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

56 interview topics with explanations, Java/Selenium examples, and sample answers

Note: Some items in the supplied list refer to an unspecified table, pattern, or output. Those questions require the interviewer's actual input to produce the exact answer; this guide explains the approach and gives representative examples. Personalize project-specific claims to your real experience.

1. Java constructor

A constructor initializes a new object. It has the same name as the class and no return type. Constructors can be overloaded. If you declare no constructor, Java normally supplies a default no-argument constructor.

```
class Employee {
    String name;
    int id;

    Employee(String name, int id) {
        this.name = name;
        this.id = id;
    }
}
Employee e = new Employee("Ravi", 101);
```

2. Method overriding

Method overriding occurs when a subclass provides a new implementation of an inherited instance method with a compatible signature. Runtime polymorphism selects the implementation based on the actual object type. Static methods are hidden, not overridden.

```
class Parent {
    void show() { System.out.println("Parent"); }
}
class Child extends Parent {
    @Override
    void show() { System.out.println("Child"); }
}
Parent p = new Child();
p.show(); // Child
```

3. Why is a constructor used?

A constructor initializes object state and required dependencies at creation time. In a Selenium Page Object, a constructor commonly receives the shared WebDriver instance.

```
class LoginPage {
    private final WebDriver driver;
    LoginPage(WebDriver driver) {
        this.driver = driver;
    }
}
```

4. Interface vs abstract class

- Abstract class: can have constructors, instance fields, abstract methods, and implemented methods; a class can extend only one class.
- Interface: defines a contract; can contain abstract, default, static, and private methods in modern Java; a class can implement multiple interfaces.

Use an abstract class for shared state/implementation among closely related classes; use an interface for a capability or contract shared by potentially unrelated classes.

5. Static methods

A static method belongs to the class, not to an individual object. Call it using the class name. A static method cannot directly access instance fields or use this.

```
class Calculator {
    static int add(int a, int b) { return a + b; }
}
int result = Calculator.add(2, 3);
```

6. How are failed test cases handled in Selenium?

Selenium itself automates browser interactions; the test framework (e.g., TestNG) marks tests passed or failed based on exceptions and assertions. On failure, capture useful diagnostics, record the stack trace, take a screenshot, publish the report, and clean up the browser. Avoid catching and suppressing failures.

```
@Test
public void loginTest() {
    try {
        // steps and assertions
        Assert.assertTrue(driver.getTitle().contains("Home"));
    } catch (AssertionError | RuntimeException e) {
        // capture screenshot/logs here
        throw e; // preserve the failure
    }
}
```

A TestNG listener such as ITestListener is often a better central place for screenshots and reporting. Use @AfterMethod(alwaysRun = true) to ensure teardown executes.

7. TestNG attributes

Common @Test attributes include priority, groups, dependsOnMethods, dependsOnGroups, invocationCount, threadPoolSize, enabled, timeOut, expectedExceptions, dataProvider, and dataProviderClass. Use only the attributes that support a real test requirement.

```
@Test(groups = {"smoke"}, priority = 1)
public void login() { }

@Test(dependsOnMethods = "login")
public void profile() { }

@Test(invocationCount = 3)
public void repeatCheck() { }
```

8. Selenium screenshot code

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

Imports include java.io.File, java.nio.file.Files, Paths, StandardCopyOption, and Selenium screenshot classes. Use unique filenames per test; screenshots are especially useful on failure.

9. Handling dropdowns and selecting values

For a native HTML select element, use Selenium's Select class. Select by visible text, value attribute, or zero-based index. For custom dropdowns built with div/button elements, click the dropdown and then locate/click the desired option; Select does not work on those.

```
WebElement element = driver.findElement(By.id("country"));
Select select = new Select(element);
select.selectByVisibleText("India");
// alternatives:
select.selectByValue("IN");
select.selectByIndex(2);
```

Prefer visible text or a stable value over index because option order can change.

