Altimetrik Selenium Interview Questions with Answers (2026) | Saravana Tech

65 interview questions with explanations, practical examples, code snippets, and sample HR answers

How to use this guide: learn the concepts, practise explaining them aloud, and adapt the project/HR answers to your actual responsibilities. Examples use Java and Selenium 4-style APIs. Exact APIs and browser setup may vary with your Selenium, browser, and driver versions.

Core interview answers

1. Interface vs abstract class

An interface defines a contract: what an implementing class must provide. An abstract class can define both incomplete behaviour (abstract methods) and shared implementation/state (concrete methods and fields). A class can implement multiple interfaces but can extend only one class. Use an interface for capabilities shared by otherwise unrelated classes; use an abstract class when related classes share state or common code.

```
interface Printable { void print(); }
abstract class Report {
    protected String title;
    Report(String title) { this.title = title; }
    abstract void generate();
    void showTitle() { System.out.println(title); }
}
class PdfReport extends Report implements Printable {
    PdfReport(String title) { super(title); }
    @Override void generate() { System.out.println("Generate PDF"); }
    @Override public void print() { System.out.println("Print PDF"); }
}
```

2. String vs StringBuffer

String is immutable: an operation that appears to modify it creates another String. StringBuffer is mutable and synchronizes its methods, making it thread-safe for shared use but often slower than StringBuilder. Use String for ordinary text; use StringBuffer when a mutable buffer must be shared safely across threads; use StringBuilder for most single-threaded string construction.

```
String s = "Test";
s.concat("ing");
System.out.println(s);              // Test
StringBuffer buffer = new StringBuffer("Test");
buffer.append("ing");
System.out.println(buffer);         // Testing
```

3. String a = ""; vs String a = " ";

An empty string ("") contains zero characters and has length 0. A string containing one space (" ") has length 1. Both are non-null strings. This distinction matters in validation: trim().isEmpty() detects empty or whitespace-only text, while isEmpty() alone detects only zero-length text.

```
String a = "";
String b = " ";
System.out.println(a.length());        // 0
System.out.println(b.length());        // 1
System.out.println(b.trim().isEmpty()); // true
```

4. JVM vs JRE

JVM (Java Virtual Machine) executes Java bytecode and provides runtime services such as memory management and garbage collection. JRE (Java Runtime Environment) traditionally means the JVM plus the libraries and files needed to run Java applications. JDK (Java Development Kit) includes development tools such as javac plus runtime components. Modern Java distributions may not package a separate JRE, but the conceptual distinction remains useful.

```
javac Hello.java   // compiler: .java source -> .class bytecode
java Hello         // launcher runs the compiled class on a JVM
```

5. HashSet vs Hashtable

HashSet is a Set implementation that stores unique elements and has no guaranteed iteration order; it permits one null element. Hashtable is a legacy Map implementation that stores key-value pairs, is synchronized, and does not permit null keys or values. They solve different problems: use HashSet for uniqueness and HashMap/ConcurrentHashMap for key-value lookup in modern code, depending on thread-safety needs.

```
Set<String> names = new HashSet<>();
names.add("Ana"); names.add("Ana");
System.out.println(names.size()); // 1

Hashtable<Integer, String> map = new Hashtable<>();
map.put(1, "Ana");
System.out.println(map.get(1));   // Ana
```

6. ArrayList vs array

An array has a fixed length once created and can store primitives or references. ArrayList is a resizable collection of objects, offers methods such as add/remove/contains, and uses generics for type safety. Arrays can be more direct and compact for fixed-size data; ArrayList is usually more convenient when the number of elements changes.

```
int[] numbers = new int[3];
numbers[0] = 10;

List<String> names = new ArrayList<>();
names.add("Ana");
names.add("Ben");
```

7. Use of Set in the Collections Framework

A Set models a collection of unique elements, so adding an element already present does not create a duplicate. HashSet is generally used for fast membership checks without order; LinkedHashSet preserves insertion order; TreeSet keeps elements sorted. Equality and hashCode must be implemented consistently for custom objects used in hash-based sets.

```
Set<String> browsers = new LinkedHashSet<>();
browsers.add("Chrome");
browsers.add("Firefox");
browsers.add("Chrome");
System.out.println(browsers); // [Chrome, Firefox]
```

