**🔹 Q1: What’s the difference between Smoke Testing & Sanity Testing?**  
👉 Both validate key functionality — but their purpose & timing are different:  
Smoke Testing is done on the initial build to check if it's stable enough for further testing (also called “build verification”).  
Sanity Testing is a subset of regression, done after minor changes or bug fixes to ensure critical flows are still working on a stable build.  
  
**🔹 Q2: What are Agile Ceremonies?**Agile has 4 key ceremonies that keep the team aligned:  
Sprint Planning – Define goals & tasks for the sprint.  
Daily Stand-up – Quick 15-min sync on progress & blockers.  
Sprint Review – Showcase completed work to stakeholders.  
Sprint Retrospective – Reflect on what went well & what can improve.  
  
**🔹 Q3: Why is the main() method static in Java?**Because the JVM needs to call it without creating an object. Static methods belong to the class, not instances — so main() must be static for the program to start.  
  
**🔹 Q4: What is Runtime Polymorphism?**  
It's when the decision of which method to call is made at runtime — achieved using method overriding. The JVM chooses the method based on the actual object, not the reference type.  
  
**🔹 Q5: Difference between List and Set in Java?**  
List allows duplicates and maintains insertion order.  
Set doesn’t allow duplicates and is unordered (unless using LinkedHashSet).  
  
**🔹 Q6: Method Overloading vs Overriding**Overloading: Same method name, different parameters, within the same class.  
Overriding: Same method signature, but redefined in a subclass (used for runtime polymorphism).

**❓Q1: Have you used Git in your project? Can you explain it?** ✅ Yes, I have used Git extensively. It’s a version control system that helps us manage and track code changes. We maintain a central repository, and each team member works on their local copy. Git helps when multiple teams work on the same project from different locations, making collaboration and code integration smooth and organized.  
  
**❓Q2: Can you list some Git commands you use daily?**  
 ✅ Sure, here are the most commonly used Git commands in my daily work:  
git status – Shows the status of files (tracked/untracked).  
git add <file> – Adds changes to the staging area.  
git commit -m "message" – Commits the staged changes with a message.  
git pull – Pulls the latest code from the remote repo.  
git push – Pushes local changes to the remote repo.  
**git merge – Merges branches together.**  
**❓Q3: How do you handle merge conflicts in Git?** ✅ Since only two testers work on our project, if there's a merge conflict, I first pull the latest changes to my local repo. I then compare the conflicting files, analyze the changes, and discuss with my teammate to ensure all important code is retained. After resolving the conflict, I commit and push the merged code back to the repository.  
  
**❓Q4: Have you worked on Jenkins? How do you create Jenkins jobs?**  
 ✅ Yes, we have a DevOps team for managing Jenkins at a broader level, but we also create our own jobs for internal testing.  
 Here's how I create a Jenkins job:  
Click on “New Item”  
Name the job → Select “Freestyle Project”  
Configure project details  
Under Source Code Management → Select Git → Enter repo URL  
Add a build trigger/schedule if needed  
In Build Section → Add Windows Batch Command (script location)  
Save → Click “Build Now” to trigger  
 You can view logs in the “Console Output” section.  
  
**❓Q5: What is the difference between Smoke and Sanity Testing?** ✅ Both focus on verifying core functionality, but they differ in purpose:  
Smoke Testing is done on the initial build to ensure it's stable enough for further testing.  
Sanity Testing is a subset of regression testing done on stable builds to verify that new changes haven't broken major functionalities.

**🎯 Struggling to explain your Automation Framework experience with Cucumber in interviews?**In my current project, we’ve developed a hybrid automation framework using Selenium with Java, and it’s integrated with TestNG and Cucumber to support BDD (Behavior-Driven Development). We chose this stack because it’s robust, scalable, and easy to maintain.  
We’re using the Page Object Model (POM) design pattern, which really helps keep our code clean and organized. So basically, for every page or screen in the application, we have a dedicated class that contains the web elements and the related methods to interact with those elements. This approach avoids code duplication and makes it super easy to manage any changes in the UI—because we only have to update the locator in one place.  
On top of that, we use Cucumber for writing our test scenarios in plain English using Gherkin syntax, which is great because it improves collaboration between QA, developers, and business analysts. Everyone can read and understand the test cases without needing to know the code. The feature files are linked with step definition classes, where the actual Selenium code is implemented.  
We use TestNG as the test runner to execute the tests, manage test suites, group test cases, and set priorities. It also helps us handle setup and teardown methods using annotations like @BeforeMethod and @AfterMethod. TestNG also provides good reporting out of the box.  
For build and dependency management, we rely on Maven. It handles all the required libraries like Selenium, Cucumber, TestNG, etc., and we define everything in the pom.xml file. It also helps in managing different test environments and makes it easy to run the tests through the command line or CI tools.  
Speaking of CI, we have integrated the framework with Jenkins, so tests are automatically triggered on code commits or during scheduled runs.  
We also use Git for version control and collaboration within the team, and for reporting, we’ve implemented Extent Reports, which gives us detailed and visually clear reports, including logs, screenshots on failure, and execution time.  
Additionally, I’ve worked on data-driven testing, where test data is fetched from external sources like Excel or JSON, and also worked on parallel execution using TestNG to speed up test runs.  
Overall, this framework supports both UI and functional testing in an efficient and maintainable way.

