
alert.accept();
```

57. What are the locators?

id, name, className, tagName, linkText, partialLinkText, cssSelector, and xpath. Prefer stable unique attributes and avoid brittle absolute XPath.

58. What framework is used in your project?

Describe the real framework, for example a hybrid Java-Selenium framework with TestNG, Page Objects, reusable utilities, externalized test data, Maven, Git, reporting, and CI. Explain how a test flows through it rather than only listing tool names.

59. What quality standards and structures are followed in your project?

Possible practices include acceptance criteria, coding standards, peer reviews, traceability between stories and tests, risk-based testing, defect severity/priority conventions, regression gates, test evidence, secure handling of credentials, and release sign-off. Mention only standards actually followed by your team; formal standards such as ISO/IEC should not be claimed without evidence.

60. Types of frameworks

- Linear/script-based: straightforward recorded or coded flows, limited reuse.
- Modular: divides the application/tests into reusable modules.
- Data-driven: separates input data from test logic.
- Keyword-driven: maps keywords to actions.
- Library architecture: reusable helper functions.
- BDD: scenarios expressed in business-readable language, commonly with Gherkin/Cucumber.
- Hybrid: combines suitable features of several approaches, such as POM + data-driven + TestNG.

61. Git commands to push changes and explain

```
git status
git switch -c feature/my-change
git add src/test/java
git commit -m "Add automation coverage"
git pull --rebase origin main
git push -u origin feature/my-change
```

After pushing, open a pull request, request review, address comments, and merge according to team policy. If a rebase creates conflicts, resolve files, git add them, and continue the rebase; do not force-push shared branches without following team rules.

62. How is Jenkins used?

Jenkins checks out source code, prepares the environment, runs build/test commands, archives reports/screenshots, and notifies the team. It can be triggered by a schedule, a source-control change, a webhook, or manually. Explain your pipeline stages and where tests run (controller or agent).

63. What is synchronization?

In Selenium, synchronization is waiting for the application to reach the required state before interacting. Use explicit waits such as visibilityOfElementLocated or elementToBeClickable. Avoid Thread.sleep as the default. In Java multithreading, synchronization has a different meaning: controlling concurrent access to shared state.

64. How do you set priorities and check threads in Java?

In TestNG, @Test(priority = n) controls ordering among applicable tests, but test dependencies and suite configuration also affect execution; avoid relying on priority for test independence. In Java, Thread.setPriority() sets a scheduling hint, but the OS/JVM is not required to honor it exactly. Use ExecutorService, futures, synchronization, or concurrent collections for robust concurrency.

```
@Test(priority = 1)
public void login() { }

@Test(priority = 2)
public void profile() { }
```

For TestNG parallel execution, configure parallel and thread-count in testng.xml. Ensure WebDriver and test data are isolated per thread.

65. Brief about TestNG annotations

See Question 28. Know the scope and order of suite/test/class/method setup annotations, plus @Test, @DataProvider, @Parameters, and @Listeners.

66. How to parameterize in TestNG

Use @Parameters for a small number of suite-level values, such as browser or environment, and @DataProvider for multiple datasets. See Questions 18 and 34.

67. Explain Selenium Grid. Can one system act as both Hub and Node?

Selenium Grid runs WebDriver tests on remote machines/browsers and supports parallel execution. In Selenium Grid 4, a standalone mode can run the Grid components on one machine, and a node can run browsers on the same machine. Hub/node architecture terminology is more associated with classic Grid; explain the mode/version used by your setup.

68. Explain collections

See Question 20. Use List for ordered elements and duplicates, Set for uniqueness, and Map for key-value data. Choose ArrayList, LinkedList, HashSet, LinkedHashSet, TreeSet, HashMap, or TreeMap based on access and ordering requirements.

69. What is the flow of a Java program?

The JVM loads classes, verifies and links them, initializes static members as needed, and invokes public static void main(String[] args). Execution creates objects, calls methods, and uses the runtime/JIT as applicable. A Selenium test run adds the test runner (e.g., TestNG), which discovers tests and invokes lifecycle hooks.

70. Mouse hover and alerts

Mouse hover: use Actions.moveToElement(element).perform() (Question 8). JavaScript alert: switch to Alert, read text, accept/dismiss (Question 10). These are different interaction types.

71. Syntax of relative and absolute XPath

```
// Absolute example
/html/body/div[1]/form/input[1]

// Relative examples
//input[@id='username']
//button[normalize-space()='Submit']
//div[@class='row']//a[contains(., 'Edit')]
```

Relative XPath is generally easier to maintain. Validate uniqueness and avoid overly broad contains() expressions.

72. How do you send reports to the client?

A professional flow is to publish or export the report from the approved tool, review pass/fail/skip counts and known failures, attach a concise execution summary, include environment/build/browser details, link defect IDs, and share through the approved client channel. Never send credentials, personal data, or confidential logs outside approved channels.

73. What are TestNG annotations?

See Question 28 for lifecycle annotations, @Test, @DataProvider, @Parameters, and @Listeners.

74. TestNG priority, dependsOn, invocationCount, threadPoolSize, thread-count, parallel