10. Get an element from a web table

Locate the table, find the row using a unique business value, then find the target cell/action inside that row. This is safer than using a hard-coded row index.

```
WebElement table = driver.findElement(By.id("employees"));
List<WebElement> rows = table.findElements(By.cssSelector("tbody tr"));
for (WebElement row : rows) {
    List<WebElement> cells = row.findElements(By.cssSelector("td"));
    if (!cells.isEmpty() && cells.get(0).getText().trim().equals("Ravi")) {
        System.out.println("Salary: " + cells.get(3).getText());
        row.findElement(By.linkText("Edit")).click();
        break;
    }
}
```

11. Interviewer provided a table

The exact answer depends on the supplied table and requested operation (for example, find a value, count rows/columns, select a row, or click Edit). Clarify the target column/value, then use row-scoped locators. Example pattern:

```
WebElement row = driver.findElement(
    By.xpath("//table[@id='employees']//tr[td[normalize-space()='Ravi']]"));
String email = row.findElement(By.xpath("./td[3]")).getText();
```

12. Rate yourself in Java out of 10

Give a defensible rating and explain what you can do at that level. Example: “I would rate myself 7.5 or 8 out of 10 for Java automation development. I’m comfortable with OOP, strings, collections, exception handling, file handling, and writing reusable framework utilities. I continue to improve in concurrency, JVM internals, and advanced design patterns.” Choose the rating that matches your skills and be ready for follow-up coding.

13. What is inheritance?

Inheritance lets a subclass reuse and specialize accessible behavior from a superclass using extends. Java classes support single inheritance; a class can implement multiple interfaces.

```
class BasePage {
    void waitForPage() { }
}
class LoginPage extends BasePage {
    void login() { waitForPage(); }
}
```

14. Collections and their advantages

The Java Collections Framework provides reusable data structures and algorithms.

- List: ordered, allows duplicates; e.g., List<WebElement> for rows/options.
- Set: unique elements; e.g., Set<String> for browser window handles.
- Map: key-value pairs; e.g., Map<String,String> for configuration/test data.
- Queue/Deque: useful for ordered processing.

Benefits include reuse, standard APIs, type safety with generics, easier data manipulation, and a choice of implementation based on ordering, uniqueness, and performance needs.

15. What is a String?

String is a Java class representing a sequence of characters. String objects are immutable: operations such as replace or concat return a new String rather than changing the original.

```
String name = "Selenium";
name.concat(" WebDriver");
System.out.println(name); // Selenium
name = name.concat(" WebDriver");
System.out.println(name); // Selenium WebDriver
```

16. What is an abstract class?

An abstract class is declared with abstract and cannot be instantiated directly. It can contain abstract methods that subclasses must implement (unless they remain abstract) and concrete methods that subclasses inherit.

17. Why choose TestNG instead of JUnit?

Both are capable Java test frameworks. TestNG is often chosen in Selenium projects for built-in groups, flexible suite XML, dependencies, DataProvider, and parallel-execution configuration. JUnit (especially JUnit 5) is also strong and has an extensive extension ecosystem. Explain why TestNG fits your actual project rather than claiming JUnit cannot do something.

18. IDE vs Selenium WebDriver

An IDE (such as IntelliJ IDEA, Eclipse, or VS Code) is software used to write, navigate, debug, and run code. Selenium WebDriver is an automation API/library that drives browsers. They serve different purposes and are used together.

19. TestNG annotations

- @BeforeSuite / @AfterSuite: once around a suite.
- @BeforeTest / @AfterTest: around a TestNG XML <test> block.
- @BeforeClass / @AfterClass: around test methods in a class.
- @BeforeMethod / @AfterMethod: before/after each @Test method.
- @BeforeGroups / @AfterGroups: around specified groups.
- @Test: defines a test method.
- @DataProvider: supplies data sets; @Parameters reads XML parameters; @Listeners registers listeners.

20. Java keywords

Keywords are reserved words with defined meaning in Java and cannot be used as ordinary identifiers. Examples include class, interface, extends, implements, public, private, protected, static, final, abstract, void, return, if, else, switch, case, for, while, do, break, continue, try, catch, finally, throw, throws, new, this, super, synchronized, volatile, transient, native, package, import, instanceof, enum, assert, and strictfp. true, false, and null are literals, not keywords.

21. Why is TestNG needed in your project?

TestNG organizes and executes tests; supplies lifecycle hooks, assertions, groups, dependencies, parameterization, DataProvider, parallel execution, and listener/reporting integration. It is not required by Selenium itself, but it helps manage larger suites.