**🎯 Struggling to explain your Automation Framework experience in interviews?**  
So here’s an easy way to structure your answers about the automation framework  
  
Ans: Yes, I have worked on a hybrid automation framework that combines Page Object Model (POM), Data-Driven Testing, and TestNG. It is built using Java, Selenium WebDriver, TestNG for managing test execution, and Maven for dependency and build management. We have also integrated Extent Reports for detailed HTML reports, Log4j for logging, and Jenkins for continuous integration. Our test scripts and data are well organized — we follow a layered structure with base classes for setup, page classes for element locators and actions, test classes for validations, and utility classes for reusable functions like reading Excel data or taking screenshots.  
We use the Page Object Model to keep the code clean and maintainable, and we keep our test data in external files like Excel or JSON to make it flexible. For running tests faster, we use parallel execution in TestNG, and for running across browsers, we use Selenium Grid or Docker containers. The framework also supports API testing using Rest Assured, so we can do end-to-end validations.  
We’ve integrated the whole setup with Jenkins, so tests run automatically after every code push to Git, and everyone gets quick feedback. Some features like retry for failed tests, screenshot capture on failure, and configurable environments (QA/UAT/Prod) make the framework more reliable and user-friendly. We did face challenges like test flakiness and data handling in parallel runs, but we overcame them using explicit waits and dynamic data generation using Faker. Overall, it’s a scalable and reusable framework, and I’ve contributed actively in designing and improving it based on project needs.

**🎯 Struggling to explain your QA experience in interviews?  
Explaining in interviews can be difficult**So here’s an easy way to structure your answers for commonly asked questions.  
   
**❓ Q1: What is TestNG and why is it used?** ✅ A: TestNG (Test Next Generation) is a powerful testing framework inspired by JUnit but with additional features. It’s widely used by Dev/QA teams for unit testing and maintaining clean, scalable code.  
 Key benefits include:  
Creating test suites to run multiple test cases at once  
Grouping test cases logically  
Prioritizing test execution  
Running tests in parallel  
Generating detailed HTML reports  
Using @Test(dependsOnMethods/Groups) to define execution order  
Running the same test with single or multiple data sets  
  
**❓ Q2: How do you run a single method multiple times in TestNG?** ✅ A: You can use the invocationCount attribute of the @Test annotation.  
 Example:  
java  
CopyEdit  
@Test(invocationCount = 3)   
public void testMethod() {   
 // This method will run 3 times   
}  
You can also add threadPoolSize if you want parallel execution.  
  
**❓ Q3: Have you used Git in your projects? How?** ✅ A: Yes, I’ve used Git extensively for version control. We use Git to manage our codebase collaboratively. It allows us to:  
Clone a central repository locally  
Make independent changes and track code history  
Collaborate across global teams  
 It’s essential for syncing, maintaining, and scaling project work in a team environment.  
  
**❓ Q4: What are some Git commands you use frequently?** ✅ A:  
git status – Check current state of working directory  
git add <file> – Stage files for commit  
git commit -m "message" – Commit changes with message  
git pull – Fetch and merge latest changes from remote  
git push – Push local changes to central repository  
git merge – Merge branches  
 These are day-to-day essentials for efficient Git operations.  
  
**❓ Q5: How do you handle a Git merge conflict?** ✅ A: In case of conflicts (since only 2 testers work on the project):  
I pull the latest changes to my local system  
Analyze differences between conflicting and local files  
Discuss with my teammate to ensure nothing important is lost  
Resolve conflicts manually, then commit and push the updated script

**❓ What are the main files used in a Cucumber framework?** ✅ Cucumber mainly includes three files:  
Feature File: Written in Gherkin syntax (plain English) using keywords like Feature, Scenario, Scenario Outline, Given, When, Then, And, Background, and Examples.  
Step Definition File: Contains the Java methods that map to each Gherkin step, acting as the glue between the feature file and the code.  
Test Runner File: Uses frameworks like TestNG or JUnit. This file configures the path of feature files, step definitions, tags, plugins, and listeners.  
  
