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

18 questions with detailed explanations, practical examples, and sample interview answers

Note: Sample answers are templates. Adjust the project, tools, achievements, browser setup, and skill ratings to reflect your actual experience. Code examples assume Java, Selenium 4, TestNG, and Maven unless otherwise stated.

1. Tell me about yourself

Use a 60–90 second answer: experience, testing skills, automation stack, responsibilities, and the role you want. Example tailored to a senior QA automation profile:

“I have around 13 years of experience in software testing, covering manual and automation testing. My automation skills include Selenium WebDriver with Java, TestNG, Cucumber, API automation, and Playwright. I have worked on designing and maintaining reusable automation frameworks, preparing test scenarios, executing regression suites, investigating failures, and coordinating with developers and business stakeholders to resolve defects. I also contribute to test planning, execution tracking, and quality reporting. I’m looking for a role where I can combine hands-on automation with broader quality ownership.”

Personalize this answer with your real project domain, framework, team responsibilities, and one measurable achievement. Avoid listing tools you cannot explain in depth.

2. How do you rate yourself in Java?

Give a realistic rating and explain what it means. For a senior automation engineer, a sample answer could be: “I would rate myself 7.5 or 8 out of 10 in Java for automation-related development. I’m comfortable with OOP, collections, exception handling, strings, interfaces, constructors, file handling, and writing reusable utilities. I can debug framework code and work with TestNG and Maven. I continue improving in advanced topics such as concurrency, JVM internals, and design patterns.”

Choose your own rating. Interviewers often follow up with coding questions, so be prepared to demonstrate the level you claim. A specific and honest rating is stronger than claiming 10/10.

3. Difference between abstract class and interface

- Abstract class: can have instance fields, constructors, abstract methods, and concrete methods. A class can extend only one class.
- Interface: defines a contract; it can contain abstract methods, constants, and (in modern Java) default, static, and private methods. A class can implement multiple interfaces.
- Use an abstract class when related classes share state or common implementation. Use an interface when you want to define a capability that different classes can implement.

```
abstract class BasePage {
    protected WebDriver driver;

    BasePage(WebDriver driver) {
        this.driver = driver;
```

```
    }

    abstract boolean isLoaded();

    void open(String url) {
        driver.get(url);
    }
}

interface ScreenshotCapable {
    void takeScreenshot();
}

class LoginPage extends BasePage implements ScreenshotCapable {
    LoginPage(WebDriver driver) { super(driver); }

    @Override
    boolean isLoaded() {
        return driver.getTitle().contains("Login");
    }

    @Override
    public void takeScreenshot() {
        // screenshot implementation
    }
}
```

4. What does this refer to in Java?

this refers to the current object instance. It is commonly used to distinguish instance fields from parameters, call another constructor in the same class using this(...), or pass the current object to another method.

```
class Employee {
    private String name;

    Employee(String name) {
        this.name = name; // field = parameter
    }

    Employee() {
        this("Unknown"); // calls another constructor
    }
}
```

A static method cannot use this because there is no current instance associated with a static call.

5. Can a static method be overridden?

No. Static methods are hidden, not overridden. Method overriding applies to instance methods and is resolved at runtime according to the actual object type. Static method calls are resolved using the reference/class type.

```
class Parent {
    static void show() { System.out.println("Parent"); }
    void display() { System.out.println("Parent instance"); }
}
class Child extends Parent {
    static void show() { System.out.println("Child"); } // hiding
    @Override
    void display() { System.out.println("Child instance"); } // overriding
}
Parent obj = new Child();
obj.show();    // Parent
obj.display(); // Child instance
```

6. Can we call a static method without creating an object?

Yes. Call it using the class name, which is the clearest approach. It can also be called from a static context in the same class without an object.

```
class MathUtil {
    static int add(int a, int b) {
        return a + b;
    }
}
int result = MathUtil.add(2, 3); // 5
```

Although Java allows some static methods to be called through an object reference, avoid that style because it can make the code misleading.

7. What is a finally block?

A finally block is used with try/catch to run cleanup code whether the try block succeeds or an exception is caught. It is commonly used to release resources, though try-with-resources is preferred for AutoCloseable resources. It may not execute if the JVM terminates abruptly.

```
try {
    System.out.println("Run test");
} catch (RuntimeException e) {
    System.out.println("Handle error: " + e.getMessage());
} finally {
    System.out.println("Cleanup");
}
```

8. What is final?

final is a Java modifier with different meanings depending on where it is used:

- final variable: can be assigned only once.
- final method: cannot be overridden by a subclass.
- final class: cannot be subclassed.

```
final int timeout = 10;

class Base {
    final void validate() { }
}
final class Utility {
    // cannot be extended
}
```

A final reference cannot be reassigned, but the object it references may still be mutable. For example, a final List can still have elements added unless the list itself is made unmodifiable.

9. What issues have you faced during automation?

Answer with a real example using Situation → Problem → Investigation → Fix → Result. Common examples include flaky tests caused by asynchronous loading, stale elements after DOM updates, intercepted clicks caused by overlays, unstable test data, browser/driver incompatibility, file upload paths, and parallel-test collisions.

Sample answer: “One challenge was a click failing intermittently because the page was still loading and a loader overlay was covering the button. I reproduced the failure, checked the browser logs and DOM state, replaced a fixed sleep with an explicit wait for the loader to disappear and the button to become clickable, and added clearer failure logging. This made the test more stable and easier to diagnose.”

Use this only if it matches a real situation you have handled; otherwise substitute your actual example.