8. Method overloading vs method overriding

Overloading means same method name with different parameter lists, usually in the same class; it is resolved at compile time. Overriding means a subclass provides a compatible implementation of an inherited instance method; the chosen implementation is determined at runtime. Return type alone cannot distinguish overloaded methods. Static methods are hidden rather than overridden.

```
class Search {
    void find(String text) {}
    void find(String text, int page) {} // overload
}
class Base { void open() { System.out.println("Base"); } }
class Child extends Base {
    @Override void open() { System.out.println("Child"); } // override
}
```

9. Polymorphism

Polymorphism means one reference type can represent objects of different concrete classes. Runtime polymorphism commonly uses method overriding: the method implementation is chosen according to the actual object. This supports extensible code and reduces dependency on concrete implementations.

```
WebDriver driver = new ChromeDriver();
driver.get("https://example.com");
// The reference is WebDriver; the object is ChromeDriver.
```

10. Assert vs verify vs waitFor

An assertion checks an expected condition and normally fails the test immediately if it is false. In Selenium with TestNG, Assert is the usual hard assertion; SoftAssert records failures and reports them when assertAll() is called. 'Verify' is often an informal term for checking an expected result rather than a separate Selenium API. waitFor usually means waiting until a condition becomes true before interacting; in Selenium use WebDriverWait/ExpectedConditions rather than fixed sleeps when possible.

```
WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(10));
WebElement submit = wait.until(
    ExpectedConditions.elementToBeClickable(By.id("submit")));
Assert.assertTrue(submit.isDisplayed(), "Submit should be visible");
```

11. super keyword

super refers to the immediate parent class. It can call a parent constructor (super(...)), access an accessible parent field/method, or invoke a parent implementation that has been overridden. A constructor call to super(...) must be the first statement in a constructor.

```
class Parent {
    Parent(String name) { System.out.println(name); }
    void display() { System.out.println("Parent"); }
}
class Child extends Parent {
    Child() { super("QA"); }
    @Override void display() { super.display(); System.out.println("Child"); }
}
```

12. Find links and manipulate them in Selenium

Locate links with By.linkText/partialLinkText when the visible text is stable, or use CSS/XPath. For an anchor, getText() reads visible text, getAttribute("href") reads its destination, and click() activates it. Prefer stable IDs or test attributes when available; avoid broad selectors that match several elements. Check count/visibility before acting if the page contains repeated links.

```
List<WebElement> links = driver.findElements(By.tagName("a"));
for (WebElement link : links) {
    System.out.println(link.getText() + " -> " + link.getAttribute("href"));
}
driver.findElement(By.linkText("Contact Us")).click();
```

13. Print data in an IDE — echo/logging

In a Java program, System.out.println() prints to the console. In a Cucumber feature file, the keyword is 'Then/When/And', not echo; 'echo' is commonly a shell command or a command in other test tools. For framework diagnostics, prefer a logger (for example SLF4J/Log4j) because it supports levels, timestamps, and file output.

```
System.out.println("Selected username: " + username);
// Example logger, assuming configured:
logger.info("Login step started");
```

14. Is String immutable?

Yes. A Java String object's character sequence cannot be changed after creation. Methods such as replace(), concat(), and toUpperCase() return a new String when a change is needed. Immutability helps safe sharing, caching, and use as map keys, but repeated concatenation in a loop can create many temporary objects; use StringBuilder for that pattern.

```
String value = "java";
String upper = value.toUpperCase();
System.out.println(value); // java
System.out.println(upper); // JAVA
```

15. STLC — Software Testing Life Cycle

Typical STLC phases are: requirement analysis, test planning, test case design/development, test environment setup, test execution, defect reporting/retesting/regression, and test closure. Each phase has entry criteria, activities, deliverables, and exit criteria. In Agile, these activities happen iteratively within a sprint rather than only once at the end.

Example: for a login feature, analyse lockout/password rules; estimate and plan; write positive/negative test cases; prepare test data/environment; execute; log defects and retest fixes; then report coverage and lessons learned.

16. A showstopper is found one day before delivery — what do you do?

First reproduce it and confirm scope, severity, business impact, and whether it blocks a critical user journey. Immediately communicate facts to the QA lead, developer, product owner, and release manager; do not hide or downgrade the issue to protect the date. Capture evidence, logs, environment, and steps. Work with developers on root cause and fix; validate the fix and run targeted regression. The accountable release stakeholders decide go/no-go based on risk, not QA alone. If unresolved, recommend delaying or disabling the affected feature where safe.

Sample interview wording: “I would establish reproducibility and impact, raise a blocker with evidence, notify stakeholders immediately, validate the fix, run focused regression, and provide a risk-based release recommendation. I would not sign off while a critical user journey remained unsafe.”

