Q1. What is automation testing?

1. Automation testing is a process of automating the manual process to test the application/system.
2. Automation testing involves the use of a separate testing tool which lets you create test scripts which can be executed repeatedly and doesn't require any manual intervention.

Q2. What are the advantages of automation testing?

1. Automation testing supports both functional and performance tests on an application,
2. It supports the execution of repeated test cases,
3. It facilitates parallel execution,
4. It improves accuracy because there are no chances of human errors,
5. It saves time and money.

Q3. What is Selenium? Selenium is one of the most popular automated testing suites. Selenium is not just a single tool or a utility, rather a whole package of several testing tools. It was originally developed by Jason Huggins in 2004.

1. Operating Systems: Android, iOS, Windows, Linux, Mac, Solaris.
2. Browsers: Google, Chrome, Mozilla Firefox, Internet Explorer, Edge, Opera, Safari, etc.
3. Programming Languages: C#, Java, Python, PHP, Ruby, Perl, JavaScript.

Q4. What are the different ways of locating a web element in Selenium?

- Locators are used to identify a web element uniquely within the webpage.
- ID, ClassName, Name, TagName, LinkText, PartialLinkText, CSS Selector, Xpath.

Q5. How can you find if an element is displayed/present on the screen? - isDisplayed()

Q6. How to handle hidden elements in selenium webdriver? Javascript Executor.

Q7. How to type text in a textbox using Selenium? - sendKeys("String");

Q8. And limitations of selenium?

1. Selenium supports testing of only web-based applications.
2. Captcha and Barcode readers cannot be tested using Selenium.
3. Reports can only be generated using third-party tools like TestNG or JUnit.
4. The user is expected to possess prior programming language knowledge.

9. What are the different methods to refresh a web page in WebDriver?

1. driver.navigate().refresh();
2. driver.get (driver.getCurrentUrl()); // reopen the same url.
3. Actions act = new Actions(driver);
    a. act.sendKeys(Keys.F5).perform();
4. JavascriptExecutor js = (JavascriptExecutor) driver;
    a. js.executeScript("location.reload()");

10. What are the different components of Selenium?

1. Selenium (IDE) -
    a. Selenium IDE is implemented as a Firefox extension which provides record and playback functionality on test scripts.
    b. It allows testers to export recorded scripts in many languages like HTML, Java, Ruby, RSpec, Python, C#, JUnit and TestNG.
    c. Selenium IDE has limited scope, and the generated test scripts are not very robust, and portable.

2. Selenium RC - Selenium RC is a server that allows a user to create test-scripts in the desired programming language.

3. Selenium WD: WD is a different tool altogether that has various advantages over Selenium Remote Control. WebDriver directly communicates with the web browser and uses its native compatibility to automate.

4. Selenium Grid -
    a. Selenium Grid facilitates you to distribute your tests on multiple machines and all of them at the same time.
    b. So, you can execute tests on Internet Explorer on Windows and Safari on Mac machines using the same text script.
    c. It reduces the time of test execution and provides quick feedback.

11. Commonly used tools that are used for Functional Automation (Selenium **) and Non-Functional Automation (** JMeter **).**

12. List out some of the Automation tools which could be integrated with Selenium to achieve continuous testing? Maven, Jenkins, & Docker. It can also be integrated with tools such as TestNG, & JUnit for managing test cases and generating reports.

13. What are the different types of annotations which are used in Selenium? Test, Before, After, Ignore, BeforeClass, AfterClass, RunWith

Q14. What is the difference between type keys and type commands? TypeKeys() will trigger JavaScript events in most of the cases whereas type() won't.

Q16. How to maximize browser windows? driver.manage().window().maximize();

15. List some of the test types that are supported by Selenium? Functional Testing, UI testing (black box), sanity Testing, Smoke Testing, Regression Testing, Cross Browser Testing.

17. How many types of web drivers apis are available in WebDriver in selenium? AndroidDriver, ChromeDriver, FirefoxDriver, InternetExplorerDriver, iPhoneDriver, RemoteWebDriver, SafariDriver, OperaDriver.

18. What do you mean by XPath?

1. XPath is used to locate a web element based on its XML path.
2. The fundamental behind locating the elements using XPath → traversing between various elements across the entire page,
3. and thus enabling the user to find an element with the references to another element.

