CTS Selenium Interview Questions with Answers Part 3 (2026) | Saravana Tech

100 Elaborate Answers with Practical Examples

> This guide follows the supplied 100-question list and preserves its numbering, including repeated and incomplete prompts. Where the original list does not include the referenced code or scenario, the answer explicitly identifies the missing information instead of inventing an exact output.

INTERVIEW PREPARATION NOTES

- Use examples that reflect your actual project, tools, responsibilities, and experience.
- Code snippets are illustrative; adapt locators, database URLs, versions, and test data to your environment.
- Do not share credentials or confidential customer/application information in interviews.
- Some terms vary by tool version or company practice; clarify the version/context when needed.

1. Tell me about yourself and relevant automation experience

Use a concise, truthful 60–90 second introduction: total experience, current role, testing scope, automation stack, leadership responsibilities, and one relevant result. Example template: “I have [X] years in QA across manual and automation testing. My hands-on experience includes [actual tools]. I work on requirement analysis, test design, automation maintenance, defect triage, and release reporting. I’m interested in this role because it aligns with my experience in [job requirement].” Keep years and tool experience consistent with your resume.

2. What is an IDE and what is it used for?

IDE means Integrated Development Environment. It combines a code editor, project explorer, build/run tools, debugger, refactoring support, and often version-control integration. Examples include IntelliJ IDEA, Eclipse, and Visual Studio Code. It helps create, compile, execute, debug, and maintain code more efficiently than using only a plain text editor.

3. Can you run your complete suite in an IDE?

Usually yes, if the IDE has the required JDK, dependencies, test framework plugin, configuration, environment access, and test data. In Eclipse/IntelliJ, you can run a TestNG XML suite or a test class; Maven projects can also run using Maven goals. Large suites may be better run through Maven/CI to match production-like settings and avoid IDE-specific configuration.

4. Can you directly reference ChromeDriver in a script? Will it work?

A Selenium script can instantiate ChromeDriver when Selenium and a compatible driver/browser setup are available. Modern Selenium versions include Selenium Manager, which can resolve/download a suitable driver in supported environments. Older setups required a chromedriver executable and sometimes System.setProperty("webdriver.chrome.driver", "path"). ChromeDriver is not a “Chrome server reference”; for remote execution use RemoteWebDriver and a Selenium Grid URL. Corporate proxy/certificate restrictions may require preinstalled drivers.

5. What is Maven and why use it?

Maven is a Java build and dependency-management tool driven by pom.xml. It downloads dependencies, manages plugin versions, standardizes build phases, runs tests, packages artifacts, and supports reproducible CI builds. Common commands: mvn clean test, mvn package, mvn verify. It reduces manual JAR management and makes local/CI builds more consistent.

6. Explain the structure of pom.xml

Core sections include modelVersion, groupId, artifactId, version, packaging, properties, dependencies, dependencyManagement, build/plugins, repositories, and profiles. Example: <groupId>com.example</groupId><artifactId>ui-tests</artifactId> <version>${project.version}</version>. Properties centralize versions; dependencies add libraries; plugins configure tasks such as compiler and Surefire. Parent POMs can provide shared configuration.

7. How do you parameterize a version in pom.xml?

Define a property and reference it wherever appropriate. Example: <properties><selenium.version>4.x.y</selenium.version> </properties>, then in the dependency use <version>${selenium.version}</version>. For the project version, use a Maven property or parent/version management as supported by the project; modern Maven supports CI-friendly version properties such as ${revision} with appropriate configuration. Prefer one controlled source for versions.

8. What is Maven Surefire and why is it needed?

Maven Surefire Plugin runs unit and test-framework tests during Maven's test phase. It supports JUnit and TestNG providers and can select tests, pass system properties, configure groups, and run TestNG XML suites. Reports typically appear under target/surefire-reports. Failsafe is commonly used for integration tests during integration-test/verify phases.

9. What is POM model?

POM can mean two different things: Maven Project Object Model (pom.xml), which describes build coordinates, dependencies, plugins, and configuration; or Selenium Page Object Model, a design pattern where page classes encapsulate locators and page actions. Clarify which meaning the interviewer intends.

