CTS Selenium Interview Questions with Answers Part 2 (2026) | Saravana Tech

100 Questions with Elaborate Answers, Examples, and Interview Guidance

> Prepared from the exact 100-question list supplied. Original numbering and repeated questions are retained. Where the prompt refers to missing code or an “above” snippet, the answer explains what can be concluded and what information is needed for an exact output.

HOW TO USE THIS GUIDE

- Understand the concept first, then practise explaining it in your own words.
- Code samples are illustrative; update locators, versions, test data, and project details for your environment.
- For project and behavioral answers, use truthful examples from your own work and do not invent achievements or metrics.
- For version-sensitive tools such as Selenium Grid and TestNG, confirm the version used by the project.

1. What is the XPath relationship between table elements?

In an HTML table, <table> contains <thead>, <tbody>, and/or <tfoot>; <tbody> contains <tr> rows; each <tr> contains <td> data cells or <th> header cells. XPath axes express these relationships: child (/), descendant (//), parent (..), ancestor, following-sibling, and preceding-sibling.

> **EXAMPLE / INTERVIEW TIP**
>
> Example: //tbody/tr[1]/td[1] selects the first cell in the first body row. //td[1]/parent::tr selects the row containing a first-position td. //td[1]/ancestor::tbody selects its containing tbody.

2. What is the relationship between the first <td> and <tr>, and between the first <td> and <tbody>?

A <td> is a direct child of its row (<tr>); the row is a direct child of <tbody>. Therefore, <td> is a descendant (usually a grandchild) of <tbody>, not its direct child.

> **EXAMPLE / INTERVIEW TIP**
>
> Examples: //tr/td[1] (td as child of tr); //tbody/tr/td[1] (td reached through tbody); //td[1]/parent::tr (parent row); //td[1]/ancestor::tbody (ancestor body).

3. Why can ArrayList be faster than LinkedList for indexed access?

ArrayList uses a resizable array, so get(index) is O(1) on average. LinkedList is a doubly linked list; get(index) must traverse nodes from an end, so indexed access is O(n). Insertion/removal in the middle can be O(1) for LinkedList only when the node/iterator position is already known; finding the position costs time. ArrayList often performs better due to memory locality.

> **EXAMPLE / INTERVIEW TIP**
>
> Use ArrayList for frequent indexed reads and iteration. Consider LinkedList for queue/deque operations, though ArrayDeque is often a better queue choice. Both implement List and preserve order/duplicates.

4. What is the Page Object Model (POM) project structure and its benefits?

POM is a design pattern in which each page or major UI component has a class containing its locators and page actions. Test classes describe business scenarios and call page methods rather than embedding locators everywhere.

> **EXAMPLE / INTERVIEW TIP**
>
> Typical structure: src/main/java/pages (LoginPage, HomePage), src/main/java/utils (WaitUtil, ExcelUtil, ConfigReader), src/test/java/tests (LoginTest), src/test/resources (config.properties, test data, testng.xml), pom.xml. Benefits: reuse, separation of concerns, easier maintenance when UI changes, readable tests, and less duplication. POM is a pattern, not a complete framework by itself.

5. What is TestNG? Explain annotations and execution order.

TestNG is a Java testing framework supporting annotations, assertions, groups, dependencies, data providers, parameterization, parallel execution, and XML suite configuration. A common order is @BeforeSuite, @BeforeTest, @BeforeClass, @BeforeMethod, @Test, @AfterMethod, @AfterClass, @AfterTest, @AfterSuite. @BeforeGroups/@AfterGroups run around tests in specified groups; @DataProvider supplies test data; @Parameters injects XML values.

> **EXAMPLE / INTERVIEW TIP**
>
> Actual execution depends on the suite, groups, dependencies, inheritance, and parallel settings. TestNG annotation methods generally use no arguments unless the annotation defines supported parameters (for example, @DataProvider can receive Method or ITestContext).

6. Which parameters run tests repeatedly? Explain thread-count, threadPoolSize, invocationCount and maxInstances.

Use @Test(invocationCount = n) to invoke one test method n times. Use @Test(threadPoolSize = n, invocationCount = m) to run those invocations concurrently using a pool; threadPoolSize is meaningful with invocationCount. In testng.xml, parallel="methods|tests|classes" and thread-count="n" control parallel

execution. maxInstances is not a standard @Test parameter in core TestNG; it may belong to a custom runner, framework, or another tool, so verify the exact context.

> **EXAMPLE / INTERVIEW TIP**
> Example: @Test(invocationCount=5, threadPoolSize=2) runs five invocations using up to two threads. Parallel tests must not share unsafe mutable state or a single WebDriver instance.

7. Difference between dependsOnGroups and dependsOnMethods?

dependsOnMethods makes a test depend on one or more named test methods. dependsOnGroups makes it depend on all methods in the named group(s). If a dependency fails or is skipped, the dependent test is normally skipped unless configuration changes the behavior.

> **EXAMPLE / INTERVIEW TIP**
> Example: @Test(dependsOnMethods="login") depends on login(). @Test(dependsOnGroups="smoke") depends on tests belonging to the smoke group. Dependencies can make tests order-sensitive; independent tests are generally more maintainable.

8. Difference between priority and dependsOnMethods, with example.

priority orders otherwise eligible test methods; lower numeric values run first by default. dependsOnMethods expresses a prerequisite and controls whether a test is eligible to run based on another method's result. Priority is ordering, not a success dependency.

> **EXAMPLE / INTERVIEW TIP**
> Example: @Test(priority=0) void login() {} and @Test(priority=1) void search() {} orders login before search. For a real prerequisite use @Test(dependsOnMethods="login") on search.

9. If a test has priority=0 and dependsOnMethods=method1, which runs first?

The dependency is the controlling constraint. TestNG must execute method1 before the dependent test can run, regardless of the dependent test's priority. Priority can order tests only when dependency constraints allow it.

> **EXAMPLE / INTERVIEW TIP**
> Example: @Test(priority=0, dependsOnMethods="method1") void method2() {} cannot run before method1. If method1 fails, method2 is normally skipped.

10. How would you automate a personal-information page with a Submit button?

Identify stable locators for each field, populate valid test data, verify entered values, click Submit, and assert the expected next-page state. Prefer IDs, accessible labels, data-testid, or robust relative XPath/CSS over brittle absolute XPath.