19. Explain the difference between Absolute XPath v/s Relative XPath?

Absolute XPath :

1. Contains the complete path location → root HTML tag to the specific elements.
2. Very specific and breaks easily if the structure changes.
3. Slower because it traverses from the root.
4. Single slash (/) is used to create XPath with absolute path.
5. Syntax: //html/body/tag[index-1]/tag[index-2]/tagN[index]

Relative XPath :

1. Relative XPath is starting simply by referencing the element you want and go from the particular location.
2. More flexible and reliable for dynamic elements.
3. Faster as it directly searches for the matching element.
4. Double slash (//) is used to create XPath with the relative path.
5. Syntax: //input[@id='username']

Q20. How to click on a hyperlink using link Text and partial link text?

driver.findElement(By.linkText("Google")).click();
driver.findElement(By.partialLinkText("Goo")).click();

Q21. What is the difference between findElement() and findElements()?

1. findElement:
   - Returns the first matching element.
   - Throws a NoSuchElementException if the element is not found.

2. findElements:
   - Returns a list (List<WebElement>) of all matching elements.
   - If no elements are found, it returns an empty list instead of throwing an exception.

Q22. How can we launch different browsers in Selenium WebDriver?
WebDriver driver = new ChromeDriver/InternetExplorerDriver/FirefoxDriver();

23. What are the types of navigation commands?

1. driver.navigate().to("url");
2. driver.navigate().back()
3. driver.navigate().refresh()
4. driver.navigate().forward().;

24. What are the different exceptions in selenium ?

1. NoSuchElementException – Raised when an element is not found in the DOM.
2. NoSuchWindowException – Occurs when trying to switch to a window that doesn't exist.
3. NoSuchFrameException – Thrown when trying to switch to a frame that is not found.
4. NoSuchSessionException – Raised when a session is no longer active or valid.
5. StaleElementReferenceException – This exception occurs when the webdriver tries to interact with an element that no longer exists in the dom or is not valid.
    a. Common reasons are - the element gets refreshed or reloaded, the element is removed and re-added dynamically.
6. NoAlertPresentException – Occurs when switching to an alert that is not present.
7. InvalidSelectorException – Thrown when an invalid XPath/CSS selector is used.
8. ElementNotVisibleException – Raised when an element exists but is not visible.
9. ElementNotSelectableException – Occurs when an element is present but not selectable.
10. TimeoutException – Happens when a command exceeds the specified wait time.

Q25. What is the difference between setSleep() and sleep method() ?

- Thread.sleep() (Hard Wait) : Pauses execution for a fixed time. Drawbacks: Slows down test execution, Not dynamic; waits even if the element is ready earlier.
- setSleep() : it sets up speed that will apply a delay time before every selenium operation.

Q26. Difference between quit and close?

- close(): Closes the current browser window where the WebDriver is focused. If multiple windows are open, only the active window is closed.
- quit(): Closes all browser windows opened by the WebDriver. Ends the WebDriver session completely.

27. How can we Press ENTER Key in a Textbox using Selenium?

```
Actions a = new Actions(driver); a.keyDown(Keys.ENTER).keyUp(Keys.ENTER).build().perform();
```

OR-

```
WebElement searchBox = driver.findElement(By.name("q"));
searchBox.sendKeys("Selenium WebDriver"+Keys.ENTER);
```

ctrl+a -

```
a.keyDown(Keys.CONTROL).sendkeys("a").keyUp(Keys.CONTROL).perform();
```

28. Write a code snippet to perform mouse hover in WebDriver.

```
WebElement product = driver.findElement(By.xpath("//button[@id='products-dd-toggle']"));
WebElement appTesting = driver.findElement(By.xpath("//div[@role='tablist']//button[contains(@title,'App')]"));

Actions act = new Actions(driver);

// mouse hover - // build - create an action ,      // perform - complete an action
act.moveToElement(product).moveToElement(appTesting).build().perform();
```

29. How do you perform dragAndDrop operations in WebDriver?

```
Actions act = new Actions(driver);

// 1
WebElement oslo = driver.findElement(By.xpath("//div[@id='box1']"));
WebElement norway = driver.findElement(By.xpath("//div[@id='box101']"));
Thread.sleep(2000);
System.out.println("1. draggin oslo to norway");
act.dragAndDrop(oslo, norway).build().perform();
```

30. Write a code snippet to perform a right-click on an element in WebDriver.

```
WebElement rightclick = driver.findElement(By.xpath("//p//span"));

Actions act = new Actions(driver);

act.contextClick(rightclick).perform();
```

31. Write a code snippet to perform a doubleClick on an element in WebDriver.

```
WebElement dc = driver.findElement(By.xpath("//input[@value='Double Click']"));

Actions a = new Actions(driver);
a.doubleClick(dc).build().perform();
```

32. How can we get the text of a web element? Get command is used to get the inner text. It returns a string value. String Text = driver.findElement(By.id("Text")).getText();

33. How to retrieve CSS properties of an element? - Using a getCssValue() method.

```
WebElement button = driver.findElement(By.xpath("//input[@id='nav-search-submit-button']"));

// retrieving the css properties of button -
String txtColor = button.getCssValue("color");
System.out.println("text-color is: " + txtColor);
```

color, font-size, background-color, height, width, visibility, display, border, etc.

