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

9 interview topics — detailed answers, examples, and sample responses

Use these questions and career-response examples as preparation guides. Tailor every answer to your actual projects, responsibilities, tools, and experience.

1. Fibonacci series

The Fibonacci series is a sequence where each number after the first two is the sum of the previous two. A common starting sequence is 0, 1, 1, 2, 3, 5, 8, 13. In interviews, clarify whether the interviewer wants a fixed number of terms or all terms below a limit. An iterative solution is efficient and avoids the repeated work of simple recursive implementations.

Example

```
public class Fibonacci {
    public static void main(String[] args) {
        int n = 10;
        int first = 0, second = 1;

        for (int i = 0; i < n; i++) {
            System.out.print(first + " ");
            int next = first + second;
            first = second;
            second = next;
        }
    }
}
// Output: 0 1 1 2 3 5 8 13 21 34
```

Interview tip: The iterative solution takes O(n) time and O(1) extra space, excluding output storage.

2. Method overriding

Method overriding occurs when a child class provides its own implementation of an inherited instance method with the same signature and a compatible return type. It enables runtime polymorphism: the implementation executed depends on the actual object type. Use @Override so the compiler can catch signature mistakes. Static methods are hidden rather than overridden, private methods are not overridden, and final methods cannot be overridden.

Example

```
class Employee {
    void work() {
        System.out.println("Employee is working");
    }
}

class Tester extends Employee {
    @Override
    void work() {
        System.out.println("Tester is testing the application");
    }
}

public class Demo {
    public static void main(String[] args) {
        Employee e = new Tester();
        e.work(); // Tester is testing the application
    }
}
```

Interview tip: Contrast overriding with overloading: overloading uses the same method name with different parameter lists.

3. Data abstraction

Abstraction means exposing essential operations while hiding internal implementation details. In Java, abstraction is commonly achieved with interfaces and abstract classes. An interface defines a contract; an abstract class can contain both abstract and concrete methods and instance state. For example, a test can use WebDriver without depending on all browser-specific implementation details.

Example

```
interface Payment {
    void pay(double amount);
}

class CardPayment implements Payment {
    @Override
    public void pay(double amount) {
        System.out.println("Paid by card: " + amount);
    }
}

class Demo {
    public static void main(String[] args) {
        Payment payment = new CardPayment();
        payment.pay(500.0);
    }
}
```

Interview tip: Abstraction focuses on what an object does; encapsulation focuses on bundling data and methods and controlling access to state.

4. Tell me about your project

Sample answer to personalize: “I work as a QA/Test Lead on a web-based application. My responsibilities include understanding business requirements, preparing test scenarios and test data, coordinating test execution, tracking defects, and reporting quality status. For automation, I use Java with Selenium WebDriver, TestNG and Cucumber where applicable, following the Page Object Model to keep locators and page actions maintainable. I also work with API testing and SQL validation when those are part of the project. I collaborate with developers, business analysts, product owners, and Scrum team members to clarify requirements, investigate failures, and support releases.” This is a template, not a claim about a specific application. Add your actual product/domain, team size, framework components, test coverage, CI tool, and a measurable result only if true.

Example

```
Suggested structure:
1. Project/domain and business purpose
2. Your role and responsibilities
3. Tech stack and framework design
4. Types of testing and integrations
5. One challenge and how you resolved it
6. A truthful outcome or metric
```

> **Interview tip:** Keep the first answer to about 60–90 seconds, then be ready for follow-up questions about your own contribution.

5. How do you upload a file in Selenium?

For a standard HTML input element with type=file, locate the input and use sendKeys with the absolute path to the file. Selenium generally does not interact directly with the operating system file chooser through ordinary WebDriver locators. If the UI uses a custom upload button, inspect the DOM to find the underlying input. Confirm the file exists on the machine or CI agent executing the test.

Example

```
import java.nio.file.Path;
import org.openqa.selenium.By;
import org.openqa.selenium.WebElement;

// Assumes driver has already been initialized
Path filePath = Path.of("C:", "TestData", "sample.xlsx").toAbsolutePath();
WebElement fileInput =
    driver.findElement(By.cssSelector("input[type='file']"));
fileInput.sendKeys(filePath.toString());

// Click submit/upload if the page requires it
driver.findElement(By.id("uploadButton")).click();
```

> **Interview tip:** If upload is asynchronous, wait for a success message or uploaded-file name rather than relying on Thread.sleep().