> **EXAMPLE / INTERVIEW TIP**
>
> Example (Java): driver.findElement(By.id("firstName")).sendKeys("Asha"); driver.findElement(By.id("email")).sendKeys("asha@example.com"); driver.findElement(By.id("submit")).click(); wait.until(ExpectedConditions.urlContains("confirmation"));. Adapt selectors to the actual HTML and use explicit waits.

11. Submit opens a page with Sign in; clicking it shows username and password labels. How do you automate it?

Wait for the next page or navigation, locate and click Sign in, then wait for the username and password fields to become visible. Assert their visibility and optionally verify the labels/text.

> **EXAMPLE / INTERVIEW TIP**
>
> Example: new WebDriverWait(driver, Duration.ofSeconds(10)).until(ExpectedConditions.elementToBeClickable(By.linkText("Sign in"))).click(); wait.until(ExpectedConditions.visibilityOfElementLocated(By.name("username"))); wait.until(ExpectedConditions.visibilityOfElementLocated(By.name("password"))). If a new tab opens, switch to its window handle first.

12. Write a relative XPath when there are no IDs, names, or values.

Inspect the DOM and anchor the XPath to stable structure, visible text, labels, attributes, or nearby unique elements. Relative XPath starts from a relevant node, commonly with //, rather than the full document path.

> **EXAMPLE / INTERVIEW TIP**
>
> Examples: //label[normalize-space()='Email']/following::input[1]; //button[normalize-space()='Submit']; //div[contains(@class,'login-form')]//input[@type='password']. If no stable identifying information exists, ask for a test attribute or improve the application markup instead of guessing.

13. Write TestNG annotations in execution order.

Typical successful flow: @BeforeSuite → @BeforeTest → @BeforeClass → @BeforeMethod → @Test → @AfterMethod → @AfterClass → @AfterTest → @AfterSuite. Group configuration methods run around relevant group tests. With multiple test methods, BeforeMethod/AfterMethod run around each method.

> **EXAMPLE / INTERVIEW TIP**
>
> This is the common lifecycle order for a single class in one test context; suite XML, inheritance, groups, data providers, parallel execution, and failures can change the observed sequence.

14. If @Test is removed, what exception occurs?

Usually, TestNG simply does not recognize that method as a test method. It does not necessarily throw an exception; the method may not execute as a test, and the run may report zero tests or no tests to run depending on configuration and other tests.

> **EXAMPLE / INTERVIEW TIP**
>
> If another test depends on that method by name, TestNG may report a dependency configuration error or fail to resolve the dependency, depending on version and setup. Exact behavior depends on the test suite.

15. What is the POM in Maven? What is SNAPSHOT?

pom.xml (Project Object Model) describes a Maven project: groupId, artifactId, version, dependencies, plugins, properties, repositories, and build configuration. A version ending in -SNAPSHOT denotes development work that can change before a formal release; repositories may update it as newer snapshots are published.

> **EXAMPLE / INTERVIEW TIP**
>
> Example: <groupId>com.example</groupId>, <artifactId>ui-tests</artifactId>, <version>1.0-SNAPSHOT</version>. Release versions should be stable and immutable according to repository policy.

16. Difference between findElement and findElements.

findElement returns the first matching WebElement and throws NoSuchElementException if no match is found. findElements returns a List<WebElement>; it returns an empty list if there are no matches and does not throw just because none are found.

> **EXAMPLE / INTERVIEW TIP**
>
> Example: WebElement button = driver.findElement(By.id("submit")); List<WebElement> rows = driver.findElements(By.cssSelector("table tbody tr")); Check list size before accessing an index.

17. How do you get the value/text of underlined text inside a frame?

Switch to the correct frame, locate the element, and read getText() for visible text or getAttribute("value") for a form control. Then switch back to default content when finished.

> **EXAMPLE / INTERVIEW TIP**
>
> Example: driver.switchTo().frame(driver.findElement(By.cssSelector("iframe"))); String text=driver.findElement(By.xpath("//u")).getText(); driver.switchTo().defaultContent();. Use the frame's id/name or a WebElement locator as appropriate.

18. What are Maven and Jenkins? Why use Maven?

Maven manages Java builds and dependencies using pom.xml and a standard lifecycle such as validate, compile, test, package, verify, install, and deploy. Jenkins is a CI/CD automation server that can check out source code, run Maven commands, publish reports, and notify teams.

> **EXAMPLE / INTERVIEW TIP**
>
> Typical pipeline: checkout Git → mvn clean test → publish TestNG/Surefire reports → archive artifacts. Maven standardizes builds; Jenkins automates them repeatedly on commits or schedules.

19. Write code to read Excel data using TestNG DataProvider.

Use Apache POI to read workbook cells and return a two-dimensional Object array from a @DataProvider. Close workbook/input stream and handle file paths reliably.

> **EXAMPLE / INTERVIEW TIP**
>
> Example: @DataProvider(name="loginData") public Object[][] data() throws Exception { try (FileInputStream in=new FileInputStream("src/test/resources/data.xlsx"); Workbook wb=WorkbookFactory.create(in)) { Sheet s=wb.getSheet("Login"); int rows=s.getLastRowNum(); Object[][] out=new Object[rows][2]; for(int r=1;r<=rows;r++){ out[r-1][0]=s.getRow(r).getCell(0).toString(); out[r-1][1]=s.getRow(r).getCell(1).toString(); } return out; } } Then @Test(dataProvider="loginData") void loginTest(String user,String pass){...}. For production code, handle blank rows, numeric cells, formulas, and sensitive passwords carefully.

20. What is settings.xml in Maven?

Maven settings.xml contains user/environment-specific configuration, commonly credentials or server IDs, mirrors, proxies, profiles, and repository settings. It is usually located at ~/.m2/settings.xml or supplied via -s path/to/settings.xml.

> **EXAMPLE / INTERVIEW TIP**
>
> It is different from pom.xml: pom.xml is project configuration that is generally checked into source control; settings.xml is usually local or CI environment configuration. Never commit plaintext credentials.

21. How does TestNG work with Maven groupId and version in pom.xml?

Maven resolves TestNG as a dependency from its coordinates: groupId, artifactId, and version. Maven Surefire Plugin can discover and run TestNG tests during the test phase, often using testng.xml for suite configuration.

> **EXAMPLE / INTERVIEW TIP**
>
> Example dependency: org.testng:testng:7.x.x with scope test. Configure maven-surefire-plugin and optionally <suiteXmlFiles><suiteXmlFile>testng.xml</suiteXmlFile></suiteXmlFiles>. Use a currently approved compatible version in your project.