10. What is a constructor and why is it needed?

A constructor initializes a new object. It has the same name as its class and no return type. Constructors can initialize fields, validate inputs, and establish dependencies. Java provides a default no-argument constructor only if no constructor is declared; once you define a constructor, the compiler does not automatically add that default constructor.

11. List a few WebDriver constructors

WebDriver is an interface and has no constructor. Browser-specific implementations have constructors, for example new ChromeDriver(), new FirefoxDriver(), and new EdgeDriver(); these return driver objects implementing WebDriver. RemoteWebDriver is commonly constructed with a URL and browser Options, for example new RemoteWebDriver(gridUrl, new ChromeOptions()). Constructor availability depends on Selenium version.

12. What are constructors with parameters?

A parameterized constructor accepts values to initialize object state. Example: class User { private String name; User(String name){ this.name=name; } } then User u=new User("Ravi");. The this keyword distinguishes the instance field from the parameter. You can overload constructors with different parameter lists.

13. Explain Jenkins

Jenkins is an automation server used for continuous integration and delivery. A pipeline can check out source code, set up the JDK, run Maven tests, publish reports, archive logs/screenshots, and notify a team. Triggers can include commits, pull requests, and scheduled runs. Store credentials in Jenkins credentials or a secret manager, not in source code.

14. Explain overloading and overriding

Overloading uses the same method name with different parameter lists and is resolved at compile time. Overriding provides a subclass implementation of an inherited instance method with a compatible signature and is resolved at runtime based on the object. Return type alone cannot distinguish overloaded methods. Static methods are hidden, not overridden.

15. Three environments have text boxes with different labels but similar properties. How do you enter data and handle exceptions?

Do not rely only on shared attributes if they identify multiple fields. Use an environment-specific locator map, a stable container/label relationship, or a data-testid. Wait for the chosen element, enter the value, and catch only expected exceptions with meaningful recovery/logging. Example (Java): try { WebElement field = wait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath("//label[normalize-space()='"+label+"']/following::input[1]"))); field.clear(); field.sendKeys(value); } catch (TimeoutException | NoSuchElementException e) { logger.error("Field not available for environment: "+env, e); throw e; }. If labels differ, pass the correct label for the selected environment.

16. TestNG XML is on one machine; run tests on three other IP-addressed machines

Use Selenium Grid or a remote execution setup. Put test code/configuration where the workers can access it, and point RemoteWebDriver to a Grid endpoint; the Grid routes sessions to nodes. If the goal is three independent machines rather than Grid nodes, use CI agents or remote commands to launch the suite on each machine. A local testng.xml does not itself distribute tests to arbitrary IPs. Ensure browser, driver, network, Java, and test data are configured on each machine.

17. Name some exceptions faced in Selenium

Examples include NoSuchElementException, TimeoutException, StaleElementReferenceException, ElementClickInterceptedException, ElementNotInteractableException, NoSuchFrameException, NoSuchWindowException, and InvalidSelectorException. Explain a real occurrence and root cause. Use explicit waits, robust selectors, correct frame/window context, and refreshed locators; do not blindly catch and ignore errors.

18. What is StaleElementReferenceException?

It occurs when a previously located WebElement no longer refers to the current DOM element, often because the page refreshed, an Angular/React component rerendered, or the element was replaced. Fix it by waiting for the new state and locating the element again. Avoid caching dynamic elements for long periods; use a locator and re-find when needed.

19. Explain static and final

static members belong to the class, not each object; they can be accessed using the class name. final on a variable prevents reassignment, on a method prevents overriding, and on a class prevents subclassing. A final reference cannot point to another object, but the referenced object's mutable state may still change.

20. What happens if you change a final value? What exception occurs?

For a local final variable assigned once, a second assignment is a compile-time error, not a runtime exception. Example: final int n=10; n=20; // compile error: cannot assign a value to final variable n. A final object reference cannot be reassigned, though methods may mutate the object if it is mutable. A blank final must be assigned exactly once on every valid construction path.

21. Additional Java and Selenium questions during the interview

This is a note rather than a specific question. Be ready for follow-ups on Java OOP/collections/exceptions, Selenium locators/waits/frames/windows, test design, TestNG, Maven, debugging, and project examples. Answer the exact question first, then add a concise example.