**❓ What’s the difference between Scenario and Scenario Outline?**  
 ✅  
Scenario is used when we want to run a test once with fixed data.  
Scenario Outline is used for data-driven testing — the same scenario runs multiple times using different data sets defined under the Examples keyword (e.g., different username-password pairs).  
  
**❓ What is the Background keyword used for in Cucumber?**  
 ✅ The Background section defines common preconditions (e.g., login or setup steps) shared across multiple scenarios in a feature file. This avoids repetition and keeps the tests clean.  
  
**❓ What is the use of dryRun in Cucumber?**  
 ✅ The dryRun = true option checks whether every step in the feature file has a matching step definition. It doesn't execute the tests, just validates the mappings.  
  
**❓ What are Hooks in Cucumber?**  
 ✅ Hooks are blocks of code that run before or after scenarios or steps.  
@Before and @After: Run before/after each scenario.  
@BeforeStep and @AfterStep: Run before/after each step.  
 Useful for setup, teardown, screenshots, and logging.  
  
**❓ How do you rerun failed scenarios in Cucumber?**  
 ✅ Use the rerun plugin in the test runner. Failed scenarios get logged in a file. You can point to that file in another runner to re-execute only the failed ones — this helps optimize regression cycles.  
  
**❓ Which is better: Cucumber or TestNG?** ✅ While both are powerful, I prefer Cucumber for its readability and collaboration benefits.  
 The Gherkin syntax allows non-technical stakeholders (like clients, product   
owners, and business analysts) to understand test scenarios easily without diving into code. It bridges the gap between QA, devs, and business teams.

**🔹 Q1: What is StaleElementReferenceException in Selenium?**🟢 It means the element you’re trying to interact with is no longer attached to the DOM — maybe the page refreshed or content changed.  
 📌 Fix: Refresh the page or use Explicit Waits. With Page Object Model, re-initialize elements to avoid this error.  
  
**🔹 Q2: What is a User-Defined Exception in Java?**🟢 It’s a custom exception created by extending the Exception class.  
 📌 Use the throw keyword to raise your custom exception. This helps when you want to handle specific business errors cleanly.  
  
**🔹 Q3: What is Assert in TestNG?**🟢 An assert checks whether your expected result matches the actual result. If they don’t match, the test fails.  
  
**🔹 Q4: Which types of asserts are used in TestNG?**  
🟢 There are two types:  
Hard Assert: Stops the test if a check fails.  
Soft Assert: Continues executing even if the check fails.  
 📌 For soft asserts, use assertAll() at the end to capture all failures.  
  
**🔹 Q5: What is the execution order of TestNG annotations?**  
🟢 Here's the correct sequence:  
text  
  
@BeforeSuite   
@BeforeTest   
@BeforeClass   
@BeforeMethod   
@Test   
@AfterMethod   
@AfterClass   
@AfterTest   
@AfterSuite  
  
**🔹 Q6: What is Priority in TestNG? Can we use negative priorities?**  
🟢 Yes, TestNG allows positive and negative priorities. Lower value executes first.  
 📌 For example: priority = -1 runs before priority = 0.  
 🟡 If a test doesn’t have any priority, it’s executed based on alphabetical order of method names.

**✅ What do you mean by Exceptions in Java?**An exception is an interruption in the normal flow of a program. Java handles exceptions using two categories:  
Checked Exceptions (Compile-time)  
Unchecked Exceptions (Runtime)  
These help developers manage error-prone situations gracefully.  
  
**✅ Difference between throw and throws in Java**throw: Used inside a method to actually throw an exception. It allows only one exception at a time.  
 ➡️ Example: throw new IOException();  
throws: Declares that a method might throw one or more exceptions. Placed in the method signature.  
 ➡️ Example: public void readFile() throws IOException, SQLException  
  
**✅ Which locators do you use in your framework and why?**Mostly:  
ID – it's the fastest and most reliable (unique).  
XPath – useful when ID is not available or dynamic.  
 Other locators include: cssSelector, className, tagName, linkText, partialLinkText.  
  
**✅ Difference between findElement and findElements?**findElement: Returns the first matching element. Throws NoSuchElementException if not found.  
findElements: Returns a list of matching elements. Returns an empty list if none are found.  
  
**✅ Handling multiple windows in Selenium:**Use driver.getWindowHandle() for the current window.  
Use driver.getWindowHandles() for all open windows.  
 To switch:  
for (String window : driver.getWindowHandles()) {  
 driver.switchTo().window(window);  
 if (driver.getCurrentUrl().equals(expectedUrl)) {  
 break;  
 }  
}  
Return to main window using: driver.switchTo().defaultContent();  
  
**✅ Difference between Implicit and Explicit Wait:**Implicit Wait: Applies globally to all elements.  
Explicit Wait: Applied to specific elements/conditions (like visibility, clickability). You can mix both, depending on test needs.  
  