17. Ways to declare a String in Java

Common ways include a string literal, using new String(...), and creating a String from a character array or another data source. Literals are usually preferred because the JVM can reuse interned literal instances. new String("Java") explicitly creates a distinct String object and is rarely needed.

```
String a = "Java";
String b = new String("Java");
char[] chars = {'J', 'a', 'v', 'a'};
String c = new String(chars);
```

18. Use of javac and java

javac is the Java compiler: it converts source files (.java) into bytecode (.class). java launches a class or JAR and runs its bytecode on a JVM. The JDK must be installed and its bin directory available on PATH, or commands must be run with full paths.

```
javac Hello.java
java Hello
// If package com.demo: compile with correct directory/classpath,
// then run: java com.demo.Hello
```

19. Have you used data structures in your project?

Answer honestly and tie the data structure to a practical task. In a Selenium framework, List<WebElement> is used to collect table rows or matching elements; Map<String,String> can hold test data or expected values; Set<String> is useful for unique window handles or duplicate detection; Queue can model ordered work in suitable utilities. Explain why the structure was selected and how it improved clarity or performance.

```
Set<String> windowHandles = driver.getWindowHandles();
List<WebElement> rows = driver.findElements(By.cssSelector("table tbody tr"));
Map<String, String> testData = new HashMap<>();
testData.put("username", "test.user");
```

20. How do you rate yourself in Java and WebDriver?

Give a defensible rating, not an inflated one. Separate strengths from areas being developed and support the rating with examples. For instance: “I would rate my Java 7.5/10 and Selenium WebDriver 8/10. I am comfortable with OOP, collections, exception handling, waits, locators, POM, and debugging automation failures. I continue improving advanced Java concurrency and framework architecture.” Adjust the numbers to your real capability and be prepared for follow-up coding.

21. Tell me about yourself

Use a 60–90 second structure: experience and current role; core technical skills; project/framework responsibilities; leadership and quality impact; role you are seeking. Keep it relevant and avoid personal details unless asked. Adapt this sample to your real experience:

“I have around 13 years of experience in software testing across manual and automation testing. My technical experience includes Java, Selenium WebDriver, Cucumber, TestNG, API automation, and Playwright, along with Agile delivery. I have worked on designing and maintaining automation suites, analysing requirements, planning test coverage, triaging defects, and coordinating with developers and business stakeholders. In my current QA lead/project lead responsibilities, I focus on risk-based testing, reliable automation, and timely quality reporting. I am looking for a role where I can contribute both hands-on automation expertise and team-level quality ownership.”

22. Explain the framework used in your project

Based on the stated framework context, a suitable explanation is a Cucumber + Page Object Model (POM) hybrid framework. Cucumber feature files express scenarios in business-readable Gherkin; step definitions map steps to Java methods; page classes encapsulate locators and UI actions; hooks handle setup/teardown; a driver manager/config layer creates browser sessions; test data and configuration are externalised; TestNG/JUnit and Maven execute tests; reports capture results; CI can run the suite. Explain only components actually used in your project.

```
Feature: Login
  Scenario: Valid user logs in
    Given the user opens the login page
    When the user enters valid credentials
    Then the home page is displayed

// Step definition calls LoginPage methods;
// LoginPage owns username/password/submit locators and actions.
```

23. Rate yourself in Selenium

Choose a realistic rating and connect it to evidence. Example: “I rate myself 8/10 in Selenium. I can design locators, handle waits, alerts, frames, windows, tables, dropdowns, actions, screenshots, and build POM-based tests. I am strengthening my expertise in advanced grid/cloud execution, accessibility, and flaky-test diagnostics.” Do not claim a skill you cannot demonstrate in a coding round.

24. What is an interface? Give an example in Selenium

An interface defines method contracts that implementing classes fulfil. Selenium's WebDriver is an interface. Browser-specific drivers such as ChromeDriver and FirefoxDriver implement WebDriver, so test code can use the common API and switch browsers with less change.

```
WebDriver driver = new ChromeDriver();
driver.get("https://example.com");
String title = driver.getTitle();
driver.quit();
```

25. What is an abstract method? Can we write code inside it?

An abstract method is declared without a method body and must be implemented by a concrete subclass, unless that subclass remains abstract. You cannot place executable method-body code inside an abstract method. An abstract class can also contain normal methods with implementations. Interfaces can contain abstract methods and, in modern Java, default/static methods with bodies.

```
abstract class BrowserPage {
    abstract void verifyPageLoaded(); // no body
    void logPageName() { System.out.println("Page"); }
}
```