22. Windows and pop-ups

For browser tabs/windows, save the original handle, perform the action, wait until the handle count changes, switch to the new handle, interact, and switch back if needed. For JavaScript alerts use driver.switchTo().alert() and accept(), dismiss(), or getText(). HTML modal dialogs are normal DOM elements, not browser alerts, and should be located as elements.

23. How do you upload a file?

For a standard HTML input type=file, use sendKeys with the absolute path on the input element; this is usually more reliable than the OS file chooser. Example: driver.findElement(By.cssSelector("input[type='file']")).sendKeys(Paths.get("src/test/resources/file.xlsx").toAbsolutePath().toString());. If the control is hidden, inspect the application's supported upload mechanism; Selenium cannot directly control native OS dialogs through ordinary WebDriver commands.

24. Explain Page Object Model

POM is a UI automation design pattern where page/component classes contain locators and actions, while test classes contain scenario flow and assertions. It improves reuse, readability, and maintenance. It is not the same as Maven's POM file.

25. If locators keep changing, how do you manage them?

First determine why they change. Ask developers for stable IDs or data-testid attributes; prefer accessible labels and stable semantic attributes; scope locators to a unique component; centralize them in page objects; and avoid absolute XPath/index-based locators. For unavoidable dynamic values, build a parameterized locator from validated input. Track repeated instability as an application testability issue.

26. Array-related questions

Know declaration, initialization, indexing, traversal, insertion limitations, sorting, searching, multidimensional arrays, and differences from ArrayList. Java arrays have fixed length; ArrayList resizes dynamically. Example: int[] values={10,20,30}; for(int value:values) System.out.println(value);. To insert, create a new array or use a collection.

27. Describe a challenge you faced

Use STAR: Situation, Task, Action, Result. Select a real example such as flaky UI timing, test data collisions, unstable locators, slow suite execution, or environment failures. Explain the root-cause investigation, the change you made, and the verified result. Avoid invented metrics.

28. A challenge in the interviewer's current project and whether you have a solution

Ask clarifying questions about constraints, scale, current behavior, and success criteria before proposing a solution. Offer a hypothesis and a small validation plan rather than claiming certainty. Separate what you have done before from what you would investigate. Consider maintainability, observability, test data, and failure recovery.

29. What is an object? What are its properties?

An object is a runtime instance of a class. It has identity, state (field values), and behavior (methods). A class defines the possible fields and methods; each object holds its own instance state, while static state is shared at class level.

30. Explain Git

Git is a distributed version-control system. Common commands: git clone, status, add, commit, pull, fetch, branch, switch, merge, rebase, push, and log. Teams use branches, pull requests/code reviews, conflict resolution, and tags. Keep commits focused and never commit secrets or generated files unintentionally.

31. Explain Agile model

Agile is an iterative approach that delivers work in small increments, incorporates feedback, and adapts to change. Scrum is one Agile framework with defined accountabilities, events, and artifacts. Teams maintain a prioritized backlog, plan a sprint, hold daily Scrum, review the increment, and retrospect to improve.

32. Use of getters and setters in Java

Getters expose a field value and setters update it, often with validation. They support encapsulation by keeping fields private and controlling access. Example: private int age; public int getAge(){return age;} public void setAge(int age){if(age<0) throw new IllegalArgumentException(); this.age=age;}. Not every field needs a setter; immutable objects may expose only getters.

33. What is Hashtable?

Hashtable is a legacy synchronized Map implementation. It does not allow null keys or null values. It is generally superseded by HashMap for single-threaded use or ConcurrentHashMap for concurrent access. Hashtable synchronizes individual operations but is not always the best choice for compound thread-safe logic.

34. What is HashSet?

HashSet implements Set and stores unique elements using hashing (backed by a HashMap). It does not guarantee iteration order and generally offers average O(1) add, remove, and contains operations when hashes are well distributed. Objects must implement equals() and hashCode() consistently.

35. How do you connect to a database? Write code

Use JDBC with a JDBC driver, connection URL, credentials supplied securely, and try-with-resources. Example: String url="jdbc:postgresql://host:5432/db"; try(Connection c=DriverManager.getConnection(url,user,password); PreparedStatement ps=c.prepareStatement("SELECT name FROM users WHERE id = ?")){ ps.setInt(1, id); try(ResultSet rs=ps.executeQuery()){ while(rs.next()) System.out.println(rs.getString("name")); } }. Add the correct driver dependency, handle SQLException, and never hardcode production credentials.