6. Difference between WebDriver and Grid

WebDriver is the Selenium API used by test code to control a browser: navigate, locate elements, click, type, and inspect results. Selenium Grid is infrastructure for running WebDriver sessions on remote machines or containers, often across different browsers and operating systems. Grid helps parallelize tests and distribute load; it does not replace the WebDriver API. A RemoteWebDriver client sends commands to a Grid endpoint, which routes the session to a suitable node.

Example

```
// Local browser session
WebDriver driver = new ChromeDriver();

// Remote session (Grid URL and capabilities must match your Grid setup)
ChromeOptions options = new ChromeOptions();
WebDriver remoteDriver = new RemoteWebDriver(
    URI.create("http://localhost:4444").toURL(), options
);
```

> **Interview tip:** A simple analogy: WebDriver is the control interface; Grid is the distributed execution infrastructure.

7. Who attends a Scrum call?

It depends on which Scrum event is meant. In the Daily Scrum, Developers participate to inspect progress toward the Sprint Goal and adapt their plan; a QA engineer who is part of the Developers participates. The Scrum Master ensures the event happens and helps the team understand its purpose, but does not need to act as a status collector. The Product Owner may attend if they are part of the Developers or if the team finds their presence useful, but the Daily Scrum is not primarily a reporting meeting to the Product Owner. Sprint Planning includes the Scrum Team; Sprint Review includes the Scrum Team and relevant stakeholders; Sprint Retrospective is for the Scrum Team.

> **Interview tip:** If the interviewer says “scrum call,” ask whether they mean Daily Scrum, Sprint Planning, Review, or Retrospective.

8. Why do you want to switch jobs?

A professional answer should focus on future growth rather than complaints about the current employer. Example: “I appreciate the experience and opportunities I have gained in my current role. I’m now looking for a role where I can take on broader quality ownership, contribute to automation strategy and framework improvements, and work on challenging products with opportunities to grow technically and as a leader. I believe this position aligns well with my experience and next career goals.” Customize this based on your real motivation—such as responsibilities, learning, role alignment, or career progression. Avoid criticizing managers or sharing confidential information.

> **Interview tip:** Be honest and consistent with your resume. If compensation is a factor, you can mention overall role fit and a fair market-aligned package without making salary the only reason.

9. How do you create your automation framework?

A good framework starts with application needs, team skills, maintainability, execution environment, and reporting requirements. For a Java Selenium framework, a common design includes Maven for dependencies/build, Selenium WebDriver for browser automation, TestNG for test lifecycle and assertions, Cucumber/Gherkin if BDD is required, and Page Object Model for page-level locators and actions. Supporting components can include configuration management, test data utilities, reusable waits, logging, screenshots on failure, reporting, and CI integration. Keep test cases independent and avoid hard-coded credentials or environment-specific paths.

Example

```
Typical structure:
src/test/java
  tests/              # TestNG tests or Cucumber runners
  pages/              # Page Object classes
  steps/              # Cucumber step definitions, if used
  hooks/              # Before/After hooks, if used
  utilities/          # Waits, config, data, logging
src/test/resources
  features/           # .feature files, if using Cucumber
  config/             # Environment configuration
  testdata/           # Non-sensitive test data
pom.xml               # Maven dependencies and build plugins
```

> **Interview tip:** Explain one design choice you made, how tests are executed in CI, how failures are debugged, and how secrets are stored securely.

Fast revision: key phrases

- Fibonacci: each term = sum of previous two terms; iterative solution is O(n) time.
- Overriding: same inherited instance-method signature; runtime polymorphism; use @Override.
- Abstraction: show a contract and hide implementation; interface and abstract class.
- File upload: sendKeys(absolutePath) to input[type=file].
- WebDriver vs Grid: browser control API vs distributed/parallel session execution.
- Scrum: answer according to the specific event, not just the phrase “Scrum call.”
- Framework: explain architecture, separation of concerns, reusable utilities, reporting, and CI.