22. Have you used data-driven testing from Excel?

A data-driven approach runs the same test logic with different data sets. Excel data can be read with Apache POI and supplied to TestNG through @DataProvider. Keep the test logic separate from input values and handle blank/missing cells.

```
@DataProvider(name = "loginData")
public Object[][] loginData() {
    return new Object[][] {
        {"user1", "pass1"},
        {"user2", "pass2"}
    };
}
@Test(dataProvider = "loginData")
public void loginTest(String user, String pass) {
    // use values in the page object
}
```

23. Replace vowels with numbers: a=1, e=2, i=3, o=4, u=5

Assumption: replace lowercase and uppercase vowels, preserving uppercase where possible is an optional variation. The following example replaces lowercase vowels and is easy to extend.

```
public class ReplaceVowels {
    public static void main(String[] args) {
        String input = "automation";
        StringBuilder output = new StringBuilder();

        for (char ch : input.toCharArray()) {
            switch (ch) {
                case 'a': output.append('1'); break;
                case 'e': output.append('2'); break;
                case 'i': output.append('3'); break;
                case 'o': output.append('4'); break;
                case 'u': output.append('5'); break;
                default: output.append(ch);
            }
        }
        System.out.println(output); // 15t4m1t34n
    }
}
```

24. Explain Page Object Model

POM is a design pattern in which each page/component has a class encapsulating its locators and operations. Tests call business-level methods instead of repeating low-level locators. This improves reuse and maintainability.

```
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("login");

    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();
    }
}
```

25. How proficient are you in Java?

Similar to Question 12. State your level, then support it with OOP, collections, exception handling, strings, file handling, test utilities, and examples of code you maintain. Avoid a rating you cannot demonstrate.

26. Program to print a pattern

The exact pattern was not included in the question list, so an exact solution cannot be determined. For example, a right triangle of stars can be printed with nested loops:

```
for (int i = 1; i <= 5; i++) {
    for (int j = 1; j <= i; j++) {
        System.out.print("* ");
    }
    System.out.println();
}
```

Output: one star in row 1, two in row 2, continuing to five. Share the actual pattern to get the exact program.

27. What is the output of the following?

The code snippet was not included, so the output cannot be determined. In an interview, trace variable values line by line, evaluate operator precedence, identify static vs instance dispatch, and consider loop boundaries and exception flow. Provide the actual snippet for an exact output.

28. OOP concepts

- Encapsulation: hide internal state and expose controlled methods.
- Abstraction: expose essential behavior while hiding implementation details.
- Inheritance: reuse/extend behavior from a superclass.
- Polymorphism: same interface/base type can refer to different implementations.

In Selenium, WebDriver is an interface implemented by browser-specific drivers; page objects encapsulate locators and actions.

29. Can Java achieve operator overloading?

Java does not support user-defined operator overloading for classes. The + operator is built in for numeric addition and String concatenation, but developers cannot redefine + for a custom class.

30. Types of inheritance

Common conceptual types: single, multilevel, hierarchical, multiple, and hybrid. Java classes support single, multilevel, and hierarchical inheritance through classes. Multiple/hybrid inheritance of type can be modeled using interfaces; a class cannot extend two classes.

```
class A { }
class B extends A { }       // single
class C extends B { }       // multilevel
class D extends A { }       // hierarchical
```

31. What is TestNG and what are its annotations?

TestNG is a Java test framework used to define, organize, parameterize, and execute tests. Review the annotations in Question 19 and the attributes in Question 7.

32. Parallel execution

Configure TestNG with parallel and thread-count in testng.xml or use invocationCount/threadPoolSize for method invocations. Isolate WebDriver per test/thread (commonly ThreadLocal in a carefully designed framework) and isolate test data to avoid race conditions.

```
<suite name="Suite" parallel="methods" thread-count="3">
  <test name="UI">
    <classes><class name="tests.LoginTest"/></classes>
  </test>
</suite>
```

33. What is TestRunner?

The term depends on the framework. In Cucumber, a TestRunner is commonly a class configured with @CucumberOptions and a JUnit/TestNG runner to locate feature files, glue code, tags, and plugins. In TestNG-only projects, testng.xml and the TestNG engine select and run tests; there may not be a class literally named TestRunner.

```
@RunWith(Cucumber.class) // JUnit-based example
@CucumberOptions(
    features = "src/test/resources/features",
    glue = "stepdefinitions",
    plugin = {"pretty", "html:target/cucumber.html"}
)
public class TestRunner { }
```

This snippet is the JUnit 4 Cucumber runner style; TestNG-based Cucumber runners use a different setup.