10. How do you switch between frames?

Use driver.switchTo().frame() by frame name/ID, index, or a WebElement. After interacting inside the frame, return to the top document with defaultContent(), or use parentFrame() to move up one level.

```
// By name or id
driver.switchTo().frame("paymentFrame");

// Or by element
WebElement frame = driver.findElement(By.cssSelector("iframe.payment"));
driver.switchTo().frame(frame);

driver.findElement(By.id("cardNumber")).sendKeys("1234");
driver.switchTo().defaultContent();
```

If the frame loads asynchronously, wait for it to be available. Elements inside a frame cannot be located until WebDriver has switched into that frame.

11. Can the same test cases run in multiple browsers?

Yes. Use a browser configuration property or TestNG parameter to create the appropriate WebDriver, then run the same test logic against Chrome, Firefox, and Edge. Keep browser-specific setup isolated and avoid browser-dependent assumptions in tests.

```
@Parameters("browser")
@BeforeMethod
public void setup(String browser) {
    switch (browser.toLowerCase()) {
        case "chrome": driver = new ChromeDriver(); break;
        case "firefox": driver = new FirefoxDriver(); break;
        case "edge": driver = new EdgeDriver(); break;
        default: throw new IllegalArgumentException("Unsupported: " + browser);
    }
}
```

You can run separate TestNG XML entries for different browsers or use Selenium Grid for remote browsers. Verify that the application supports the target browsers and versions.

12. How do you achieve concurrent test-case execution?

TestNG supports parallel execution at suite, test, class, or method level through testng.xml and annotation options. Selenium tests must use isolated WebDriver instances and independent test data to avoid one test changing another test's browser or application state.

```
<suite name="Regression" parallel="methods" thread-count="4">
  <test name="UI Tests">
    <classes>
      <class name="tests.LoginTest"/>
      <class name="tests.ProfileTest"/>
    </classes>
  </test>
</suite>
```

For invocation-level concurrency, TestNG also supports invocationCount and threadPoolSize on @Test. Start with a small thread count and check machine resources, environment capacity, test-data isolation, and report/listener thread safety.

13. What methods do you follow to optimize code?

- Use Page Object Model and reusable utilities to reduce duplication.
- Prefer stable locators and explicit waits instead of repeated fixed sleeps.
- Keep test data and environment configuration outside test logic.
- Use meaningful method/class names and small single-purpose methods.
- Use appropriate collections and avoid unnecessary browser calls.
- Capture screenshots/logs on failure and remove noisy debugging output.
- Run tests in parallel only when browser sessions and test data are isolated.
- Review code, refactor duplicated logic, and track flaky-test rates and execution time.

Optimization should improve readability, reliability, and runtime without making tests difficult to maintain. Do not optimize only for fewer lines of code.

14. How do you structure POM (Page Object Model)?

POM gives each page or component a class containing its locators and user-facing actions. Test classes express business scenarios and assertions, while page objects encapsulate page interaction. Common setup, waits, configuration, test data, and reporting can live in separate utilities.

```
// 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 loginButton = 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(loginButton).click();
    }
}
```

```
// LoginTest.java
@Test
public void validLogin() {
    LoginPage page = new LoginPage(driver);
    page.login("testuser", "password");
    Assert.assertTrue(driver.getTitle().contains("Home"));
}
```

Avoid placing every assertion or all test flow logic inside a page object. Keep the division consistent with the framework conventions.

15. Why do you use Maven projects?

Maven manages dependencies, compiles code, runs tests, and standardizes builds through pom.xml. It makes the project easier to run on another developer's machine or CI agent when the required Java and environment settings are available.

```
mvn clean       # removes prior build output
mvn compile     # compiles main sources
mvn test        # runs configured tests
mvn clean test  # clean build and tests
```

Maven also supports plugins, profiles for environments, dependency version control, and integration with TestNG/Jenkins.

16. Other build mechanisms you have heard of

Gradle is another widely used build automation tool, using Groovy or Kotlin DSLs. Ant is an older Java build tool that uses XML build files. npm scripts are common in JavaScript projects. Choose the tool based on ecosystem, dependency management, build complexity, and team conventions.

17. Can Jenkins automate build time?

Yes. Jenkins can trigger a build on a schedule, a source-control webhook, a code change, or a manual request. Scheduled builds use cron-like syntax in Jenkins. For example, a nightly regression job can check out code, install/build dependencies, run tests, publish reports, and notify the team.

```
# Example Jenkins cron schedule: every day at 1:30 AM
30 1 * * *
```

Jenkins uses a scheduling syntax with a hash option (H) to help distribute jobs, for example H 1 * * *. The job must have the correct timezone, agent availability, credentials, and environment configured. Do not assume the Jenkins server's timezone matches your local timezone.

18. How good are you in SQL?

Give a truthful level and back it with examples. Sample answer: "I'm comfortable with SQL used in testing, including SELECT, WHERE, ORDER BY, GROUP BY, joins, aggregate functions, subqueries, and validating data against application behavior. I can write queries to check test data, investigate mismatches, and verify backend records. I use transactions and update/delete statements carefully in non-production test environments."

```
-- Verify a record
SELECT id, status
FROM orders
WHERE order_id = 1001;

-- Count records by status
SELECT status, COUNT(*) AS total
FROM orders
GROUP BY status;

-- Join related tables
SELECT o.order_id, c.customer_name
FROM orders o
JOIN customers c ON c.customer_id = o.customer_id
WHERE o.order_id = 1001;
```

If the role expects advanced SQL, revise joins, NULL behavior, grouping, window functions, indexes, transactions, and query plans. Never run destructive SQL against production data without explicit authorization and safeguards.

Final interview preparation checklist

- Use an honest Java rating and be prepared to code at that level.
- Explain one real automation issue from diagnosis through resolution.
- Draw your framework architecture and explain POM responsibilities.
- Know how your project selects browsers and executes parallel tests.
- Explain Maven and Jenkins with the commands and schedules your team actually uses.
- Prepare SQL examples you can write and explain without assistance.