36. Get a unique ID from the string "id:9864" in multiple ways

If the format is fixed, substring after the colon is simple: String id=s.substring(s.indexOf(':')+1).trim();. Alternatives include split(":",2)[1].trim(), replaceFirst("^id\\s*:","").trim(), or regex: Pattern.compile("id\\s*:\\s*(\\d+)").matcher(s). If IDs may contain letters or format varies, adjust validation and check matcher.find() before accessing group(1).

37. What is an interface?

An interface defines a contract. A class implements it and supplies required behavior; a class can implement multiple interfaces. Interfaces support loose coupling and polymorphism. They can have abstract, default, and static methods, and modern Java permits private helper methods.

38. What is inheritance?

Inheritance allows a subclass to inherit accessible behavior and state from a superclass and specialize or extend it. Java classes support single class inheritance and multiple interface implementation. Use inheritance for a genuine “is-a” relationship; prefer composition when it better models reuse.

39. What is an abstract class?

An abstract class cannot be instantiated directly and may contain abstract methods without bodies and concrete methods with implementations. Subclasses must implement inherited abstract methods unless they are abstract too. It can hold fields and constructors.

40. How do you insert data into an array?

Arrays have fixed length, so they do not support insertion in place when full. To insert at index i, allocate a larger array, copy elements before i, place the new value, then copy the remainder.

> **EXAMPLE / INTERVIEW TIP**
>
> Example: int[] a={1,3,4}; int pos=1, value=2; int[] b=new int[a.length+1]; System.arraycopy(a,0,b,0,pos); b[pos]=value; System.arraycopy(a,pos,b,pos+1,a.length-pos);. For frequent insertion, consider ArrayList.

41. Move to a tag with no attributes—write two ways

Without attributes, locate it by tag name, text, or relationship to a stable nearby element. Examples: driver.findElement(By.tagName("u")) if only one underlined element exists; or driver.findElement(By.xpath("//label[normalize-space()='Email']/following-sibling::span[1]")). If several elements match, scope to a unique parent. Do not rely on tag name alone unless uniqueness is verified.

42. What framework did you use in your project?

Describe only your real framework. A common answer for web UI automation is a hybrid framework combining POM, data-driven tests, reusable utilities, TestNG, Maven, logging/reporting, and Jenkins. Explain each layer and one reason for choosing it. If your actual project differs, describe that accurately.

43. Explain TestNG annotations in detail

Common annotations: @BeforeSuite/@AfterSuite once per suite; @BeforeTest/@AfterTest around a testng.xml <test>; @BeforeClass/@AfterClass around class execution; @BeforeMethod/@AfterMethod around each test method invocation; @Test marks a test; @BeforeGroups/@AfterGroups run around group methods; @DataProvider supplies data; @Parameters reads XML parameters; @Factory creates test instances; @Listeners registers listeners. Order can be affected by inheritance, groups, dependencies, parallel execution, and configuration failures.

44. Explain Jenkins usage

Jenkins automates the test lifecycle: checkout code, build with Maven, run TestNG suites, publish test results, archive artifacts, and notify the team. A pipeline as code (Jenkinsfile) makes the process versioned and repeatable. Secure credentials and make failures diagnosable with logs/screenshots.

45. Select the second dropdown value without Select class or select command

For a native HTML <select>, keyboard actions can select an option, or click the dropdown and click the desired option if the UI exposes options. Example: WebElement dropdown=driver.findElement(By.id("country")); dropdown.click(); new Actions(driver).sendKeys(Keys.ARROW_DOWN).sendKeys(Keys.ENTER).perform();. This selects relative to the current selection; first establish the current selected option and confirm the expected result. For custom dropdowns, locate and click the second visible option in its overlay instead.

46. Default dropdown selection is the last value; use Actions to choose the second value

Do not assume one ArrowDown selects the second option when the current selection is the last. Click/open the control, use HOME to move to the first option and then ARROWDOWN once to the second, followed by ENTER, if that control supports keyboard navigation. Example: new Actions(driver).click(dropdown).sendKeys(Keys.HOME).sendKeys(Keys.ARROW_DOWN).sendKeys(Keys.ENTER).perform();. Verify selected text. For custom controls, use the DOM options and click the intended item.