26. Read data from a dynamic web table

First wait for the table or target row to appear. Locate rows and cells relative to the table, then iterate through them and match a unique key (such as order ID) instead of relying on a fixed row index. Dynamic tables may paginate, virtualise rows, or load asynchronously; handle pagination and wait for the target data. Use getText() for visible cell text.

```
WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(10));
wait.until(ExpectedConditions.visibilityOfElementLocated(By.cssSelector("table tbody")));
List<WebElement> rows = driver.findElements(By.cssSelector("table tbody tr"));
for (WebElement row : rows) {
    List<WebElement> cells = row.findElements(By.cssSelector("td"));
    if (!cells.isEmpty() && cells.get(0).getText().trim().equals("ORD-1002")) {
        System.out.println("Status: " + cells.get(3).getText());
        break;
    }
}
```

27. Right-click and double-click a button

Use Selenium's Actions class for context-click (right-click) and doubleClick. Ensure the target is visible and interactable first. A right-click may open a browser context menu that Selenium cannot inspect like normal DOM content; if the application has a custom context menu, locate its DOM element after the action.

```
WebElement button = driver.findElement(By.id("actionButton"));
new Actions(driver).contextClick(button).perform(); // right-click
new Actions(driver).doubleClick(button).perform();   // double-click
```

28. Available locators and XPath syntax

Selenium's standard locator strategies are id, name, className, tagName, linkText, partialLinkText, cssSelector, and xpath. Prefer stable unique IDs or test attributes; CSS is often concise; XPath is useful for text matching and navigating relationships. Avoid brittle absolute XPath and indexes unless justified.

```
driver.findElement(By.id("username"));
driver.findElement(By.name("password"));
driver.findElement(By.className("login-button"));
driver.findElement(By.cssSelector("button[type='submit']"));
driver.findElement(By.xpath("//button[normalize-space()='Login']"));
// Relative XPath: //input[@name='email']
```

29. Is WebDriver a class or interface?

WebDriver is an interface in Selenium. ChromeDriver, FirefoxDriver, EdgeDriver, and other browser drivers implement it. This allows test code to depend on the interface rather than one browser-specific implementation.

```
WebDriver driver = new ChromeDriver();
```

30. List vs Set

List is ordered and permits duplicates; it supports index-based access. Set represents unique elements and does not provide index-based access. HashSet does not guarantee iteration order, LinkedHashSet preserves insertion order, and TreeSet sorts elements. Choose based on whether ordering, duplicates, or fast membership checks matter.

```
List<String> list = new ArrayList<>(Arrays.asList("A", "A", "B"));
Set<String> set = new LinkedHashSet<>(list);
System.out.println(list.size()); // 3
System.out.println(set.size());  // 2
```

31. Tell me about yourself — alternate concise version

Use a version suited to the role. For a hands-on automation role, emphasise coding and framework design; for a QA Lead role, emphasise planning, risk, people coordination, and release decisions. Keep the answer consistent with your resume and avoid memorised claims that you cannot support with examples.

“I combine hands-on automation experience with QA coordination. I have worked with Java and Selenium, Cucumber, TestNG, API testing, and modern browser automation. I translate requirements into test coverage, help maintain reusable automation components, analyse failures, and work with cross-functional teams to resolve defects. I value maintainable tests and risk-based quality decisions.”

32. Which framework have you used? Explain

Use the same project-specific answer as Question 22, then describe one real scenario end to end: feature file → step definition → page object → driver/wait utility → assertion → report. Explain how test data, configuration, parallel execution, and CI are managed only if your project genuinely includes them.

```
Example flow:
Feature scenario -> StepDefinition.login() -> LoginPage.enterUsername()
-> LoginPage.enterPassword() -> LoginPage.clickSubmit()
-> HomePage.isDisplayed() -> assertion -> report.
```

33. Rate yourself in Selenium and Java

Give two separate ratings and explain the evidence for each. Example: “Java 7.5/10; Selenium 8/10.” For Java, mention OOP, collections, strings, exceptions, and coding. For Selenium, mention locators, waits, browser interactions, POM, debugging, and test stability. Ratings are subjective; consistency and demonstrated skill matter more than the number.