34. Dynamic XPath

Dynamic XPath handles changing attributes by using stable parts, text, or relationships. Validate uniqueness.

```
// ID begins with a stable prefix
//input[starts-with(@id,'user_')]

// Attribute contains a stable substring
//button[contains(@class,'submit')]

// Match by visible text
//button[normalize-space()='Save']

// Locate Edit inside the row containing a name
//tr[td[normalize-space()='Ravi']]//a[normalize-space()='Edit']
```

35. Randomly generated ID: how do you locate the element?

Prefer stable attributes such as name, aria-label, data-testid, visible text, or a stable parent/child relationship. If the ID has a stable prefix/suffix, use starts-with() or contains(). Avoid relying on a random ID or position alone.

```
By field = By.xpath("//input[starts-with(@id,'customer_')]");
WebElement input = new WebDriverWait(driver, Duration.ofSeconds(10))
    .until(ExpectedConditions.visibilityOfElementLocated(field));
```

36. Reverse a string

```
String input = "Selenium";
String reversed = new StringBuilder(input).reverse().toString();
System.out.println(reversed); // muineleS
```

If the interviewer asks to reverse words rather than characters, clarify the requirement and use split/join or a word-level algorithm.

37. Is String mutable or immutable? Why?

String is immutable: once created, its character sequence cannot change. Operations such as replace/concat return a new String. Immutability supports safe sharing, predictable hashing, and String pool reuse. Use StringBuilder for repeated modifications in a single thread.

```
String s = "Java";
s.concat(" Selenium");
System.out.println(s); // Java
s = s.concat(" Selenium");
System.out.println(s); // Java Selenium
```

38. Collection

See Question 14. The key interfaces are List, Set, Queue/Deque, and Map (Map is part of the collections framework but does not extend Collection).

39. Common Selenium exceptions

- NoSuchElementException: locator did not find a matching element.
- StaleElementReferenceException: DOM changed after the element was located.
- TimeoutException: wait condition did not become true in time.
- ElementClickInterceptedException: another element/overlay intercepted the click.
- ElementNotInteractableException: element is present but not interactable.
- InvalidSelectorException: malformed CSS/XPath.
- NoSuchFrameException: requested frame not found.
- SessionNotCreatedException: browser/driver/session setup issue.

Diagnose the underlying cause; use explicit waits, stable locators, frame/window switching, and compatible browser configuration rather than indiscriminately retrying.

40. Technical Round 2

This appears to be a section heading rather than a question. Use it to organize the following framework and coding topics during preparation.

41. Explain wrapper methods in your framework

Wrapper methods are reusable functions that wrap common Selenium operations with consistent waits, logging, and error handling. Examples might include clickElement(), enterText(), selectByVisibleText(), waitForVisible(), and captureScreenshot(). Avoid creating wrappers that hide useful failure details.

```
public void clickWhenReady(By locator) {
    WebElement element = new WebDriverWait(driver, Duration.ofSeconds(10))
        .until(ExpectedConditions.elementToBeClickable(locator));
    element.click();
}
```

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

Separate test logic from test data. Store data in Excel/CSV/JSON/database or TestNG DataProvider; feed each data row to the same test method and validate each result. Use unique test data where tests modify application state and avoid embedding secrets in files.

43. Read data from Excel

Apache POI can read .xls and .xlsx. WorkbookFactory.create() detects supported workbook formats. Use DataFormatter to obtain displayed cell values and try-with-resources to close resources.

```
try (FileInputStream in = new FileInputStream("testdata.xlsx");
     Workbook workbook = WorkbookFactory.create(in)) {
    Sheet sheet = workbook.getSheet("Login");
    DataFormatter formatter = new DataFormatter();
    String username = formatter.formatCellValue(sheet.getRow(1).getCell(0));
    String password = formatter.formatCellValue(sheet.getRow(1).getCell(1));
    System.out.println(username);
}
```

44. Arrays vs ArrayList

- Array has fixed length after creation and can store primitives or references.
- ArrayList is a resizable List of objects (primitives use wrapper types), supports methods such as add/remove, and allows duplicates.

```
int[] values = {1, 2, 3};
List<Integer> list = new ArrayList<>();
list.add(1);
list.add(2);
list.add(2); // duplicate allowed
```

45. How do you handle exceptions?

Catch an exception when you can recover, add context, or translate it into a meaningful failure. Do not swallow it. Use specific exception types, preserve the original cause, and use finally/try-with-resources for cleanup.

```
try {
    driver.findElement(By.id("save")).click();
} catch (NoSuchElementException e) {
    throw new RuntimeException("Save button was not found", e);
} finally {
    // cleanup if appropriate
}
```