47. Write code for the scenario below

The specific scenario/code referred to is not included in the supplied list, so an exact solution cannot be written without inventing requirements. In an interview, ask for the full scenario, HTML, expected behavior, and constraints. If it refers to the next question, see the frame/window flow in Question 50.

48. Click OK, handle a frame, select first radio, click Next; repeat for another frame, then next window

Frames are not windows. Keep the current window handle, switch into each frame, select the first radio, click Next, then detect whether the next frame replaces the current one or nests inside it. If a new frame appears at top level, return to defaultContent() and locate that frame; if nested, use parentFrame()/frame switching. After the final Next opens a new window, wait for the handle count to increase and switch to the new handle.

EXAMPLE / INTERVIEW TIP

Illustrative Java pattern: String main=driver.getWindowHandle(); driver.findElement(By.id("ok")).click(); WebDriverWait wait=new WebDriverWait(driver,Duration.ofSeconds(10)); wait.until(ExpectedConditions.frameToBeAvailableAndSwitchToIt(By.id("frame1"))); driver.findElement(By.cssSelector("input[type='radio']")).click(); driver.findElement(By.id("next")).click(); driver.switchTo().defaultContent(); wait.until(ExpectedConditions.frameToBeAvailableAndSwitchToIt(By.id("frame2"))); driver.findElement(By.cssSelector("input[type='radio']")).click(); driver.findElement(By.id("next")).click(); driver.switchTo().defaultContent(); wait.until(d->d.getWindowHandles().size()>1); for(String h:driver.getWindowHandles()) if(!h.equals(main)){driver.switchTo().window(h); break;} Use page-specific selectors and frame handling.

49. Difference between throw and throws

throw explicitly raises one exception object inside code: throw new IllegalArgumentException("Invalid input");. throws declares in a method signature that the method may propagate one or more exception types: void read() throws IOException. throw is a statement; throws is part of the declaration. Checked exceptions must be caught or declared.

50. How do you use regular expressions?

Regular expressions match text patterns. In Java use Pattern and Matcher, or String methods such as matches(), replaceAll(), and split(). Example: Pattern p=Pattern.compile("\\d+"); Matcher m=p.matcher("id:9864"); if(m.find()) System.out.println(m.group());. Java string literals require escaping backslashes. Anchor patterns with ^/$ when matching the whole string.

51. How do you connect to Excel?

In Java automation, Apache POI is commonly used to read/write Excel .xls/.xlsx files. Open with FileInputStream and WorkbookFactory or XSSFWorkbook, access Sheet/Row/Cell, then close resources. For formulas and numeric/date cells, use DataFormatter or evaluate formulas as needed. Store test data outside source code and avoid passwords in plain-text spreadsheets.

52. What is Selenium Grid? Explain and how used in a project

Grid enables remote WebDriver sessions across multiple browsers, machines, and operating systems, and supports parallel execution. In Selenium Grid 4, components include Router, Distributor, Session Map, Event Bus, and Nodes; standalone mode is simpler for small setups. Use RemoteWebDriver with a Grid URL and browser Options. Explain actual usage honestly—local tests, shared Grid, or CI nodes—and how you isolated test data.

53. Difference between Selenium and QTP/UFT

Selenium is an open-source browser automation ecosystem focused primarily on web applications and supports several programming languages/browsers. QTP, now UFT One in current product naming, is a commercial testing tool from OpenText with vendor-provided IDE/features and broader packaged integrations depending on license/version. Compare licensing, language/tooling, browser support, integration, and your project needs; neither tool is universally best.

54. What is implicit wait?

Implicit wait configures WebDriver to poll the DOM for a specified duration when locating elements before returning a not-found error. Example: driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(5));. It applies globally to element-location calls. Prefer explicit waits for specific UI states and avoid mixing wait types because timing can become confusing.

55. How many waits are there in Selenium?

Interviewers usually expect three common wait approaches: implicit wait, explicit wait, and fluent wait. ExplicitWait and FluentWait are closely related in Selenium's API: WebDriverWait is a specialization of FluentWait for WebDriver. Thread.sleep is a fixed pause, not a Selenium wait strategy, and is generally less reliable.