```
@Test(priority = 1)
public void first() { }

@Test(dependsOnMethods = "first")
public void runsAfterFirstPasses() { }

@Test(invocationCount = 3)
public void repeatsThreeTimes() { }

@Test(invocationCount = 6, threadPoolSize = 3)
public void runsInPool() { }
```

threadPoolSize is used with invocationCount to control the thread pool for repeated invocations of that method. thread-count is a suite XML setting used with parallel execution. Parallel execution can cause data collisions if tests share a browser, account, or mutable state.

75. What is DataProvider?

TestNG @DataProvider returns an Object[][] (or supported iterator) containing test inputs. TestNG runs the annotated test for each dataset.

```
@DataProvider(name = "users")
public Object[][] users() {
    return new Object[][] {
        {"alice", "pass1"},
        {"bob", "pass2"}
    };
}
@Test(dataProvider = "users")
public void loginTest(String user, String password) {
    // test login for each row
}
```

76. What are listeners?

Listeners hook into TestNG events for logging, reports, screenshots, and custom result handling. See Question 29. Ensure screenshot capture can access the correct WebDriver instance, especially in parallel runs.

77. What is @Override?

@Override is a Java annotation that tells the compiler a method is intended to override a superclass or interface method. It catches signature mistakes at compile time.

```
class Base {
    void run() { }
}
class Child extends Base {
    @Override
    void run() { }
}
```

78. Implicit vs explicit vs FluentWait

- Implicit wait: global timeout applied to element searches; not condition-specific.
- Explicit wait: waits for a named condition using WebDriverWait.
- FluentWait: configurable timeout, polling interval, and ignored exceptions; WebDriverWait is a specialized wait for WebDriver.

```
Wait<WebDriver> wait = new FluentWait<>(driver)
    .withTimeout(Duration.ofSeconds(15))
    .pollingEvery(Duration.ofMillis(500))
    .ignoring(NoSuchElementException.class);
WebElement item = wait.until(d -> d.findElement(By.id("item")));
```

Avoid mixing implicit and explicit waits. Ignore only expected exceptions.

79. Immutable classes and access modifiers for static/final

An immutable object cannot change observable state after construction. Typical design: final class (or otherwise prevent unsafe subclassing), private final fields, no setters, defensive copies for mutable inputs/outputs, and constructor validation. A class can be immutable without the final keyword if subclassing is safely controlled, but final simplifies the guarantee.

```
public final class Person {
    private final String name;
    private final List<String> skills;

    public Person(String name, List<String> skills) {
        this.name = name;
        this.skills = List.copyOf(skills);
    }
    public String getName() { return name; }
    public List<String> getSkills() { return skills; }
}
```

static is a modifier, not an access modifier. Access modifiers are public, protected, package-private (no keyword), and private. final is a non-access modifier for variables, methods, and classes.

80. Command to get a database connection

```
Connection connection = DriverManager.getConnection(
    "jdbc:mysql://localhost:3306/testdb",
    System.getenv("DB_USER"),
    System.getenv("DB_PASSWORD"));
```

Add the relevant JDBC driver dependency and close the connection with try-with-resources. The URL varies by database.

81. Regular expression

A regular expression (regex) is a pattern used to match or validate strings. In Java, backslashes in regex strings usually need escaping.

```
String email = "qa@example.com";
boolean valid = email.matches("^[A-Za-z0-9+_.-]+@[A-Za-z0-9.-]+$");

String digits = "12345";
boolean onlyDigits = digits.matches("\\d+");

// Selenium example: find elements with an attribute prefix using XPath,
// because Selenium locators do not accept Java regex directly.
driver.findElement(By.xpath("//*[starts-with(@id,'user_')]"));
```

Regex examples are simplified; production validation should use the application's actual requirements. Selenium's CSS/XPath locators do not support arbitrary Java regex syntax.

Quick revision checklist

- Prepare a truthful self-introduction and project/framework explanation.
- Know how your tests are selected, executed, reported, and triggered in CI.
- Distinguish JavaScript alerts, browser windows/tabs, HTML modals, and OS dialogs.
- Use explicit waits and stable locators; keep page objects and test logic separate.
- Review TestNG dependencies, groups, parameters, DataProvider, listeners, and parallel settings.
- Understand Git branch/commit/pull/push flow and your actual repository workflow.
- Use try-with-resources and environment variables for JDBC credentials.
- Personalize all team, client, project, and tool details before the interview.