34. How to select a value in a dropdown? - Using select class for select dropdowns.

```
// capturing the drop down -
WebElement dpCountry = driver.findElement(By.xpath("//select[@id='country']"));

// use select for select drop down - passing to select class object

// way 1 -
// Select dpC = new
// Select(driver.findElement(By.xpath("//select[@id='country']")));

// way 2 -
Select dpC = new Select(dpCountry);

dpC.selectByVisibleText("France");
dpC.selectByValue("uk");
dpC.selectByContainsVisibleText("Ger");
dpC.selectByIndex(4);

// capture all options from dropdown -
List<WebElement> op = dpC.getOptions(); /// returns all options
System.out.println("number of options available: " + op.size()); // 10

// using for each
for (WebElement txt : op) {
    System.out.println("Option: " + txt.getText());
}

// for loop
for (int i = 0; i < op.size(); i++) {
    System.out.println("options: " + op.get(i).getText());
}
```

35. How to capture a screenshot in WebDriver? - using the Takes Screenshot interface.

```
// 1. full page -

The driver (WebDriver instance) is cast to TakesScreenshot, allowing it to capture screenshots -
TakesScreenshot ts = driver;
Captures a screenshot of the entire web page and stores it as a temporary file -
File sourcefile1 = ts.getScreenshotAs(OutputType.FILE);
Creates a new file path in the screenshots folder inside the project directory -
File targetfile1 = new File(System.getProperty("user.dir") + "\\screenshots\\ss1.png");
Moves the temporary screenshot file to the specified location and renames it as ss1.png -
sourcefile1.renameTo(targetfile1);

// 2. specific area/element/poster -
WebElement loc_section = driver.findElement(By.xpath("//a[normalize-space()='Test Screenshots in Selenium']"));
File sourcefile2 = loc_section.getScreenshotAs(OutputType.FILE);
File targetfile2 = new File(System.getProperty("user.dir") + "\\screenshots\\ss2.png");
sourcefile2.renameTo(targetfile2);
```

36. Have you created any framework? No, I didn't get a chance to create a framework from scratch. I have used the framework which is already available.

37. What is the System.setProperty() method?

1. This method is required to initialize the webdriver for different browsers like chrome, etc.
2. It is used to set the properties of browsers. It takes two parameters :
    a. Property name - name of the browser-specific driver.
    b. Value - the path to the browser driver.
3. For example - System.setProperty("webdriver.chrome.driver","path/to/chromdriver.exe");

38. How to delete cookies in selenium?

1. driver.manage().deleteAllCookies(); - clear all session data before starting tests.
2. driver.manage().deleteCookieNamed("C_name"); - remove a specific cookie by name.
3. Cookie cookie = driver.manage().getCookieNamed("cookieName");
    a. driver.manage().deleteCookie(cookie);

39. How to deal/locate frames in WebDriver?

Frame is inserted in another document within the current document.

Ways to switch to a frame -

1. driver.switchTo().frame(index);
2. driver.switchTo().frame("frameName/frameID");
3. driver.switchTo().frame(frameElement);

Switching back to main page -

1. driver.switchTo().defaultContent(); - switch back to main page.
2. driver.switchTo().parentFrame(); - to move one level up from a nested frame.

```
// frame 1-
System.out.println("switching to frame 1 is under process");
// i. find the frame
WebElement fm1 = driver.findElement(By.xpath("//frame[@src='frame_1.html']"));
// ii. switch to that frame
driver.switchTo().frame(fm1);
// iii. locate the element
driver.findElement(By.xpath("//input[@name='mytext1']")).sendKeys("princy");
System.out.println("switching to frame 1 done");
// switch back
driver.switchTo().defaultContent();
```

40. Explain assertion in Selenium?

Used to verify that the app. under test behaves as expected. Used to verify expected vs. actual results in test cases. If an assertion fails, the test case stops execution (for hard assertions) or continues execution (for soft assertions).