22. What is artifactId?

artifactId is the name of a Maven artifact within a groupId. Together with groupId and version, it identifies a dependency or build output. The packaged artifact often has a filename like artifactId-version.jar.

> **EXAMPLE / INTERVIEW TIP**
>
> Example: com.example:automation-tests:1.0-SNAPSHOT. groupId identifies the organization/project namespace; artifactId identifies the particular module; version identifies its release.

23. Tech Round 1: how should you prepare?

Treat this as a round label rather than a technical question. Prepare a concise project summary, Java fundamentals, Selenium locators/waits/frames/windows, TestNG lifecycle and dependencies, Maven, reporting, CI, SQL/API basics, and one real debugging story.

> **EXAMPLE / INTERVIEW TIP**
>
> For each topic, explain the definition, why it matters, a small example, and how you used it. If asked to code, state assumptions and write compilable core logic.

24. Tell me about your project.

Use a 60–90 second structure: business purpose, application/domain, your role, team/process, tools, automation architecture, test scope, CI/reporting, and measurable outcome. Do not disclose confidential information.

> **EXAMPLE / INTERVIEW TIP**
>
> Template: “I work on a [domain] application used for [business purpose]. I lead QA activities including requirement analysis, test planning, automation design, reviews, defect triage, and release reporting. Our stack includes [truthful tools]. I focus on stable reusable automation, risk-based coverage, and CI execution. One improvement I delivered was [specific truthful result].”

25. Explain static, final, abstract, interface, and exception handling in Java.

static belongs to the class rather than an instance. final prevents reassignment of a variable, overriding of a method, or inheritance from a class. abstract classes can contain abstract and concrete methods and cannot be instantiated. Interfaces define a contract and can have abstract, default, static, and (newer Java) private methods. Exception handling uses try/catch/finally, throw, and throws.

> **EXAMPLE / INTERVIEW TIP**
>
> Example: final int max=10; static int count; abstract class Shape { abstract double area(); } interface Login { void signIn(); } try { ... } catch (IOException e) { ... } finally { ... }. Prefer specific exception types and avoid swallowing exceptions.

26. Explain frame handling, window handling, and Excel read/write in Selenium.

For frames, switch using frame name/id, index, or WebElement and return with defaultContent()/parentFrame(). For windows, capture driver.getWindowHandles(), switch to the target handle, and close only the intended window. Use Apache POI for Excel read/write.

> **EXAMPLE / INTERVIEW TIP**
>
> Frame: driver.switchTo().frame("paymentFrame"); ...; driver.switchTo().defaultContent(). Window: String main=driver.getWindowHandle(); for(String h:driver.getWindowHandles()) if(!h.equals(main)) driver.switchTo().window(h). Excel write: Workbook wb=new XSSFWorkbook(); Sheet sh=wb.createSheet("Results"); sh.createRow(0).createCell(0).setCellValue("PASS"); write via FileOutputStream and close resources.

27. How have you implemented automation in your project?

Explain the process: analyze requirements and identify repeatable/high-value scenarios; select a tool; build reusable driver, page, wait, data, and reporting layers; implement assertions; manage data and environments; integrate with CI; review failures and maintain scripts.

> **EXAMPLE / INTERVIEW TIP**
>
> A credible example: automate smoke/regression flows with Selenium Java, POM, TestNG, Maven, and a reporting library; use explicit waits, stable locators, data-driven tests, Git reviews, and Jenkins runs.
> Describe tools and results from your experience.

28. Explain your framework in detail.

Describe layers: test/scenario layer, page objects, reusable utilities, driver/browser factory, configuration, test data, assertions, logging/reporting, build/dependency management, and CI. Explain how data flows from test to page object to application and how failures are captured.

> **EXAMPLE / INTERVIEW TIP**
>
> A common layout: src/test/java/tests, src/main/java/pages, utils, driver, listeners; src/test/resources/testdata and config; pom.xml; testng.xml; reports. Explain parallel safety, environment selection, secret handling, retries (if used), and test isolation.

29. What difficulties did you face and how did you overcome them?

Choose a real example and answer with Situation, Task, Action, Result. Good technical examples include flaky dynamic elements, stale elements, unstable test data, asynchronous API/UI behavior, environment instability, or long execution time.

> **EXAMPLE / INTERVIEW TIP**
>
> Example structure: “A dynamic table caused intermittent failures. I reproduced the issue, inspected DOM timing, replaced fixed sleeps with explicit waits, improved the locator, and added diagnostic screenshots/logs. The failure rate reduced.” Use your actual situation and measurable result; do not invent metrics.

30. How did you report automation results?

Describe the reporting tool and the information captured: passed/failed/skipped tests, duration, stack traces, screenshots, browser/environment, build number, and links to logs. Explain how reports are shared and how failures are triaged.

> **EXAMPLE / INTERVIEW TIP**
>
> TestNG with Maven Surefire produces XML/text reports under target/surefire-reports by default. ExtentReports or Allure can provide richer HTML reports if configured. Jenkins can publish HTML/XML results and archive screenshots/logs.

31. Tell me about yourself, why leave, and why join CTS?

Keep each answer concise, positive, and specific. Introduce experience, current role, relevant skills, leadership, and target role. For switching, focus on growth, broader responsibilities, technical challenges, or role alignment; avoid criticizing your current employer. For CTS/Cognizant, connect your skills to the job description and the role's domain.

> **EXAMPLE / INTERVIEW TIP**
>
> Template: “I have around [truthful years] in QA and automation, with experience in [actual tools] and leading test delivery. I’m looking for a role with stronger ownership, modern automation, and opportunities to contribute to quality strategy. This position interests me because [specific role/company reason].”

32. Nothing was asked outside what we learned in two months—what does this mean?

This is a candidate's observation, not a technical question. It means interview preparation covering the stated fundamentals was sufficient for that particular interview; it does not guarantee other interviewers will ask the same topics.

> **EXAMPLE / INTERVIEW TIP**
>
> Prioritize understanding and practical exercises over memorizing answers. Be prepared for follow-up questions and to explain trade-offs or debug a short program.

33. What are implicit wait and explicit wait?

Implicit wait tells WebDriver to poll when locating elements for a configured duration. Explicit wait waits for a specific condition, such as visibility or clickability, using WebDriverWait and ExpectedConditions. Avoid mixing implicit and explicit waits because combined timing can be unpredictable.