34. What is an interface? Can we create objects for an interface?

An interface specifies behaviour and cannot be instantiated directly, so `new WebDriver()` is invalid. You can declare an interface reference and assign it an instance of an implementing class. An anonymous class or lambda can implement certain interfaces, but it still creates an implementation object rather than an interface object.

```
WebDriver driver = new ChromeDriver(); // valid
// WebDriver driver = new WebDriver(); // invalid
```

35. Method overloading — Selenium example and project use

Overloading is having methods with the same name but different parameter lists. It can make helper APIs convenient, for example one click helper accepting a locator or a WebElement. Selenium itself has overloaded methods such as findElement(By) and various WebDriverWait/Actions APIs, but in project discussions, explain your own actual helper implementation if you used one.

```
// Example utility overloads
void click(By locator) { driver.findElement(locator).click(); }
void click(WebElement element) { element.click(); }

// Usage
click(By.id("submit"));
click(driver.findElement(By.id("cancel")));
```

36. Third way to inherit/use parent properties besides objects and extends

The likely interview concept is interface implementation: a class uses `implements` to adopt an interface contract, rather than `extends` a class. However, implements is not inheritance of class state in the same sense as extending a class. Java also supports composition, where one class contains an instance of another and delegates work. Ask for clarification if the interviewer means a specific inheritance pattern.

```
interface Auditable { void audit(); }
class LoginService implements Auditable {
    public void audit() { System.out.println("Audit login"); }
}
```

37. Initialize a Chrome browser

In Selenium 4, install compatible Selenium and Chrome versions and let Selenium Manager resolve the driver in common setups. Create the driver, optionally configure browser options, navigate, and always quit in teardown. In frameworks, keep driver creation in a driver factory or hooks rather than duplicating it in every test.

```
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.chrome.ChromeOptions;

ChromeOptions options = new ChromeOptions();
WebDriver driver = new ChromeDriver(options);
driver.manage().window().maximize();
driver.get("https://example.com");
// At end of test:
driver.quit();
```

38. Challenges faced while automating a project

Choose two or three genuine examples and explain symptom, root cause, action, and outcome. Common examples include flaky tests due to timing, unstable/dynamic locators, test-data dependencies, OTP/captcha restrictions, asynchronous UI overlays, parallel-test isolation, browser/version changes, and slow suites. Good responses show root-cause analysis rather than adding Thread.sleep everywhere. Use explicit waits, stable selectors, isolated test data, logging/screenshots, retries only for known transient infrastructure issues, and collaboration with developers.

Example: “A page loaded asynchronously, so tests intermittently clicked before a button became enabled. I replaced fixed sleeps with an explicit wait for the enabled/clickable condition, improved failure logs, and reduced flaky failures.”

39. Different collections available in Java

The Java Collections Framework includes List (ArrayList, LinkedList, Vector), Set (HashSet, LinkedHashSet, TreeSet), Queue/Deque (PriorityQueue, ArrayDeque, LinkedList), and Map (HashMap, LinkedHashMap, TreeMap, Hashtable, ConcurrentHashMap). Map is part of the Collections Framework but does not extend the Collection interface. Choose implementations based on ordering, duplicates, lookup needs, sorting, and concurrency.

Examples: List for ordered test steps; Set for unique window handles; Map for named test data; Queue for first-in-first-out processing.

40. POM code for login page navigating to home page

Create a LoginPage class that owns locators and actions, a HomePage class that verifies the destination, and a test that orchestrates them. Use explicit waits and assertions. The example is intentionally compact; adapt locators to the actual application and add driver lifecycle handling in your framework.

```
// LoginPage.java
public class LoginPage {
    private final WebDriver driver;
    private final By username = By.id("username");
    private final By password = By.id("password");
    private final By submit = By.id("submit");

    public LoginPage(WebDriver driver) { this.driver = driver; }
    public void login(String user, String pass) {
        driver.findElement(username).sendKeys(user);
        driver.findElement(password).sendKeys(pass);
        driver.findElement(submit).click();
    }
}

// HomePage.java
```

```
public class HomePage {
    private final WebDriver driver;
    private final By heading = By.id("homeHeading");
    public HomePage(WebDriver driver) { this.driver = driver; }
    public boolean isDisplayed() {
        return new WebDriverWait(driver, Duration.ofSeconds(10))
            .until(ExpectedConditions.visibilityOfElementLocated(heading)).isDisplayed();
    }
}

// Test
LoginPage login = new LoginPage(driver);
login.login("demo", "secret");
Assert.assertTrue(new HomePage(driver).isDisplayed());
```

41. Why are you leaving your current organisation?

Keep the answer positive and future-focused. Avoid criticising managers, colleagues, or the company. Mention growth, broader responsibility, deeper automation/quality engineering work, or alignment with long-term goals. Do not claim a reason that is not true.