Hard Assertion : When a failed condition should immediately stop execution.

1. Assert.assertEquals(1,1); //pass
2. Assert.assertNotEquals(1,1); //fails
3. Assert.assertTrue(true); //pass
4. Assert.assertFalse(true); //fails
5. Assert.assertNotNull(yaha kuch hona chahiye=pass, agar kuch nahi h=fails);
6. Assert.assertNull(yaha kuch nahi hona chahiye=pass, agar kuch hai=fails);
7. Assert.fail(); //fails

Soft Assertion : When you want to continue execution and check multiple conditions. Ex-

41. What is the difference between Assert vs Verify? *Same as hard and soft assertions…*

- The difference of commands mainly lies in how they handle test failures.
- Assert and verify command checks condition is true or false.

Assert: If the condition is false → Stops test execution. Use Assert when a failure means the test cannot proceed.

Verify: If the condition is false → Continues test execution & logs the failure. Use Verify when a failure should be logged but the test should continue.

```
SoftAssert sa = new SoftAssert();
String expectedTitle = “abc”; String actualTitle = “abcdefg”;
sa.assertEquals(actualTitle, expectedTitle );  // if this fails, the execution continues.
sa.assertAll(); // reports all verification failures
```

42. What is a Page Object Model?

- It is a design pattern used in selenium to create an object repository for web elements.
- It helps in maintaining and reusing code by separating the test logic from UI elements.
- Advantages: Readability, Maintaining, Code Reusability, Scalability (Easy to expand the framework as the application grows).

43. What is Page Factory?

- Page Factory is an implementation of Page object model that uses the @FindBy annotation to initialize web elements dynamically.
- Advantages: Faster Execution, Better Performance, Improved Code Readability (Reduces the use of driver.findElement()).

44. Difference between Page object model and Page factory?

- Page object model is a class that represents a web page and holds the functionality and members.
- Page factory is a way to initialize the web elements you want to interact with within the page object when you create an instance of it.

45. How can we handle web-based pop-up? Using an alert interface.

Way 1 - Without creating an alert interface object.

```
driver.findElement(By.xpath("//button[@onclick='jsAlert()']")).click();
driver.switchTo().alert().accept();
```

Way 2 - with creating an alert interface object.

```
// 2. contain OK and cancel button -
    driver.findElement(By.xpath("//button[@onclick='jsConfirm()']")).click();
    Alert alrt1 = driver.switchTo().alert();
    alrt1.dismiss();

// 3. alert box with textbox, OK and cancel button -
    driver.findElement(By.xpath("//button[@onclick='jsPrompt()']")).click();
    Alert alrt2 = driver.switchTo().alert();
    alrt2.sendKeys("I m Princy");
    alrt2.accept();
```

46. What is the difference between get() and navigate() ?

driver.get(URL)	driver.navigate().to(URL)
1. Open the specified url.	Open the specified url.
2. Does not support.	Supports back, forward, refresh navigation.
3. Wait until the page fully loads.	Don't wait.
4. Use get() when you just need to open an url.	Use navigate().to() when you might need to navigate back, forward or refresh.

47. What is the wait? How many types of waits in selenium?

1. Wait is used to manage the synchronization between the test script and web elements.
2. To ensure that the script does not fail due to elements not being visible immediately.

1. Implicit Wait:

- Tell the web driver to wait for a certain amount of time and before throwing an "no such element Exception".

- The default setting is 0. Applies to all elements in the session.
- The main disadvantage is that it **slows down test performance.** Suppose, you set the waiting limit to be 10 seconds, and the elements appear in the DOM in 11 seconds, your tests will fail because you told it to wait a maximum of 10 seconds.
- Syntax-

```
driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(5)) ;
```

2. Explicit Wait:

- Waits for a specific condition to be met for an element before proceeding and before throwing an "Element Not Visible Exception".
- It gets first preference and then implicit wait (if both are mixed).
- Syntax-

WebDriverWait mywait = new WebDriverWait(driver, Duration.*ofSeconds*(5));
mywait.until(ExpectedConditions.*visibilityOfElementLocated*(By.xpath("xyz"))).sendkeys("adm");
mywait.until(ExpectedConditions.*elementToBeClickable*(By.xpath("xyz"))).click();