56. Difference between implicit and explicit wait

Implicit wait applies to locating elements across the driver and waits up to a configured duration for a match. Explicit wait polls for a specific condition, such as visibility, clickability, text, or URL. Explicit waits are more targeted and easier to diagnose for dynamic UI behavior. Avoid mixing them.

57. How will you find elements?

Use WebDriver findElement(By...) or findElements(By...). Locators include id, name, className, tagName, linkText, partialLinkText, cssSelector, and xpath. findElement returns the first match or throws NoSuchElementException; findElements returns a list, possibly empty. Prefer stable unique locators.

58. Write the line of code for findElements

Java example: List<WebElement> elements = driver.findElements(By.cssSelector(".result-item"));. Iterate using for (WebElement element : elements) { System.out.println(element.getText()); }. Import java.util.List and Selenium WebElement/By types.

59. Element may or may not appear; find it and catch the exception

If presence is optional, findElements avoids an exception: List<WebElement> found=driver.findElements(By.id("optional")); if(!found.isEmpty()){...} else {...}. If using findElement, catch NoSuchElementException; for asynchronous appearance, use a short explicit wait and catch TimeoutException. Do not catch a broad Exception unless there is a reason.

60. What does data-driven testing mean?

Data-driven testing runs the same test logic with multiple data sets, such as valid/invalid credentials, users, or environments. Test logic is separated from inputs. TestNG @DataProvider, CSV, Excel, JSON, and database sources are common approaches. Include clear data ownership, validation, and cleanup.

61. Are you aware of JDBC?

JDBC (Java Database Connectivity) is Java's standard API for connecting to relational databases, executing SQL, and reading results. It uses a JDBC driver, Connection, Statement/PreparedStatement, and ResultSet. PreparedStatement with parameters helps prevent SQL injection and handles values safely.

62. How do you connect using JDBC?

Load/add the appropriate database driver dependency, use a JDBC URL, connect with credentials, execute SQL, read the ResultSet, and close resources. Example: try(Connection c=DriverManager.getConnection(url,user,password); PreparedStatement ps=c.prepareStatement("SELECT id,name FROM users WHERE id=?")){ps.setInt(1, id); try(ResultSet rs=ps.executeQuery()) {while(rs.next()) System.out.println(rs.getString("name"));}}. Handle SQLException and use secret management.

63. If Excel is input, how do you connect to Excel?

Use Apache POI rather than JDBC for a standard local workbook. Read rows/cells from an Excel sheet, convert values safely, and feed them into a TestNG @DataProvider or test. For an Excel file exposed through a database-like service, the approach could differ; clarify what “connect” means.

64. Explain POM

Selenium Page Object Model encapsulates page locators and actions in page classes, while test classes focus on scenarios and assertions. It improves reuse and maintenance. Example: LoginPage.login(username,password) enters values and clicks sign-in; LoginTest verifies the home page. Maven's pom.xml is a different use of the acronym.

65. Is POM better or data-driven testing better?

They solve different problems and are not alternatives. POM organizes UI interaction and page logic; data-driven testing supplies multiple inputs to the same test logic. They are commonly combined: a TestNG data provider passes inputs to a test, and the test calls POM page methods.

66. What is a hybrid framework?

A hybrid automation framework combines techniques, often POM, data-driven testing, reusable utilities, and sometimes keyword-driven execution. It may include configuration, logging, reporting, test runners, and CI integration. The term is not a strict standard, so explain exactly which patterns your framework combines.

67. Explain project structure

A common Maven Selenium/TestNG layout: pom.xml; testng.xml; src/main/java/pages, utils, driver, listeners, models; src/test/java/tests; src/test/resources/config, testdata; and target for generated build outputs/reports. Projects may place page objects under test source depending on design. Explain where tests, page objects, data, configuration, and reports live in your actual repository.

68. What unit-testing frameworks do you know?

Common Java test frameworks include JUnit 4, JUnit 5, and TestNG. TestNG is popular for groups, dependencies, data providers, and suite XML; JUnit is widely used for unit and integration tests and has a strong ecosystem. Choose based on project conventions and features.