Sample: “I appreciate the experience and opportunities I have had in my current organisation. I am now looking for a role with broader ownership and opportunities to contribute to automation strategy, quality engineering, and team delivery. I want to apply my experience while continuing to grow technically and professionally.”

42. Current and expected CTC

State the current CTC accurately, clarify fixed versus variable components if asked, and give an expected range based on role, responsibilities, market, and benefits. Your previously mentioned current CTC was around ₹18.9 LPA; verify the latest figure and whether it includes variable pay before using it in an interview. Avoid inventing an expected figure without considering the offer and your target.

Sample wording: “My current CTC is ₹[accurate amount] LPA, including [fixed/variable details]. For the next move, I am looking for a market-aligned package based on the role and responsibilities, and I am open to discussing the complete compensation structure.”

43. Tell me about your family background

Keep it brief and comfortable: family members and general support, if you wish. Interviewers should not need sensitive personal details. Then return the conversation to your professional suitability. Do not feel obliged to disclose private financial, medical, or other sensitive information.

Sample: “I come from a supportive family. They encourage my professional growth, and I balance family responsibilities with my work commitments.”

44. Any constraints for a 2 PM–10 PM shift?

Answer honestly. If you can work the shift, confirm that clearly and ask whether it is fixed or rotational and whether transport or occasional schedule changes apply. If you have a genuine constraint, state it professionally and propose a workable alternative; do not agree if you cannot sustain it.

Sample if available: “I do not have a constraint with a 2 PM–10 PM shift. I would like to understand whether the schedule is fixed or rotational and what the team’s support arrangements are.”

45. Explain your Selenium experience in projects

Describe project context, what you automated, the stack, your contribution, and the result. Include examples such as regression/smoke suites, reusable POM components, data-driven scenarios, CI execution, reporting, and defect triage only if true. Quantify outcomes only when you have reliable measurements. Explain how you handle flaky tests and maintain scripts as the UI changes.

Sample structure: “In my project, I used Java, Selenium WebDriver, Cucumber, and TestNG in a POM/hybrid framework. I automated high-value regression flows, maintained reusable page methods, used explicit waits, reviewed failures, and collaborated with developers on defects. I also supported execution and reporting so the team could make release decisions based on evidence.”

46. Program to remove duplicate characters from a String

Use a LinkedHashSet to retain the first occurrence order while removing repeated characters. Decide how to treat spaces and case; the example preserves both and is case-sensitive. For Unicode code points beyond basic characters, iterate over code points rather than char values.

```
String input = "programming";
StringBuilder result = new StringBuilder();
Set<Character> seen = new LinkedHashSet<>();
for (char ch : input.toCharArray()) {
    if (seen.add(ch)) result.append(ch);
}
System.out.println(result); // progamin
```

47. Palindrome String program

A palindrome reads the same forwards and backwards. Decide whether comparison should ignore case, spaces, and punctuation. The example normalises to lowercase and removes non-alphanumeric characters before comparison.

```
String input = "Madam";
String cleaned = input.toLowerCase().replaceAll("[^a-z0-9]", "");
String reversed = new StringBuilder(cleaned).reverse().toString();
System.out.println(cleaned.equals(reversed)); // true
```

48. Why Extent Reports? Syntax for logging

Extent Reports produces readable HTML reports with test status, steps, screenshots, exceptions, and execution details. It is useful for sharing failures with developers and stakeholders. It is a reporting library, not a substitute for proper assertions or logs. Exact setup differs by ExtentReports version; keep one report instance per run and flush it at the end.

```
// Common ExtentReports 5-style pattern (imports/configuration omitted)
ExtentReports extent = new ExtentReports();
ExtentTest test = extent.createTest("Valid login");
test.info("Entering credentials");
test.pass("Home page displayed");
test.fail("Expected heading was not visible");
extent.flush();
```

49. Listeners and their usage

TestNG listeners receive lifecycle callbacks, for example when a test starts, succeeds, fails, or is skipped. Implement ITestListener to centralise logging, screenshots on failure, and reporting. Register listeners with @Listeners or testng.xml. Avoid putting all test logic into listeners; keep them focused on cross-cutting reporting and diagnostics.

```
public class TestListener implements ITestListener {
    @Override
    public void onTestFailure(ITestResult result) {
        System.out.println("FAILED: " + result.getName());
        // Capture screenshot here using the driver for the current test thread.
    }
}
// Register: @Listeners(TestListener.class) on the test class
```