3. Fluent Wait:

- The fluent wait is used to tell the web driver to wait for a condition, as well as the frequency with which we want to check the condition before throwing an "Element Not Visible Exception".
- Similar to Explicit Wait but provides more flexibility and allows polling intervals.

```
public static void main(String[] args)
{
    WebDriver driver = new ChromeDriver();

    // Fluent wait Declaration -
    Wait<WebDriver> mywait = new FluentWait<WebDriver>(driver)
                .withTimeout(Duration.ofSeconds(10)) // this defines the total amount of time to wait for
                .pollingEvery(Duration.ofSeconds(2)) // this defines the polling frequency
                .ignoring(NoSuchElementException.class); // this defines the exception to ignore

try {
    driver.get("https://opensource-demo.orangehrmlive.com/web/index.php/auth/login");
    driver.manage().window().maximize();

    // Fluent wait use -
    WebElement txtUsername = mywait.until(new Function<WebDriver,WebElement>(){
        public WebElement apply(WebDriver driver){
            return driver.findElement(By.xpath("//input[@placeholder='Username']"));
        }
    });

    txtUsername.sendKeys("Admin");
    System.out.println("admin logged in");

    driver.findElement(By.xpath("//input[@placeholder='Password']")).sendKeys("admin123");
}finally {
    driver.close();
}}
```

48. How to handle multiple windows in selenium webdriver?

1. driver.getWindowHandle() - Returns: the current window handle.
2. driver.getWindowHandles() - Returns: A set of window handles which can be used to iterate over all open windows.

```
WebDriver driver = new ChromeDriver();

// launching at 0 index
driver.get("https://opensource-demo.orangehrmlive.com/web/index.php/auth/login");
driver.manage().window().maximize();

// launching at 1 index
driver.findElement(By.linkText("OrangeHRM, Inc")).click();

// fetching id's
Set<String> windows_Ids = driver.getWindowHandles();

// way 1 - converting a set to list.
// way 2 - using for each loop.
```

```
// way 1 -> to get individual id from a set, convert the set collection into list -
        List<String> winList = new ArrayList<String>(windows_Ids);
        String parentID = winList.get(0);
        String childID = winList.get(1);

        // switch to child win coz the driver will always focus on parent windwow.
        driver.switchTo().window(childID);
        System.out.println("title of child window: " + driver.getTitle());

        driver.switchTo().window(parentID);
        System.out.println("title of parent window: " + driver.getTitle());
```

```
// way 2
        for (String windows_Idss : windows_Ids) {
            String title = driver.switchTo().window(windows_Idss).getTitle();
            if (title.equals("OrangeHRM")) {
                System.out.println(driver.getCurrentUrl());
            } else {
                System.out.println(driver.getCurrentUrl());
            }
        }
```

49. What is JavascriptExecutor and its use in selenium webdriver?

1. JavascriptExecutor is an interface in Selenium WebDriver that allows the execution of JavaScript code within the browser.
2. It helps perform actions that are not possible directly using WebDriver methods.
3. It is mainly used when standard Selenium commands fail or when we need additional browser-level interaction.

Scrolling the Page: Scroll up, down, or to a specific element. Horizontally -

```
ChromeDriver driver = new ChromeDriver();
driver.manage().window().maximize();
driver.get("https://www.geeksforgeeks.org/");

JavascriptExecutor js = driver;

-- performing operation -

1. default/current location first -
System.out.println(js.executeScript("return window.pageYOffset;")); // 0

2. scroll down a bit -
js.executeScript("window.scrollBy(0,1500)");

3. scroll down till any element is visible -
WebElement loc = driver.findElement(By.xpath("//h2[normalize-space()='GfG School']"));
js.executeScript("arguments[0].scrollIntoView();", loc);

4. scroll down till end -
js.executeScript("window.scrollBy(0,document.body.scrollHeight)");

5. scroll up to the begining -
js.executeScript("window.scrollBy(0,-document.body.scrollHeight)");
```

Handling Hidden Elements: it is an example of zooming using javascript.

```
JavascriptExecutor js = driver;

// Check the default zoom level
Object zoom1 = js.executeScript("return window.devicePixelRatio;");
System.out.println("Zoom level: " + zoom1);

Thread.sleep(2000);
js.executeScript("document.body.style.zoom='50%'"); // set zoom=50%
```

Highlighting Elements: Temporarily change an element's background color for debugging. Ex- js.executeScript(“arguments[0].style.border=’3px solid red’”,element);

Fetching Browser and Page Details: Get the page title, URL, or inner text. Ex - String title = (String) js.executeScript("return document.title;");

