**INTERVIEW QUESTIONS:**

1. What are the Design Patterns you have used in writing the tests in the Framework?
2. How are reusable utilities handled within the framework?
3. Where did you use Inheritance OOPS Concept in your framework?
4. How do you derive the data from external files in the Framework?
5. Did you use Interfaces in the Framework? If so, its usage?
6. How are you achieving Encapsulation in the framework?
7. How does your Framework support parallel runs? How are you writing Thread safe code?
8. Do you have static keywords in the framework? If so, its usage?
9. How are you sending Global Properties to your test at run time?
10. What is the mechanism you use to run only selected set of tests inside the framework?
11. How do you handle flaky tests in the Framework?
12. Does your Framework take screenshot on test failure? How did you implement it?
13. What are the Design Patterns you have used in writing the tests in the Framework?

* Page Object Model.
* All the lines in the test code is derived from page object classes. For example, login page credentials are handled by landingPage class. Then Product Catalogue class handles the add to cart methods.
* Java classes designed upon the pages to be tested!
* Mainly we used page object model, but we have also used PageFactory pattern to create objects and locators.
* A screenshot of a computer code

  Description automatically generated
* Also last method of every page is an action method that directs us to the next page class.

1. How are reusable utilities handled within the framework?

* Two different ways:
  + Abstract components:
    - WaitForElementToAppear
    - goToCartPage
    - goToOrdersPage
  + Base Test: Test related reusable methods go into Base Test. example: initializeDriver, beforetest, afterTest. And All of the test cases will inherit BaseTest!
* How to use them? To use abstract components, abstract components class should be inherited using extends keyword. Then, super keyword is used in the constructor of the class, letting abstract class know about its objects.

1. Where did you use Inheritance OOPS Concept in your framework?

* When a method is need to be used many times, it is desirable to put these methods in an abstract or parent class so that in every class definition with extend keyword, those methods are also inherited.
* To keep the test class clean, it is also desired to take common concepts from a parent class.

1. How did you derive the data from external files in the Framework?

* PurchaseOrder.json is used to get data from external file. HashMap data structure is used to gather data from json.
* There is an annotation in TestNG called “DataProvider”. In TestNG, the @DataProvider annotation allows you to run a single test multiple times with different sets of data. This is especially useful for test cases that need to validate multiple input values or scenarios without writing separate test methods for each case.

**What is @DataProvider?**

* @DataProvider is an annotation in TestNG that enables parameterized testing. It allows you to supply multiple sets of data to a single test method, which is particularly useful in data-driven testing.

**How Does @DataProvider Work?**

* @DataProvider methods return a 2D Object[][] array, where each inner array represents a set of parameters passed to the test method.
* Each row in the 2D array corresponds to one test execution with its own set of arguments.
* TestNG will call the test method for each data set, resulting in multiple test executions with different inputs.

**Creating a Simple @DataProvider**

* Define a method annotated with @DataProvider and return a 2D array of Object. You can then link it to a test method by setting the dataProvider attribute in the @Test annotation.
* A computer screen shot of a program code

  Description automatically generated

in our example:

A computer code with black text

Description automatically generated

SUBQUESTIONS:

1---- **Advanced Usage of @DataProvider**

* **Providing External Data**: You can load data from external sources like Excel, databases, or JSON files by writing logic in the @DataProvider method to fetch and structure the data.
* **Parallel Execution**: You can use @DataProvider(parallel = true) to run tests with different data sets concurrently, improving execution speed in large test suites.

**2------Benefits of Using @DataProvider**

* + **Code Reusability**: The same test method can validate multiple input scenarios, avoiding redundant code.
  + **Easier Maintenance**: It’s simpler to modify test data in the @DataProvider than to change individual test cases.
  + **Data-Driven Testing**: @DataProvider is essential in scenarios where the test logic is the same, but the data varies, making it a crucial feature for automation and validation.