50. StringBuffer vs StringBuilder

Both are mutable character sequences with similar APIs. StringBuffer synchronizes its methods and is thread-safe for individual operations; StringBuilder is not synchronized and is generally faster in single-threaded code. Neither makes a multi-step business operation automatically atomic. Prefer StringBuilder unless shared concurrent access requires synchronisation.

```
StringBuilder sb = new StringBuilder("Hello");
sb.append(" Java");
System.out.println(sb); // Hello Java
```

51. Limitations of Selenium

Selenium automates web browsers, not native desktop applications by itself. It does not provide built-in test reporting, test-data management, or a full test runner; teams combine it with TestNG/JUnit, reporting libraries, and build tools. CAPTCHA and OTP are intentionally difficult to automate reliably and should be handled through approved test environments or service-level test hooks. Selenium can be affected by dynamic UI, browser differences, and asynchronous timing, so stable locators, waits, and good test design are essential.

Good interview nuance: Selenium 4 supports modern browser automation and Grid, but it is not a universal solution for every application type or security control.

52. When to use Robot, Actions, and JavaScriptExecutor

Actions is the preferred Selenium API for complex user interactions such as hover, drag-and-drop, right-click, double-click, and key combinations. JavaScriptExecutor runs JavaScript in the page; use it sparingly for diagnostics or a genuine application need, because bypassing normal user interactions can hide usability defects. Java Robot controls OS-level keyboard/mouse events and may be useful for native dialogs or OS interactions, but it is platform-sensitive and less reliable in headless/remote environments. Prefer DOM-based file upload via sendKeys on input[type=file] rather than Robot when possible.

```
new Actions(driver).moveToElement(menu).perform();
new Actions(driver).doubleClick(button).perform();

JavascriptExecutor js = (JavascriptExecutor) driver;
js.executeScript("arguments[0].scrollIntoView(true);", element);

// Robot is java.awt.Robot; use only when OS-level input is truly required.
```

53. Multiple inheritance in Java

Java does not allow a class to extend two classes because it can create ambiguity (the diamond problem). A class can implement multiple interfaces, and an interface can extend multiple interfaces. Shared implementation can also be achieved with composition or delegation.

```
interface Printable { void print(); }
interface Auditable { void audit(); }
class Report implements Printable, Auditable {
    public void print() {}
    public void audit() {}
}
```

54. Marker interface

A marker interface is an interface with no methods that marks a class as having a property or being eligible for special handling. Classic examples include Serializable and Cloneable. The marker can be checked by frameworks or the JVM/library. Modern Java also has annotations, which often provide metadata more flexibly, but marker interfaces remain relevant in some APIs.

```
class TestRecord implements java.io.Serializable {
    private static final long serialVersionUID = 1L;
}
```

55. XPath following and following-sibling axes

`following-sibling::` selects siblings after the current node with the same parent. `following::` selects nodes later in document order, not only siblings. Use these axes when relationships are stable; prefer a locator tied to a label or container when possible.

```
// Input following a label's sibling:
 //label[normalize-space()='Username']/following-sibling::input
// Any later button after a heading:
 //h2[normalize-space()='Details']/following::button[1]
// The following-sibling axis does not cross to unrelated parent containers.
```

56. this vs super keyword

`this` refers to the current object; it can distinguish instance fields from parameters and call another constructor in the same class using this(...). `super` refers to the immediate parent portion of the object; it can call a parent constructor or parent implementation. A constructor's this(...) or super(...) invocation must be first, and you cannot use both as separate constructor invocations in the same constructor.

```
class Parent { String name = "Parent"; }
class Child extends Parent {
    String name = "Child";
    void show() {
        System.out.println(this.name);  // Child
        System.out.println(super.name); // Parent
    }
}
```

57. Multithreading

Multithreading allows multiple threads to execute concurrently. It can improve throughput for independent work but introduces shared-state risks such as race conditions and visibility issues. Java supports Thread, Runnable, Callable, ExecutorService, and concurrency utilities. In Selenium, parallel tests need isolated WebDriver instances (often ThreadLocal in a carefully designed framework), isolated test data, and thread-safe reporting; never share one driver across parallel tests.

```
ExecutorService pool = Executors.newFixedThreadPool(2);
try {
    pool.submit(() -> System.out.println("Task 1"));
    pool.submit(() -> System.out.println("Task 2"));
} finally {
    pool.shutdown();
}
```