> **EXAMPLE / INTERVIEW TIP**
>
> Example: driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(5)); Explicit: new WebDriverWait(driver, Duration.ofSeconds(10)).until(ExpectedConditions.elementToBeClickable(By.id("submit"))).click(); Prefer explicit waits for dynamic UI conditions.

34. Write code to launch Google.

Create a WebDriver, navigate to the URL, and close the browser after the test. Ensure the correct browser driver is available or use Selenium Manager in modern Selenium versions.

> **EXAMPLE / INTERVIEW TIP**
>
> Example: WebDriver driver = new ChromeDriver(); try { driver.get("https://www.google.com"); System.out.println(driver.getTitle()); } finally { driver.quit(); }. In a test framework, initialize/quit the driver in setup/teardown and avoid leaving browsers running.

35. Write syntax for relative XPath.

A relative XPath starts from anywhere in the document using // or from a known node. It is generally less brittle than an absolute path such as /html/body/div/... .

> **EXAMPLE / INTERVIEW TIP**
>
> Examples: //input[@id='email']; //button[normalize-space()='Submit']; //label[.='Username']/following::input[1]; //tr[td[normalize-space()='John']]/td[2]. Validate the XPath against the actual DOM.

36. Write XPath for visible text.

Use text() or normalize-space() to match element text. Exact text: //button[normalize-space()='Submit']. Partial text: //a[contains(normalize-space(.),'Sign in')].

> **EXAMPLE / INTERVIEW TIP**
>
> For nested text, normalize-space(.) is often more robust than text() because it includes descendant text. Avoid matching generic text if it appears in multiple elements; scope it to a unique container.

37. How do you achieve a Data-Driven Framework?

Separate test logic from test data. Supply multiple data sets through TestNG @DataProvider, CSV/Excel/JSON/DB, or configuration files. Reuse the same test method for each data row and report each invocation clearly.

> **EXAMPLE / INTERVIEW TIP**
>
> Example: @Test(dataProvider="users") void login(String username,String password){...}. A good design validates data, avoids hard-coded secrets, supports environment-specific data, and cleans up test-created records.

38. What is POM?

POM means Page Object Model in Selenium: a class represents a page/component, owns locators, and exposes actions such as login() or search(). Test classes call these actions and assert outcomes.

> **EXAMPLE / INTERVIEW TIP**
>
> Example: LoginPage has username, password, and submit locators plus enterUsername(), enterPassword(), and clickLogin(). Keep business assertions mostly in tests or clearly designed page/component methods.

39. What is TestNG?

TestNG is a Java testing framework inspired by JUnit and designed for flexible test configuration. It provides annotations, assertions, grouping, dependencies, data providers, parameterization, listeners, and parallel execution.

> **EXAMPLE / INTERVIEW TIP**
>
> It integrates with Maven through a TestNG dependency and Surefire plugin, and can run suites from testng.xml. It is commonly used for Selenium UI automation.

40. Difference between @BeforeTest and @BeforeMethod.

@BeforeTest runs once before test methods belonging to a <test> element in testng.xml. @BeforeMethod runs before each @Test method invocation, including data-provider invocations.

> **EXAMPLE / INTERVIEW TIP**
>
> Use @BeforeTest for setup shared by a test XML section (when appropriate); use @BeforeMethod for per-test setup such as creating a fresh driver or resetting state. Pair with @AfterTest/@AfterMethod.

41. What is a collection? Explain List and Set.

The Java Collections Framework provides interfaces and implementations for groups of objects. List is ordered and permits duplicates; Set represents unique elements and may or may not guarantee iteration order depending on implementation.

> **EXAMPLE / INTERVIEW TIP**
>
> Examples: ArrayList for ordered indexed data; LinkedList for list/deque operations; HashSet for uniqueness without guaranteed order; LinkedHashSet for insertion order; TreeSet for sorted order. Use generics such as List<String>.

42. What are DesiredCapabilities?

DesiredCapabilities historically represented requested browser/platform capabilities for Selenium Grid or remote sessions. Modern Selenium uses browser-specific Options classes such as ChromeOptions, FirefoxOptions, and EdgeOptions, which implement capabilities.

> **EXAMPLE / INTERVIEW TIP**
>
> Example: ChromeOptions options = new ChromeOptions(); options.addArguments("--headless=new"); WebDriver driver = new RemoteWebDriver(gridUrl, options); Prefer Options in modern Selenium; exact capability names depend on browser and Grid version.

43. There are 100 table rows, each with three td values. How do you check duplicate names in 20 rows?

Collect the name column from each row and count occurrences using a Map, or compare the number of values with a Set. If the name is in the second column, extract cell index 1 from each row; adapt to the actual table.

> **EXAMPLE / INTERVIEW TIP**
>
> Example: List<WebElement> rows=driver.findElements(By.cssSelector("tbody tr")); Map<String,Integer> counts=new HashMap<>(); for(WebElement row:rows){ List<WebElement> cells=row.findElements(By.tagName("td")); if(cells.size()>1){String name=cells.get(1).getText().trim(); counts.merge(name,1,Integer::sum);} } counts.forEach((name,count)->{if(count>1) System.out.println(name+" duplicated "+count+" times");}); Filter to the relevant 20 rows if that is the requirement.

44. How do you check duplicate name values in a table?

Read the name column, normalize whitespace/case if the business rule says values are case-insensitive, and count values. A HashSet can detect whether a duplicate has appeared; a Map gives counts.

> **EXAMPLE / INTERVIEW TIP**
>
> Set approach: if (!seen.add(name)) System.out.println("Duplicate: "+name). Ignore blank names if required. Decide whether duplicates like “John” and “ john ” should be treated as the same based on requirements.

45. Difference between List and Set.

List preserves sequence and permits duplicate values; it supports index-based access. Set prevents duplicate elements and has no index-based get method. Ordering depends on implementation.

> **EXAMPLE / INTERVIEW TIP**
>
> ArrayList is a List; HashSet is a Set; LinkedHashSet preserves insertion order; TreeSet sorts according to natural order/comparator. Choose based on required uniqueness and ordering.

46. What is an interface in Java?

An interface defines a contract that implementing classes agree to provide. It supports abstraction and allows a class to implement multiple interfaces. Interfaces can contain constants, abstract methods, and default/static methods; newer Java also supports private helper methods.