### Common Interview Questions about @DataProvider

1. **How is @DataProvider different from passing parameters with XML files?**
   * XML parameters are static and defined upfront, while @DataProvider is dynamic and allows for flexible, data-driven testing within the code.
2. **How would you handle different data types in a @DataProvider?**
   * @DataProvider returns an Object[][] array, allowing you to pass any data type, including complex objects, by casting them appropriately in the test method.
3. **Can @DataProvider use data from external files?**
   * Yes, you can write code in the @DataProvider method to fetch data from external sources like databases, files, or APIs.
4. **What are some limitations of @DataProvider?**
   * A key limitation is that the data is defined within the @DataProvider method, meaning large data sets may make the method bulky. For very large data, consider using lightweight file formats or other testing frameworks.
5. **How can you usse json data in your code?**
   * With HashMaps.
   * First read json to a string with the help of FileUtils.readFileToString....
   * Then, There is a dependency called Jackson Databind which converts string to hashmap
   * A computer screen shot of a code

     Description automatically generated
6. **What are some limitations of @DataProvider?**
   * A key limitation is that the data is defined within the @DataProvider method, meaning large data sets may make the method bulky. For very large data, consider using lightweight file formats or other testing frameworks.
7. Did you use Interfaces in the Framework? If so, its usage?

* First and the foremost, we use WebDriver objects. WebDriver is an interface as far as I know.
* There are also TestNG Interfaces. Listeners class implements an interface called ITestListener

### Key Points to Cover

1. **Why Use Interfaces in Selenium Automation?**
   * Interfaces enable abstraction, allowing you to define behaviors (methods) without specifying implementations. This makes code more modular and adaptable to changes, as you can define high-level actions in interfaces and implement them in different ways.
   * Interfaces help manage dependencies, especially in larger test frameworks where you may have multiple types of actions or elements to interact with.
2. **Common Ways Interfaces are Used in Selenium Projects**
   * **Defining Page Actions in the Page Object Model (POM)**
     + In POM, interfaces are used to define actions or behaviors for each page. This approach allows for flexibility in implementing or modifying actions without changing the core test logic.
     + For example, you might define an interface for a LoginPageActions to outline methods like login(String username, String password), navigateToLoginPage(), and getLoginStatus().
     + Implementations of LoginPageActions can vary depending on the environment (e.g., web vs. mobile) without altering the main test code.

**Abstraction of WebDriver Actions**

In large frameworks, interfaces can be used to abstract WebDriver methods. For example, you could have an ElementActions interface that defines methods for click, sendKeys, getText, etc.

* This interface is then implemented by classes that use different WebDriver instances (e.g., for different browsers or environments) or even different implementations for handling retries, error handling, or logging. 

**nhancing Reusability Across Test Cases**

* + Using interfaces for common actions, such as navigation and logging in, helps make the code reusable and maintainable. Since test cases only rely on interfaces, implementations can be swapped easily.
* **Handling Cross-Browser Testing**
  + Interfaces are particularly useful in cross-browser testing. You can define a common interface for browser actions and then create multiple implementations tailored for different browsers (e.g., Chrome, Firefox, etc.). This allows test cases to remain browser-agnostic while utilizing the correct implementation based on runtime conditions.

3- **Benefits of Using Interfaces in Selenium Automation**

* **Modularity**: Interfaces separate the definition of actions from their implementations, making it easy to modify or extend behavior without impacting test cases.
* **Easy to Mock**: Interfaces facilitate mocking and stubbing, which is useful for unit testing in a test automation project.
* **Dependency Injection**: Interfaces work well with dependency injection (e.g., using frameworks like Spring), making it easier to inject mock or alternative implementations.