Interacting with Elements: Click elements, enter text, or modify attributes when WebDriver methods fail.

```
-- Instantiate a Driver class.

way 1 -
ChromeDriver driver = new ChromeDriver();

way 2 -
WebDriver driver = new ChromeDriver();

-- Create a reference.

way 1 -
JavascriptExecutor js = driver;

way 2 - by using type casting -
JavascriptExecutor js = <JavascriptExecutor> driver;

-- perform operation.

driver.get("https://www.geeksforgeeks.org/");
WebElement java = driver.findElement(By.xpath("//a[@class='link'][normalize-space()='Java']"));
js.executeScript("arguments[0].click();", java);
```

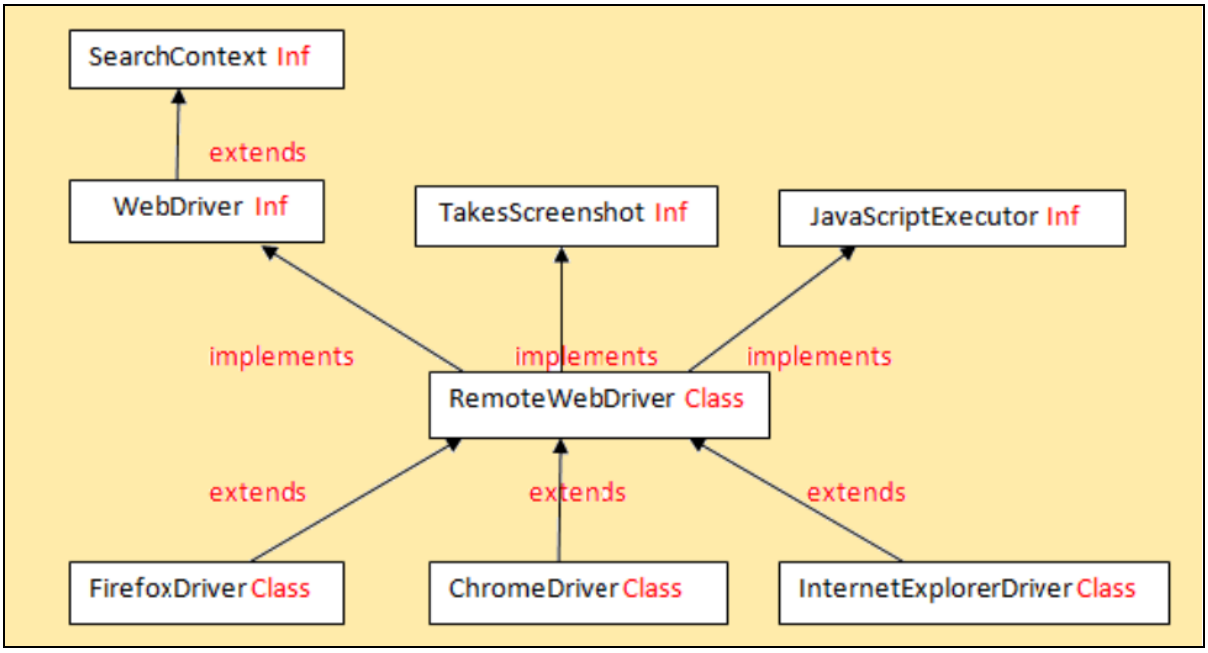
50. What is a Framework in Selenium?

A framework in Selenium is a structured approach for writing and managing test scripts efficiently. Types of Selenium Frameworks -

1. Data-Driven Framework - Test data is stored in external files (Excel, CSV, JSON, etc.). Uses Apache POI or libraries like Jackson for data handling.
2. Hybrid Framework (Combination of Data-Driven & Keyword-Driven) - Uses both test data from external sources and keywords for execution.
3. Page Object Model (POM) - Each web page is represented as a Java class with locators and methods. Improves code reusability and maintainability.
4. Behavior-Driven Development (BDD) – Cucumber Framework : Uses Gherkin language (Given-When-Then).

Why Use a Framework?

1. Code Reusability – Avoids duplication.
2. Scalability – Easy to add more test cases.
3. Maintainability – Centralized object repository and reporting.
4. Integration – Works with Jenkins, TestNG, and CI/CD pipelines.


SearchContext Inf
extends
WebDriver Inf
TakesScreenshot Inf
JavaScriptExecutor Inf
implements
implements
implements
RemoteWebDriver Class
extends
extends
extends
FirefoxDriver Class
ChromeDriver Class
InternetExplorerDriver Class