46. Print numbers greater than 10 using a for loop

```
int[] numbers = {4, 11, 7, 18, 10, 25};
for (int number : numbers) {
    if (number > 10) {
        System.out.println(number);
    }
}
```

47. How does a for loop work?

A traditional for loop has initialization, a condition, and an update. Initialization runs once; the condition is checked before each iteration; the body runs if true; then the update runs.

```
for (int i = 1; i <= 5; i++) {
    System.out.println(i);
}
```

48. Handle windows and alerts

Windows/tabs use getWindowHandle(s) and switchTo().window(handle); JavaScript alerts use switchTo().alert(). Do not mix these APIs.

```
// New window
String original = driver.getWindowHandle();
driver.findElement(By.id("open")).click();
new WebDriverWait(driver, Duration.ofSeconds(10))
    .until(d -> d.getWindowHandles().size() > 1);
for (String handle : driver.getWindowHandles()) {
    if (!handle.equals(original)) {
        driver.switchTo().window(handle);
        break;
    }
}

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

49. Code for window handles

Save the original handle, wait for a new handle, compare the set, switch to the new context, and return to the original when finished. A browser tab is treated as a separate WebDriver window handle.

50. Why is String immutable?

See Question 37. Immutability improves safe sharing, stable hash codes, predictable behavior, and string-pool reuse. Use StringBuilder for repeated string modifications.

51. Static keyword

static can be used for fields, methods, initialization blocks, and nested classes. A static field has one class-level copy shared by instances; a static method can be called through the class name and cannot directly access instance members without an object.

52. Constructor and its use

See Questions 1 and 3. Constructors initialize objects and dependencies; in a page object, passing WebDriver through the constructor helps every page use the same session.

53. XPath syntax

```
// By ID-like attribute
//input[@id='username']

// By multiple attributes
//input[@type='text' and @name='email']

// Text
//button[normalize-space()='Submit']

// Parent/child relationship
//div[@class='form']//input[@name='email']

// Following sibling
//label[normalize-space()='Email']/following-sibling::input
```

Prefer relative XPath anchored on stable attributes; check that it matches the intended element exactly.

54. Deadlock in Java

A deadlock occurs when threads wait indefinitely for locks held by each other. For example, thread 1 holds lock A and waits for B, while thread 2 holds B and waits for A. Prevent it by acquiring locks in a consistent order, reducing nested locking, using timed locks where suitable, and avoiding unnecessary shared mutable state.

```
// Conceptual prevention: always acquire lockA before lockB
synchronized (lockA) {
    synchronized (lockB) {
        // critical section
    }
}
```

55. What is Map?

Map stores key-value pairs; keys are unique, values may repeat. HashMap has no guaranteed order, LinkedHashMap preserves insertion order, and TreeMap sorts keys.

```
Map<String, String> roles = new HashMap<>();
roles.put("ravi", "Tester");
roles.put("ravi", "Lead"); // replaces prior value
System.out.println(roles.get("ravi")); // Lead
```

56. Selenium Grid

Selenium Grid runs WebDriver tests on remote browser instances and supports parallel execution across browsers, versions, and operating systems. A test sends WebDriver commands to Grid, which routes them to a suitable node/browser session. Grid 4 supports standalone, hub/node, and distributed configurations. A single machine can run Grid in standalone mode; browser availability and machine resources still limit concurrency.

```
// Example remote WebDriver (Grid URL depends on your environment)
ChromeOptions options = new ChromeOptions();
WebDriver driver = new RemoteWebDriver(
    URI.create("http://localhost:4444").toURL(), options);
```

In interviews, describe how your team provisions nodes, selects browser capabilities, collects reports, and isolates test data if you have actually used Grid.

Quick revision checklist

- Practise constructors, overriding, interfaces/abstract classes, static, final, and String immutability.
- Be ready to write screenshot, dropdown, table, frame, alert, and window-handling code.
- Know TestNG annotations, attributes, DataProvider, listeners, and parallel execution.
- Review collections, arrays vs ArrayList, Map, exceptions, and basic Java programs.
- Explain your real framework architecture, wrapper methods, and data-driven approach.
- For questions missing a pattern, code snippet, or table, ask the interviewer to provide the exact input.