4- **Real-World Example**

* If you’ve used interfaces in a real Selenium project, explain how they helped with scalability or reusability. For example, mention if you used interfaces for cross-browser compatibility or to handle different environments (web vs. mobile) without changing test case logic.
* ITestListener:

A screenshot of a black screen

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer error message

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer error

Description automatically generated

1. How are you achieving Encapsulation in the framework?

A screenshot of a black and white screen

Description automatically generated

A screen shot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A black and white text on a black background

Description automatically generated

1. How does your Framework support parallel runs? How are you writing Thread safe code?

* In the testNG xml file, we can define parallel tests. In parallel tests, each test run in a separate thread. But in reporting, there can be issues. To solve this issue ThreadLocal is used to give each thread an ID:

xml:

A screen shot of a computer code

Description automatically generated

ThreadLocal:

A screen shot of a computer program

Description automatically generated

1. Do you have static keywords in the framework? If so, its usage?

* You should not use static keyword for configuration variables like WebDriver object. You can use static keyword in never changing data to decrease disk space usage, but for concurrency issues, static keyword should be carefully used.

A black and white sign with white text

Description automatically generated

A black background with white text

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

1. How are you sending Global Properties to your test at run time?

A computer screen shot of a computer code

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

1. What is the mechanism you use to run only selected set of tests inside the framework?

There are 6 different ways for it:

A screenshot of a computer program

Description automatically generated

Groups

First, tag the class with the group name:

A screen shot of text

Description automatically generated

Then in testNG xml file (in your test suite),

A screen shot of a computer program

Description automatically generated

in pom.xml file, you should create the profile for the group:

A screen shot of a computer program

Description automatically generated

A screenshot of a test

Description automatically generated

A screenshot of a black and white screen

Description automatically generated

A black background with white text

Description automatically generated

1. How do you handle flaky tests in the Framework?

Using Retry mechanism

A screen shot of a computer program

Description automatically generated

A screenshot of a computer code

Description automatically generated

A screenshot of a computer

Description automatically generated

A black and white text on a black background

Description automatically generated

A black screen with white text

Description automatically generated

1. Does your Framework take screenshot on test failure? How did you implement it?

Can use a try and catch block. Whole test goes into try block, if code fails, catch block takes the screenshot. But this is not the right way.

Correct way is to use ITestListener’s onTestFailure(ITestResult result) method.

A screen shot of a computer code

Description automatically generated

1. Explain Framework Architecture.

This is the whole process...

**OTHER QUESTIONS:**

1. What does Singleton mean in Testing?

A black and white screen with white text

Description automatically generated

A screenshot of a computer program

Description automatically generated

A black and white screen shot of a black and white screen shot of a black and white screen shot of a black and white screen shot of a black and white screen shot of a black and white

Description automatically generated

1. TestNG vs Junit?\_

A screenshot of a black screen

Description automatically generated

A black and white screen with white text

Description automatically generated

1. What is proxy?

A black screen with white text

Description automatically generated

A black screen with white text

Description automatically generated

1. adf
2. ads
3. fadsf
4. adsf

**STUDYING TEST AUTOMATION INTERVIEW:**

**What are the best sites to practice selenium online projects?**

I think the more the types of elements on a website, the more it helps in practicing the automation using selenium.

You can practice Selenium on any website. You can choose a website based on the following buckets:

eCommerce: this is a leading market for a website and provides lots of opportunities for automation. in this category, My favorite is amazon’s website because its the perfect example of a site with most of the web elements present and you’ll have to use most of the locator types in order to automate adding a particular item your cart and order it.

Job Portals: this category gives you a different type of look and feel. They give you a simpler UI and are good for practicing attaching files.

Social media: everyone is on social media. Automating something like sharing or commenting etc gives you more confidence for practicing and applying the learnings.

Entertainment: Mostly for flash/ Iframe related automation practice.

My main criteria to decide on a website is the category of the website and the content (web elements).