> **EXAMPLE / INTERVIEW TIP**
>
> Example: interface Payment { void pay(double amount); } class CardPayment implements Payment { public void pay(double amount){...} }. Call through the interface type to reduce coupling.

47. Difference between abstract class and interface.

An abstract class can hold instance state, constructors, and both abstract and concrete methods; a class can extend only one class. An interface defines a contract and a class can implement multiple interfaces; interfaces do not have ordinary per-instance fields.

> **EXAMPLE / INTERVIEW TIP**
>
> Use an abstract class for closely related classes sharing state or implementation. Use an interface for a capability/contract across potentially unrelated classes. Modern Java interfaces may contain default and static methods.

48. Why is multiple inheritance of classes not supported in Java?

Java disallows extending multiple classes mainly to avoid ambiguity and complexity such as the diamond problem when parent classes define the same method or state. Java supports multiple inheritance of type through interfaces.

> **EXAMPLE / INTERVIEW TIP**
>
> Example: class C cannot extend A, B, but can implement AContract, BContract. If two interfaces provide conflicting default methods, the implementing class must resolve the conflict explicitly.

49. Explain your framework.

Explain the framework's architecture, folder structure, design pattern, driver lifecycle, locator strategy, waits, data management, assertions, reporting, configuration, logging, CI, and how it scales. Mention only components you really use.

> **EXAMPLE / INTERVIEW TIP**
>
> Use a concrete flow: test reads data → page object performs actions → utility waits for state → assertion validates result → listener/report captures status → CI publishes report. Be ready to show a real test and trace it through layers.

50. How do you execute groups in TestNG?

Assign groups using @Test(groups={"smoke"}) and include/exclude them in testng.xml, or configure Maven Surefire group filters. Groups support suites such as smoke, regression, and integration.

> **EXAMPLE / INTERVIEW TIP**
>
> Example XML: <groups><run><include name="smoke"/></run></groups>. Ensure testng.xml is connected to the Maven Surefire configuration if Maven is the execution entry point.

51. Explain your framework (repeated question).

Give a consistent, concise version of your framework explanation rather than a different story. Cover architecture, reusable components, test data, reporting, CI, and one design decision.

> **EXAMPLE / INTERVIEW TIP**
>
> Prepare a 2-minute explanation and a simple diagram: Test layer → Page objects → Selenium/WebDriver → application; shared utilities/config/data and reporting surround the flow.

52. How is data-driven testing achieved?

Provide different input sets to the same test logic through a data provider or external data source. The test method accepts parameters and executes once per data row.

> **EXAMPLE / INTERVIEW TIP**
>
> TestNG @DataProvider is common. External data can come from Excel via Apache POI, JSON, CSV, or databases. Include validation, unique test data, cleanup, and clear report naming.

53. What are class and package?

A class is a blueprint that defines fields, constructors, and methods for objects. A package groups related types, helps organize code, and provides a namespace/access boundary.

> **EXAMPLE / INTERVIEW TIP**
>
> Example: package com.company.pages; public class LoginPage { ... }. The fully qualified name is com.company.pages.LoginPage. Use lowercase package names and organize by responsibility.

54. List packages used in Selenium projects.

Common Java imports include org.openqa.selenium.*, org.openqa.selenium.chrome.ChromeDriver, org.openqa.selenium.support.ui.WebDriverWait, org.openqa.selenium.support.ui.ExpectedConditions, org.openqa.selenium.interactions.Actions, org.openqa.selenium.support.ui.Select, and org.testng.annotations.*.

> **EXAMPLE / INTERVIEW TIP**
>
> Utilities may use java.time.Duration, java.util.List/Map, java.io, Apache POI packages such as org.apache.poi.ss.usermodel.*, and logging/reporting libraries. Name only those actually used in your project.

55. What are class and object?

A class defines the structure/behavior; an object is a runtime instance of that class. Objects have state (fields) and behavior (methods).

> **EXAMPLE / INTERVIEW TIP**
>
> Example: class Car { String model; void start(){...} } then Car c = new Car(); c.model="Sedan"; c.start();. A reference variable holds a reference to an object, not the object itself.

56. Can static variables or methods be overridden?

Static methods are hidden, not overridden, because they belong to the class and method selection is based on the reference/class type. Static variables are also hidden, not overridden. Instance methods can be overridden subject to access and signature rules.

EXAMPLE / INTERVIEW TIP

Example: class A { static void show(){} } class B extends A { static void show(){} } is method hiding. Calling A.show() versus B.show() chooses by class. Use @Override for genuine instance-method overrides.

57. Answer questions about your project.

Prepare the business goal, users, system components, integrations, test environments, release cadence, your ownership, team size, test strategy, automation coverage, defects, and results. Avoid confidential customer data.

EXAMPLE / INTERVIEW TIP

Use real examples: a critical end-to-end flow, a defect caught before release, a flaky test you stabilized, or a CI improvement. Explain what you personally did versus what the team did.

58. Tell me about yourself.

Deliver a tailored 60–90 second introduction: experience, current role, core skills, domain, leadership, achievements, and target role. Do not recite every tool or personal detail.

EXAMPLE / INTERVIEW TIP

Template: “I’m a QA professional with [years] of experience across manual and automation testing. In my current role I handle [responsibilities] using [truthful stack]. I contribute to test strategy, automation framework maintenance, defect analysis, and delivery coordination. I’m now looking for [role] where I can apply this experience and grow in [relevant area].”

59. Rate yourself in Java and Selenium out of 5.

Give an honest rating and justify it with examples. A rating of 4/5 can communicate strong practical experience while acknowledging areas to improve; do not choose a number just to impress.

EXAMPLE / INTERVIEW TIP

Example: “Java 4/5 and Selenium 4/5. I can build and maintain automation, work with collections/OOP, handle waits/frames/windows, and debug failures. I’m continuing to deepen [specific advanced area].” Adjust to your actual level.

60. Explain your project architecture.

Describe application layers and testing architecture: UI/API/services/data stores, environments, automation layers, test data, build pipeline, and reporting. Explain which parts your tests cover and how they integrate.

EXAMPLE / INTERVIEW TIP

A generic automation architecture is tests → page objects/API clients → utilities/config/data → application; Maven builds; TestNG executes; Jenkins triggers; reports/logs are published. Adapt it to your actual system.