69. Write an assertion comparing actual and expected login name

TestNG: Assert.assertEquals(actualName, expectedName, "Login name mismatch");. JUnit 5: Assertions.assertEquals(expectedName, actualName). Check each framework's argument order: TestNG commonly uses actual then expected; JUnit uses expected then actual. Read the actual displayed value after the page is ready.

70. Difference between JUnit and TestNG

Both are Java testing frameworks. TestNG includes XML suites, groups, dependencies, @DataProvider, and flexible parallel options. JUnit 5 uses the Platform, Jupiter API, and extensions, with parameterized tests and a broad ecosystem. Capabilities overlap and versions matter; select based on team needs and tooling.

71. What is SQL?

SQL (Structured Query Language) is used to define, query, and manipulate relational database data. Common statements: SELECT, INSERT, UPDATE, DELETE, CREATE, and ALTER. SQL also supports joins, aggregation, subqueries, transactions, and constraints. Use WHERE clauses carefully for updates/deletes.

72. What do you know about Oracle?

Oracle Database is a relational database management system. SQL can query and update tables; PL/SQL adds procedural constructs such as variables, loops, procedures, functions, and exception handling. Explain what you actually used—basic SELECT/JOIN/aggregation, data validation, or more advanced work—rather than implying deep Oracle administration experience.

73. How many joins are you aware of?

Common joins are INNER JOIN, LEFT OUTER JOIN, RIGHT OUTER JOIN, and FULL OUTER JOIN. CROSS JOIN produces a Cartesian product. A self join joins a table to itself; it is a pattern rather than a separate join keyword in many SQL dialects. Oracle also supports join syntax according to its version.

74. Write SELECT and INSERT queries

SELECT example: SELECT employee_id, employee_name FROM employees WHERE department_id = 10;. INSERT example: INSERT INTO employees (employee_id, employee_name, department_id) VALUES (101, 'Ravi', 10);. Use the actual schema and bind parameters in application code; verify constraints and transaction behavior.

75. What is a constructor?

A constructor initializes a new object, shares the class name, and has no return type. It can be no-argument or parameterized; constructors can be overloaded. If no constructor is declared, Java supplies a default no-argument constructor. Constructors are not inherited.

76. What is an interface?

An interface defines a contract of behavior. Classes implement it, and a class can implement multiple interfaces. Interfaces support polymorphism and reduce coupling between callers and implementations.

77. What is an abstract class?

An abstract class cannot be instantiated directly and may contain abstract and concrete methods, fields, and constructors. Subclasses must implement abstract methods unless they remain abstract.

78. Difference between abstract class and interface

An abstract class supports instance state, constructors, and shared implementation; a class can extend only one class. An interface expresses a contract and allows a class to implement multiple interfaces. Modern interfaces can include default/static methods, so the difference is not simply “interface has only abstract methods.”

79. Explain method overloading and overriding

Overloading: same name, different parameter lists; compile-time selection. Overriding: subclass supplies a compatible implementation of an inherited instance method; runtime dispatch. Return type alone cannot overload; use @Override to catch signature mistakes.

80. What is STLC?

STLC means Software Testing Life Cycle: the activities performed to plan, design, execute, and close testing. Typical phases are requirement analysis, test planning, test case development, test environment setup, test execution, defect reporting/retesting/regression, and test closure. Exact phase names vary by organization.

81. Write the STLC life cycle

A practical sequence: 1) Requirement analysis and testability review. 2) Test planning and estimation. 3) Test case/test data design. 4) Environment setup. 5) Test execution and defect logging. 6) Retesting fixes and regression testing. 7) Test closure with metrics, lessons learned, and sign-off. Activities can overlap in Agile teams.

82. Scenario

The scenario text is missing from the question list. An exact answer would require the complete scenario, expected behavior, and any supplied code/HTML. In an interview, ask a clarifying question instead of inventing requirements.

83. Do you have Agile knowledge?

Explain Agile principles and your real team practices: iterative delivery, prioritizing a backlog, collaboration, continuous feedback, and adapting to change. If using Scrum, describe Sprint Planning, Daily Scrum, Sprint Review, and Sprint Retrospective; the Product Backlog and Sprint Backlog are key artifacts.