**some websites to practice selenium:**

Basic Element Interaction: <https://the-internet.herokuapp.com/>

Form Handling: <https://toolsqa.com/>

Web UI (bug testing): <https://candymapper.com/>

Dynamic Content: <https://the-internet.herokuapp.com/dynamic_loading>

Online shop Simulation: <https://www.saucedemo.com/>

E-commerce: <https://www.demoblaze.com/>

Multi-browser and Cross-platform Testing: CrossBrowserTesting

Advanced Element Interaction: <https://the-internet.herokuapp.com/upload>

Some basic questions:   
  
**Q: What is a web element?**

**A:** A web element refers to any component or element on a web page, such as buttons, text boxes, drop-down lists, etc. Selenium provides methods to locate and interact with web elements.

**Q: How can I locate web elements using Selenium?**

**A:** Selenium provides multiple methods to locate web elements, such as xpath, CSS selectors, class name, id, name, link text, and partial link text.

**Q: What is a frame in Selenium?**

**A:** A frame, also known as an iframe, is an HTML element that allows content from another HTML document to be embedded within the current document. Selenium provides methods to switch between frames for interacting with elements inside them.

**Q: What is an alert in Selenium?**

**A:** An alert is a pop-up window that appears on a web page to notify the user or request input. Selenium provides methods to handle alerts programmatically.

**Q: What are some best practices for using Selenium?**

**A:** Some best practices for using Selenium include using explicit waits instead of hard coding wait times, using Page Object Model design pattern, organizing tests into separate test classes, and maintaining clear and descriptive test names.

**Q: Can I write tests using Selenium in other languages?**

**A:** Yes, Selenium supports multiple programming languages including Java, Python, C#, and Ruby.

**Q: What does the "navigator" term refer to in Selenium?**

**A:** In Selenium, the "navigator" term refers to the WebDriver interface's method that allows navigation actions on a web page, such as navigating to a URL, navigating back, refreshing the page, etc.

The **Page Object Model** is a best practice in Selenium that encourages separating the test code from the web element locators and actions. For each page, create a class representing that page, encapsulating the elements and methods that interact with them. This makes tests more maintainable, as UI changes require updates only in the page object, not across all tests.

# JAVA RECALL:

A black screen with white text

Description automatically generated

Abstraction:

A screenshot of a computer

Description automatically generated

A black screen with white text

Description automatically generated

A screenshot of a computer

Description automatically generated



A screen shot of a computer program

Description automatically generated

A screen shot of a computer program

Description automatically generated

**Question:** *"Can interfaces have methods with implementation?"*

**Response:** "By definition, interfaces are contracts with abstract methods. However, since Java 8, interfaces can include methods with implementation using default and static methods. Default methods allow interfaces to evolve without breaking backward compatibility by providing a default implementation that classes can use or override. Static methods in interfaces belong to the interface itself and are not inherited by implementing classes. Additionally, from Java 9, interfaces can also have private methods to share common logic between default methods."

Encapsulation:

A screenshot of a computer program

Description automatically generated

A black screen with white text

Description automatically generated

A screen shot of a computer program

Description automatically generated

A black screen with white text

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

**Inheritance:**

A black and white text on a black background

Description automatically generated

A screen shot of a computer program

Description automatically generated

A screenshot of a black screen

Description automatically generated

A computer screen shot of a program code

Description automatically generated A screen shot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

NOTE: You cannot inherit two superclasses. You can only implement two or more interfaces but not for abstract or super classes.

**Polymorphism:**

A black screen with white text

Description automatically generated

A black screen with white text

Description automatically generated

A screenshot of a computer program

Description automatically generated

A black and white text on a black background

Description automatically generated

A screen shot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

A computer screen shot of a program

Description automatically generated

A black screen with white text

Description automatically generated

Super keyword:

A screenshot of a computer code

Description automatically generated

**Dynamic Method Dispatch**: This is the mechanism that supports polymorphism in Java. When the JVM calls the draw() method, it dynamically selects the method to execute based on the actual object (Circle or Square), not the reference type (Shape).

Method calls are resolved based on the **reference type at compile time** and the **object type at runtime**.