61. Explain the design pattern for a login page.

Use Page Object Model: LoginPage encapsulates username/password locators and login actions; LoginTest provides test data and verifies the outcome. Avoid exposing implementation details unnecessarily.

> **EXAMPLE / INTERVIEW TIP**
>
> Java sketch: class LoginPage { private WebDriver driver; private By username=By.id("username"); private By password=By.id("password"); private By signIn=By.id("signIn"); LoginPage(WebDriver d){driver=d;} HomePage login(String u,String p){driver.findElement(username).sendKeys(u); driver.findElement(password).sendKeys(p); driver.findElement(signIn).click(); return new HomePage(driver);} }. Add waits and handle actual navigation/validation.

62. Write a wrapper click method that works with a String and a WebElement.

Overload a utility method: one overload accepts a locator string and resolves it; another accepts a WebElement. Prefer By locators over arbitrary XPath strings when possible, and avoid swallowing exceptions.

> **EXAMPLE / INTERVIEW TIP**
>
> Example: void click(By by){ wait.until(ExpectedConditions.elementToBeClickable(by)).click(); } void click(WebElement element){ wait.until(ExpectedConditions.elementToBeClickable(element)).click(); }. If the requirement literally specifies String, define click(String xpath){ click(By.xpath(xpath)); }. Overloading differs by parameter type.

63. What is an object repository?

An object repository stores UI element locators separately from test logic, commonly in page-object fields, properties files, or a locator map. This centralizes locator maintenance.

> **EXAMPLE / INTERVIEW TIP**
>
> POM classes often act as a structured object repository. For example, By loginButton=By.id("login"). Avoid scattering raw XPath strings throughout test steps.

64. A list of 10 elements shares id=xyz. Get and print all text.

Use findElements, because an ID should ideally be unique in HTML but the locator can still return multiple matches. Iterate over the returned list and print each element's text.

> **EXAMPLE / INTERVIEW TIP**
>
> Example: List<WebElement> items=driver.findElements(By.id("xyz")); for(WebElement item:items) {System.out.println(item.getText());}. If the markup uses repeated elements, a class or data attribute is more appropriate than duplicate IDs.

65. From those elements, select only the fifth and check whether it is a String.

The fifth element is at index 4 because Java collections are zero-based. getText() returns a String, so you can assign it to String or check with instanceof if the value is held as Object.

> **EXAMPLE / INTERVIEW TIP**
>
> Example: List<WebElement> items=driver.findElements(By.id("xyz")); if(items.size()>=5){ String text=items.get(4).getText(); System.out.println(text instanceof String); } else { System.out.println("Fewer than five elements"); }. If the question means checking a variable's runtime type, use Object value=text; value instanceof String.

66. What locators have you used in Selenium?

Selenium's classic By strategies include id, name, className, tagName, linkText, partialLinkText, cssSelector, and xpath. Newer Selenium also supports relative locators such as above/below/near for suitable cases.

> **EXAMPLE / INTERVIEW TIP**
>
> Prefer stable unique IDs or test attributes; use CSS or XPath when needed. Avoid brittle absolute XPath, dynamic indexes, and duplicate-ID assumptions.

67. What is relative XPath and how many types of XPath are there?

Relative XPath starts from a matching node anywhere in the document, commonly using //. Absolute XPath starts at the document root using a path such as /html/body/... and is typically more brittle. There is no universal fixed count of XPath “types”; interviewers often mean absolute versus relative XPath, or XPath axes/functions.

> **EXAMPLE / INTERVIEW TIP**
>
> Examples: //input[@name='email']; //button[contains(.,'Save')]; //label[.='Password']/following::input[1]. Axes include parent, child, ancestor, descendant, following-sibling, preceding-sibling, following, preceding, and self.

68. What is an annotation in TestNG?

An annotation is metadata written with @ that tells TestNG how to treat a method or class. Examples include @Test, @BeforeMethod, @AfterMethod, @DataProvider, @Parameters, and @Listeners.

> **EXAMPLE / INTERVIEW TIP**
>
> Annotations reduce XML/boilerplate and define lifecycle, test grouping, data supply, dependencies, and configuration. The Java compiler and TestNG interpret annotations according to their definitions.

69. How many TestNG annotations are there?

There is no single stable count across every TestNG version because annotations and features can evolve. Common core lifecycle annotations are @BeforeSuite, @AfterSuite, @BeforeTest, @AfterTest, @BeforeClass, @AfterClass, @BeforeMethod, and @AfterMethod. Other common annotations include @Test, @DataProvider, @Parameters, @Factory, @Listeners, @BeforeGroups, and @AfterGroups.

> **EXAMPLE / INTERVIEW TIP**
> In an interview, list the commonly used annotations and explain them instead of asserting a fixed count. Verify the version used in the project.

70. A subclass extends a superclass with four methods—what happens?

The subclass inherits accessible superclass methods unless they are overridden or hidden. The exact output cannot be determined without the actual method signatures, access modifiers, and calls.

> **EXAMPLE / INTERVIEW TIP**
> If the superclass has four public/protected instance methods and the subclass adds none, an instance of the subclass can generally call those inherited methods. Private methods are not inherited as accessible members; constructors are not inherited.

71. And in the superclass…

The question is incomplete without the superclass code. In general, superclass fields, constructors, static methods, and instance methods follow different inheritance/initialization rules. Provide the exact snippet to determine exact output.

> **EXAMPLE / INTERVIEW TIP**
> When tracing output, check static initialization, instance initialization, constructor chaining via super(), overridden-method dispatch, field hiding, and method overloading.

72. What will be the output of the above code?

The referenced code is not included in the question list, so the exact output cannot be determined safely. Do not guess in an interview; ask to see the full snippet.

> **EXAMPLE / INTERVIEW TIP**
> Trace statements in execution order, including static blocks first, then object initialization and constructors, then method calls. For overridden instance methods, runtime object type determines dispatch; static methods and fields are resolved by reference/class type.

73. What is Selenium Grid and how do you implement it?

Selenium Grid runs WebDriver tests on remote machines/browsers and supports parallel execution across browser and OS combinations. Modern Grid commonly uses a standalone server or distributed components such as Router, Distributor, Session Map, and Nodes.

> **EXAMPLE / INTERVIEW TIP**
>
> Start Grid (for example, a compatible Selenium Server standalone JAR), register or configure nodes as needed, then create RemoteWebDriver with the Grid URL and browser Options. In CI, distribute tests carefully and isolate test data. Exact commands depend on Selenium/Grid version and deployment.