84. What is a release backlog?

“Release backlog” is not a universally defined Scrum artifact. Teams may use it to mean the prioritized set of features, fixes, and technical work intended for a release. In Scrum, the official artifacts are Product Backlog, Sprint Backlog, and Increment. Explain your organization's usage and how release scope is prioritized and tracked.

85. What should be discussed in a daily meeting?

In a Daily Scrum, Developers inspect progress toward the Sprint Goal and adapt the plan for the next day. Teams often discuss what was completed, what is next, and blockers, but the meeting is not meant to be a status report to a manager. Keep it focused and take detailed problem-solving offline.

86. Tell me about yourself

Use a concise introduction: years of experience, current role, manual/automation skills, project responsibilities, leadership, and the role you want. Make every statement consistent with your resume and the job description.

87. Why Cognizant?

Connect your answer to the specific role, project/domain, technology, learning opportunities, scale, or career path. Avoid generic praise alone. Example: “This role aligns with my QA automation and delivery experience, and I’m interested in contributing to quality engineering while growing in [specific technology/domain]. From the job description, the focus on [actual requirement] is particularly relevant.” Confirm the details before the interview.

88. What framework did you use?

Describe the framework you actually used. If it combines POM, data-driven tests, TestNG, Maven, utilities, reporting, and Jenkins, call it hybrid and explain the responsibilities of each layer. Do not label it hybrid merely because it has multiple folders.

89. Draw and explain POM structure

Example structure: pom.xml and testng.xml at root; src/main/java/pages for page objects, utils for helpers, driver for browser lifecycle, listeners for reporting hooks; src/test/java/tests for test classes; src/test/resources for configuration and test data; target for build output/reports. Some projects put page objects in src/test/java. Explain the real structure rather than treating one layout as mandatory.

90. Write a polymorphism program

Compile-time polymorphism example (overloading): class Calculator { int add(int a,int b){return a+b;} double add(double a,double b){return a+b;} }. Runtime polymorphism example (overriding): class Animal { void sound(){System.out.println("animal");} } class Dog extends Animal { @Override void sound(){System.out.println("dog");} } Animal a=new Dog(); a.sound(); prints dog.

91. Overloading and overriding

Overloading is the same method name with different parameters and is resolved at compile time. Overriding is a subclass's replacement implementation of an inherited instance method and is resolved at runtime. Static methods are hidden, not overridden.

92. HashMap and Hashtable

HashMap is unsynchronized, permits one null key and null values, and is generally preferred for ordinary single-threaded map use. Hashtable is synchronized at method level and prohibits null keys/values; it is a legacy class. For concurrent use, ConcurrentHashMap is usually preferred, but compound operations should use atomic methods.

93. Vector and ArrayList

Both are resizable-array List implementations that preserve order and allow duplicates. Vector's legacy methods are synchronized; ArrayList is not synchronized and is usually preferred for single-threaded use. Their indexed get is O(1) on average; insertion/removal in the middle requires shifting elements. For concurrent access, choose synchronization/concurrent design based on the use case.

94. Implicit and explicit wait

Implicit wait globally affects element lookup; explicit wait polls for a specific condition such as visibility or clickability. Explicit waits give better control for dynamic pages. Avoid combining them unless you understand the resulting timeout behavior.

95. Regression testing and retesting

Retesting verifies that a specific reported defect is fixed. Regression testing checks that changes have not broken existing functionality. A defect fix typically requires retesting the failed scenario and regression testing related/high-risk areas.

96. Smoke testing and sanity testing

Smoke testing is a broad, shallow check that a build's critical paths work and is suitable for further testing. Sanity testing is a narrower check focused on a specific change or area after a minor change/fix. Definitions can vary by organization, so state how your team uses them.

97. Explain your project

Cover business purpose, architecture, your role, team and Agile process, testing scope, automation stack, framework layers, CI/reporting, major risks, and one genuine achievement/challenge. Avoid confidential details and distinguish your contribution from the team's work.

98. Explain Agile

Agile is an iterative, incremental way of delivering software with collaboration, feedback, and adaptation. Scrum is one framework within Agile. Explain backlog prioritization, sprint planning, daily Scrum, review, retrospective, and how QA contributes throughout the sprint—not only at the end.