58. Ways to compare two Strings

Use equals() for case-sensitive content comparison and equalsIgnoreCase() for case-insensitive comparison. `==` compares object references, not text content. compareTo() provides lexicographic ordering. For null-safe comparison, use Objects.equals(a, b). Normalise whitespace/case first only when the business requirement says those differences should be ignored.

```
String a = new String("Java");
String b = new String("Java");
System.out.println(a == b);             // false
System.out.println(a.equals(b));        // true
System.out.println(a.equalsIgnoreCase("JAVA")); // true
System.out.println(Objects.equals(a, b)); // true
```

59. Window handling in Selenium

Each browser tab/window has a handle. Store the parent handle before clicking a link that opens a new window, wait until the handle count increases, switch to the new handle, perform checks, and switch back when needed. Do not rely on handle order being meaningful; identify the target by title/URL where possible.

```
String parent = driver.getWindowHandle();
driver.findElement(By.id("openWindow")).click();
new WebDriverWait(driver, Duration.ofSeconds(10))
    .until(d -> d.getWindowHandles().size() > 1);
for (String handle : driver.getWindowHandles()) {
    if (!handle.equals(parent)) {
        driver.switchTo().window(handle);
        System.out.println(driver.getTitle());
        driver.close();
    }
}
driver.switchTo().window(parent);
```

60. static and final

`static` belongs to the class rather than a particular object; static methods are called on the class and cannot directly access instance members without an object. `final` means a variable can be assigned once, a method cannot be overridden, or a class cannot be subclassed. A final reference cannot point to another object, but the referenced object's mutable state may still change. A static final field is commonly used for constants.

```
class Config {
    static int browserCount = 0;
    static final String BASE_URL = "https://example.com";
    final String name;
    Config(String name) { this.name = name; }
}
```

61. Correct syntax for a static class declaration

`public class StaticClass()` is invalid Java syntax: class declarations do not have parentheses. A top-level class cannot be declared static. A nested class may be static. Use `public class StaticClass { ... }` for a top-level class, and `static class Nested { ... }` inside another class.

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

class Outer {
    static class Nested { }
}
```

62. How are static variables stored in memory?

Conceptually, static fields belong to the class rather than each object and have one class-level value per loaded class. In modern JVM implementations, class metadata is managed in Metaspace, while static field values are associated with the class runtime data; exact implementation details vary by JVM version. Do not say static variables are stored on each object's heap instance: instance fields belong to individual objects. Static fields remain available while the defining class is loaded, subject to class-loader lifecycle.

> Interview tip: distinguish the conceptual Java memory model from implementation-specific JVM storage details.

63. Convert "abcde" to "abc*de"

Insert the asterisk at the required position. If the requirement is always after the third character, validate that the input is long enough and use substring.

```
String input = "abcde";
String output = input.length() > 3
    ? input.substring(0, 3) + "*" + input.substring(3)
    : input;
System.out.println(output); // abc*de
```

64. Handle frames in Selenium

A frame/iframe contains a separate browsing context. Selenium must switch into it before locating its elements, and switch back to defaultContent() afterward. You can switch by index, name/id, or WebElement; the WebElement approach is often clearest when there are multiple frames. Wait for frame availability rather than using a fixed sleep.

```
WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(10));
wait.until(ExpectedConditions.frameToBeAvailableAndSwitchToIt(By.id("paymentFrame")));
driver.findElement(By.id("cardNumber")).sendKeys("4111111111111111");
driver.switchTo().defaultContent();
```

65. Drivers available in Selenium

Selenium WebDriver supports browser-specific driver classes, commonly ChromeDriver, FirefoxDriver, EdgeDriver, SafariDriver, and InternetExplorerDriver in legacy environments (Internet Explorer is retired and support is deprecated/limited). RemoteWebDriver connects to a remote browser/Grid or cloud provider. Selenium Manager can help discover/download compatible drivers in supported Selenium versions. Availability depends on the operating system, browser version, and Selenium version.

```
WebDriver chrome = new ChromeDriver();
WebDriver firefox = new FirefoxDriver();
WebDriver edge = new EdgeDriver();
// SafariDriver requires supported macOS/Safari setup.
// RemoteWebDriver is configured with a remote URL and capabilities.
```

Quick revision checklist

- Be ready to explain interface vs abstract class, overloading vs overriding, and String immutability.
- Practise coding: duplicate-character removal, palindrome, String insertion, collections, and POM login.
- Revise Selenium interactions: waits, links, tables, frames, windows, Actions, JavaScriptExecutor, and Robot.
- Explain your actual Cucumber + POM hybrid framework from feature file through reporting.
- Prepare two genuine automation challenges and describe the root cause, fix, and result.
- Keep HR answers positive, truthful, concise, and aligned with your latest resume.