74. How is Jenkins useful?

Jenkins automates repeatable CI jobs: checkout, build, test, report publication, artifact archiving, notifications, and scheduled/nightly runs. It can trigger after commits or on a schedule.

> **EXAMPLE / INTERVIEW TIP**
>
> Example pipeline stage: sh 'mvn clean test'. Configure credentials securely, select JDK/Maven, publish Surefire or HTML reports, and archive logs/screenshots. Track flaky tests separately rather than hiding failures.

75. Explain keyword-driven testing.

In keyword-driven testing, test steps are represented by keywords such as OPEN_BROWSER, ENTER_TEXT, CLICK, and VERIFY_TEXT. A driver/engine maps each keyword to reusable code; test cases and data can be maintained in a spreadsheet or external file.

> **EXAMPLE / INTERVIEW TIP**
>
> Benefits include reuse and easier authoring for non-programmers; drawbacks include designing and maintaining a generic keyword engine and debugging indirection. It can be combined with data-driven testing and POM.

76. What is Maven and how have you implemented it?

Maven is a build and dependency management tool. A pom.xml defines project coordinates, dependencies, plugins, properties, and build lifecycle. In a test project, it resolves Selenium/TestNG/POI dependencies and runs tests through Surefire.

> **EXAMPLE / INTERVIEW TIP**
>
> Typical command: mvn clean test. Explain how your project handles environments, versions, testng.xml, profiles (if used), and report generation. Avoid hardcoding local file paths or secrets.

77. How does testng.xml work?

testng.xml defines a suite and one or more <test> sections, with classes/packages, groups, parameters, listeners, and parallel settings. It lets a team select and configure a test run without editing source code.

> **EXAMPLE / INTERVIEW TIP**
>
> Example: <suite name="Regression"><test name="UI"><classes><class name="tests.LoginTest"/> </classes></test></suite>. Configure Maven Surefire to use the file if tests are launched via Maven.

78. How do you run a suite level in XML?

Create testng.xml with a <suite> root and include classes or packages under one or more <test> elements. Run it from the IDE as a TestNG suite or configure Maven Surefire to point to the suite file.

> **EXAMPLE / INTERVIEW TIP**
>
> Example Maven configuration uses <suiteXmlFiles><suiteXmlFile>testng.xml</suiteXmlFile> </suiteXmlFiles>. A command-line invocation can also run TestNG directly with the suite file, depending on project setup.

79. What is an exception?

An exception is an event that disrupts normal program flow. Java exceptions are objects. Checked exceptions must be caught or declared; unchecked exceptions (RuntimeException and its subclasses) generally indicate programming or runtime conditions.

> **EXAMPLE / INTERVIEW TIP**
>
> Use try/catch to handle recoverable cases, finally or try-with-resources to release resources, throw to raise an exception, and throws to declare possible exceptions. Catch the narrowest meaningful type and preserve diagnostic information.

80. What will happen in the above code?

The code referred to is not included, so the exact result cannot be established. To answer accurately, review the complete code and any relevant inputs.

> **EXAMPLE / INTERVIEW TIP**
>
> Check control flow, variable initialization, exception paths, loop conditions, object state, and whether the exception is caught or propagated. Share the snippet to get an exact output trace.

81. What happens if a WebElement is not found?

findElement throws NoSuchElementException if no matching element is found at lookup time. With an implicit wait, it retries locating until the timeout expires. findElements returns an empty list instead.

> **EXAMPLE / INTERVIEW TIP**
>
> For dynamic pages, wait for the expected condition and check locator correctness. Do not catch and ignore the exception unless there is a meaningful recovery path.

82. When does an exception occur and when does else execute?

Java has no general try/except/else construct. In Java, catch executes when a matching exception is thrown in the try block; finally normally executes whether an exception occurs or not. An else block is a feature of languages such as Python, not Java exception syntax.

> **EXAMPLE / INTERVIEW TIP**
>
> If the interviewer shows pseudocode with else, clarify the language. In Java: try { ... } catch (NoSuchElementException e) { ... } finally { ... }. A normal if/else block is unrelated to exception handling.

83. What exceptions have you faced in a project? Explain try/catch.

Common Selenium exceptions include NoSuchElementException, TimeoutException, StaleElementReferenceException, ElementClickInterceptedException, ElementNotInteractableException, and NoSuchFrameException. Mention only those you actually encountered and explain the cause and fix.

> **EXAMPLE / INTERVIEW TIP**
>
> Example: catch TimeoutException to capture screenshot/logs and fail the test with context. Use explicit waits for expected state, refresh stale locators by re-finding elements, and fix overlays/interception rather than blindly retrying.

84. What is SoapUI and service testing?

SoapUI is a tool for testing APIs and web services, especially SOAP and REST. Service testing validates request/response behavior, status codes, headers, payloads, schemas, authentication, negative cases, and integration rules without relying solely on the UI.

> **EXAMPLE / INTERVIEW TIP**
>
> For SOAP, inspect WSDL and XML envelopes; for REST, validate HTTP methods, status, JSON/XML payloads, headers, and error handling. Automate repeatable API checks and manage test data/environment carefully.

85. Where have you executed SQL in your project?

Describe the real use case, such as validating UI data against a database, checking test data, investigating defects, verifying batch/ETL outcomes, or confirming records after API operations. Name the database only if accurate.

> **EXAMPLE / INTERVIEW TIP**
>
> Example query: SELECT status, COUNT(*) FROM appointments GROUP BY status; or SELECT * FROM customer WHERE customer_id = ?. Use parameterized queries, least-privilege access, and non-production data according to policy. Do not claim direct database access if you did not have it.

86. What are function/method overloading and overriding?

Overloading uses the same method name with different parameter lists in the same class or hierarchy; it is resolved at compile time. Overriding redefines an inherited instance method with a compatible signature; runtime dispatch selects the implementation based on the object type.

EXAMPLE / INTERVIEW TIP

Overloading example: click(By by), click(WebElement e). Overriding example: class Animal { void speak() {} } class Dog extends Animal { @Override void speak(){} }. Return type alone cannot distinguish overloads.

87. Do you want to execute two or three tests and go home, or take challenges?

This is a behavioral question about ownership, learning, and accountability. Say that you value reliable delivery and are willing to take on challenges, while prioritizing risk and quality rather than equating long hours with commitment.