**✅ Common Selenium Exceptions:**NoSuchElementException  
NoSuchWindowException  
NoSuchFrameException  
StaleElementReferenceException  
TimeoutException

**🔹 Q1: What is CI/CD?** A process to build, test, and deploy code automatically — helping teams catch bugs early and deliver faster.  
  
**🔹 Q2: Tools Used?** 1) Jenkins  
 2) Git  
 3) Maven  
 4) Selenium/Rest Assured  
 5) Allure/Extent Reports  
  
**🔹 Q3: How to run Selenium tests in Jenkins?** 1) Install Jenkins  
 Get Jenkins set up on your machine or server.  
 2)Create a New Job  
 Go to Jenkins dashboard → New Item  
 Select Freestyle project and give it a name.  
 3)Connect to Your Code  
 Under Source Code Management, choose Git  
 Paste your repo URL (where your Selenium code is stored).  
 4)Add Build Step  
 If using Maven, add: clean test under Invoke top-level Maven targets  
 Or, use a shell command to run your test script manually.  
 5)Publish Reports (Optional)  
 Add "Publish JUnit test result report"  
 Path: target/surefire-reports/\*.xml  
 6)Save & Build  
 Click Build Now to run the tests.  
  
**🔹 Q4: Benefits?** 1) Fast feedback  
 2) Early bug detection  
 3) Smooth deployments  
 4) Better code quality  
  
**🔹 Q5: Jenkins Pipeline?** A scripted workflow that defines steps like build → test → deploy.  
  
**🔹 Q6: CI vs CD?** CI = Continuous Integration (build/test)  
 CD = Continuous Delivery (auto-deploy after test pass)  
  
**🔹 Q7: What happens when a test fails?** Check logs, alert team, fix or re-run — block deploys if needed.

**🔹 Q1: What is the static keyword in Java?**  
 A: Static means the member belongs to the class and not the object. You can use it with variables, methods, or even inner classes. A static variable will have the same value across all objects of the class.  
  
**🔹 Q2: How can we call static methods or variables?**  
 A:  
 ✅ Directly  
 ✅ Using the class name (e.g., ClassName.methodName();)  
  
**🔹 Q3: Can we access static methods using an object?** A: Yes, but it's not recommended. Java will show a warning suggesting you use class name instead.  
  
**🔹 Q4: How do we call non-static methods and variables?**  
 A: You need to create an object of the class first, then use that object to access non-static members.  
  
**🔹 Q5: Can we overload or override the main method?** A:  
 ✔️ Overload – Yes, you can create multiple main methods with different parameters.  
 ❌ Override – No, main is a static method, and static methods cannot be overridden.

**✅ Q1: What is Data Abstraction? How is it used in Selenium Framework?**  
Answer:  
 Data Abstraction means hiding internal details and showing only the required functionality.  
 In Java, we achieve this using interfaces and abstract classes.  
📌 Selenium Example:  
WebDriver driver = new ChromeDriver();  
WebDriver is an interface.  
ChromeDriver is a class that implements it.  
📌 In Framework (POM):  
 In Page Object Model, we hide locators and actions inside page classes. Tests use only public methods without knowing how they work inside.  
  
**✅ Q2: What is Encapsulation? How is it applied in Automation Framework?**Answer:  
 Encapsulation means keeping variables private and accessing them via public methods.  
📌 Example in POM:  
  
@FindBy(id = "loginBtn")  
private WebElement loginButton;  
  
public void clickLogin() {  
 [**loginButton.click**](http://loginbutton.click/)

();  
}  
Locators are private  
Accessed via public methods  
Keeps data safe and code clean  
  
**✅ Q3: What is Inheritance? How is it used in Selenium Framework?**Answer:  
 Inheritance allows one class to reuse properties and methods from another class.  
📌 Example:  
  
public class BaseTest {  
 public void setup() {  
 // browser setup  
 }  
}  
  
public class LoginTest extends BaseTest {  
 public void testLogin() {  
 setup(); // inherited from BaseTest  
 }  
}  
We keep common logic in Base Class  
Other classes extend it to reuse code  
  
**✅ Q4: What is Polymorphism? How do we use it in Automation?**  
Answer:  
 Polymorphism means performing the same action in different ways.  
✅ Method Overloading:  
Same method name, different parameters.  
  
assertTrue(condition);  
assertTrue(condition, "Custom message");  
  
**✅ Method Overriding:**Subclass provides a new version of a method from its parent.  
📌 Example:  
 Custom WebDriver class overriding findElement() for logging.  
  
**✅ Q4: What is the difference between Interface and Abstract Class?**Answer:  
Interface provides full abstraction with only abstract methods (by default), while an abstract class can have both abstract and concrete methods. An interface supports multiple inheritance; an abstract class does not.