EXAMPLE / INTERVIEW TIP

Example: “I prefer taking ownership of meaningful quality challenges—improving coverage, stabilizing automation, investigating failures, and helping the team deliver. I also plan work realistically and communicate risks early.” Add a genuine example.

88. Do you know mobile automation?

Mobile automation tests native, hybrid, or mobile web apps. Appium is a widely used option that exposes WebDriver-style APIs and supports Android and iOS, subject to environment/tooling requirements.

EXAMPLE / INTERVIEW TIP

Explain capabilities, device/emulator setup, app package/activity or bundle ID, locators, waits, gestures, permissions, and device-farm execution. If your experience is theoretical, say so honestly rather than claiming hands-on production use.

89. Give a brief introduction about yourself.

Use the same focused introduction as question 58, tailored to the job description. State years of experience accurately, key tools, current responsibility, leadership contribution, and the next role you seek.

EXAMPLE / INTERVIEW TIP

Keep it under 90 seconds and finish with why your background fits the role. Avoid personal history unrelated to the job.

90. How have you implemented TestNG annotations in your project?

Explain setup/teardown, test methods, grouping, data providers, and listeners as used in your project. @BeforeMethod may initialize test state; @Test runs scenarios; @AfterMethod captures results/cleans up; listeners can capture failures and screenshots.

EXAMPLE / INTERVIEW TIP

Illustrative lifecycle: @BeforeClass creates shared configuration; @BeforeMethod starts a driver; @Test executes a scenario; @AfterMethod captures failure details and quits driver. Ensure driver lifecycle matches parallel execution and your framework design.

91. Explain Maven projects and their importance.

Maven standardizes project structure and build lifecycle, manages transitive dependencies, makes builds repeatable, and integrates with CI. Developers use pom.xml to declare dependencies/plugins instead of manually managing JAR files.

> **EXAMPLE / INTERVIEW TIP**
>
> Mention commands such as mvn clean test, mvn package, and profiles if relevant. Version pinning and reproducible configuration reduce “works on my machine” issues.

92. Explain Jenkins and its implementation.

Describe a Jenkins job or pipeline that checks out code, selects the JDK/Maven runtime, runs tests, publishes results, archives artifacts, and notifies stakeholders. Explain triggers such as pull requests, commits, or nightly schedules.

> **EXAMPLE / INTERVIEW TIP**
>
> Keep credentials in Jenkins Credentials or a secret manager. Make reports and logs accessible; define failure thresholds and avoid marking a build successful when critical tests fail.

93. Explain wrapper classes and POM; how are they implemented in methods?

Wrapper classes (Integer, Double, Boolean, Character, etc.) represent primitive values as objects and support collections/generics, conversion, and utility methods. POM is the Page Object Model pattern for UI automation. They are separate concepts.

> **EXAMPLE / INTERVIEW TIP**
>
> Example wrapper: Integer count = Integer.valueOf("10"); int n=count; (autoboxing/unboxing). POM example: loginPage.login(user, pass). If “wrapper method” means a reusable Selenium helper, describe a click/type/wait method separately.

94. How do you implement TestNG with POM?

Create page classes for UI actions and TestNG test classes for scenarios/assertions. Use TestNG annotations for setup/teardown, @DataProvider for test data, groups for suites, and listeners for reporting. Maven/Surefire runs the suite.

> **EXAMPLE / INTERVIEW TIP**
>
> Flow: LoginTest → LoginPage.login() → HomePage/expected state → TestNG assertion. Keep locators/actions in page objects and test intent/assertions in test classes; use driver factories and thread-safe per-test state for parallel runs.

95. Where are reports stored in a Maven project?

Maven Surefire reports are typically stored under target/surefire-reports, including XML and text files. Other HTML reporting tools store reports in configured locations, such as target/site or a project-specific report folder.

> **EXAMPLE / INTERVIEW TIP**
> The exact path depends on plugin and framework configuration. Jenkins may archive or publish these reports elsewhere; mention the real path used by your project.

96. What are SoapUI and SoapUI Pro?

SoapUI is used for functional testing of SOAP and REST services. “SoapUI Pro” has historically referred to commercial/ReadyAPI capabilities, offering enhanced features such as data-driven testing, advanced reporting, and team-oriented workflows depending on product version/licensing.

> **EXAMPLE / INTERVIEW TIP**
> Focus on what your team actually used. Core service-testing concepts—requests, assertions, environments, authentication, data sources, and regression suites—matter more than product branding.

97. Explain Grid hub and node implementation and why use Grid.

In older Selenium Grid architecture, the Hub receives session requests and routes them to registered Nodes that provide browser slots. Grid 4 uses a more distributed architecture with components such as Router, Distributor, Session Map, Event Bus, and Nodes; standalone mode packages them together.

> **EXAMPLE / INTERVIEW TIP**
> Use Grid to run tests remotely and in parallel across browser/OS combinations. Configure browser capabilities, register nodes (for older Grid), point RemoteWebDriver to the Grid URL, and ensure tests/data are isolated. Explain which Grid version you used.

98. Do you know mobile automation?

Answer honestly about hands-on experience. For Appium, describe the WebDriver-based model, Android/iOS setup, capabilities, device/emulator, app lifecycle, locators, gestures, and reporting.

> **EXAMPLE / INTERVIEW TIP**
> If you have not used it on a real project, say: “I understand the concepts and have studied Appium, but my primary hands-on automation has been web/API testing. I’m prepared to extend my experience.”

99. Why do you want to switch companies?

Give a positive, forward-looking answer based on role scope, learning, ownership, technology, domain, or career growth. Avoid complaints about managers, colleagues, or compensation as the only reason.

> **EXAMPLE / INTERVIEW TIP**
>
> Example: “I’m grateful for what I’ve learned, and I’m looking for a role with broader QA ownership, opportunities to improve automation strategy, and alignment with my longer-term goals. This role appears to offer that through [specific reason].”

100. What is your total experience and relevant experience?

State total experience and relevant experience separately and accurately. Relevant experience should be broken down by the skills asked for, and the numbers should match your resume and employment dates.

> **EXAMPLE / INTERVIEW TIP**
>
> Template: “My total experience is [X years]. Relevant experience: [Y years] in Selenium Java, [Z years] in Playwright, [A years] in API automation, and [B years] in QA leadership.” Do not double-count overlapping years unless you explain that skills were used concurrently.