WebDriver Interface:

1. **WebDriver is interface or class?**
   - WebDriver is an interface in Selenium.
2. **Why do we upcast driver to WebDriver Interface?**
   - Upcasting to WebDriver interface allows you to change the browser driver (e.g., ChromeDriver, FirefoxDriver) without changing the code that interacts with the driver, thus making your code more flexible and maintainable.
3. **If we do not add driver exe file what will happen and what kind of exception will be generated?**
   - If the driver executable is not added, Selenium won't be able to communicate with the browser, and a `WebDriverException` will be thrown.
4. **Difference between `close` and `quit`?**
   - `close()`: Closes the current browser window.
   - `quit()`: Closes all the browser windows and ends the WebDriver session.
5. **Difference between `get` and `navigate().to()`?**
   - `get()`: Loads a new web page in the current browser window.
   - `navigate().to()`: Does the same as `get()` but allows for additional navigation options like back, forward, and refresh.
6. **Difference between `findElement` and `findElements`?**
   - `findElement`: Returns a single WebElement or throws `NoSuchElementException` if not found.
   - `findElements`: Returns a list of WebElements. If no elements are found, it returns an empty list.
7. **How to get all the links in the current page? Which locator will you use other than XPath?**
   - You can use the CSS selector `a` to find all links.

     ```
     List<WebElement> links =
     driver.findElements(By.cssSelector("a"));
     ```

8. **Methods of WebDriver?**
   - Some common methods are: `get()`, `getCurrentUrl()`, `getTitle()`, `findElement()`, `findElements()`, `getPageSource()`, `close()`, `quit()`, `navigate()`, `manage()`.
9. **Who is the parent of WebDriver?**
   - `SearchContext` is the parent interface of `WebDriver`.
10. **How do you download/install WebDriver? Explain the steps.**
    - Download the WebDriver executable from the official site of the respective browser (e.g., ChromeDriver for Chrome).
    - Extract the executable.
    - Set the path to the WebDriver executable in your code:

      ```
      System.setProperty("webdriver.chrome.driver",
      "path/to/chromedriver");
      ```

11. **Explain Selenium-WebDriver architecture.**

- Selenium-WebDriver architecture consists of four components: Selenium Client Library, JSON Wire Protocol over HTTP, Browser Drivers, and Real Browsers. The client library sends commands to the browser driver via JSON Wire Protocol. The driver executes commands in the actual browser.

12. **Explain Selenium-Java client architecture.**
    - The Selenium-Java client library communicates with browser-specific drivers using JSON Wire Protocol. These drivers then control the browser instances.
13. **Difference between `getWindowHandle` and `getWindowHandles`?**
    - `getWindowHandle()`: Returns a string representing the current window handle.
    - `getWindowHandles()`: Returns a set of strings representing all open window handles.
14. **How do you pass URI into browser?**
    - Use the `get()` method to pass a URI:

      ```
      driver.get("http://example.com");
      ```
15. **What is the use of `getCurrentPageSource` method?**
    - It returns the source code of the current page.
16. **Difference between Selenium-IDE, Selenium RC, WebDriver?**
    - Selenium IDE: A record-and-playback tool.
    - Selenium RC: A server-based tool to run Selenium scripts.
    - WebDriver: A tool to directly interact with web browsers without needing a server.
17. **How do you get the URL of the current web page?**
    - Use the `getCurrentUrl()` method:

      ```
      String currentURL = driver.getCurrentUrl();
      ```
18. **Difference between `WebDriver driver = new FirefoxDriver()` and `FirefoxDriver driver = new FirefoxDriver()`?**
    - `WebDriver driver = new FirefoxDriver()`: Polymorphism, allows switching drivers easily.
    - `FirefoxDriver driver = new FirefoxDriver()`: Tightly coupled with FirefoxDriver.
19. **Do we use any constructor in WebDriver?**
    - WebDriver is an interface, so it doesn't have constructors. Browser-specific classes like `ChromeDriver` have constructors.
20. **Assume browser version is different and exe file version is different, what kind of exception will you get?**
    - You will likely get a `SessionNotCreatedException`.
21. **How do you maximize and minimize the browser?**
    - Maximize:

      ```
      driver.manage().window().maximize();
      ```
    - Minimize:

```
driver.manage().window().minimize();
```

22. **Find title of the page without using `getTitle` method?**
    - You can use JavaScript:

    ```
    JavascriptExecutor js = (JavascriptExecutor) driver;
    String title = js.executeScript("return
    document.title;").toString();
    ```

23. **How do you capture a screenshot for a failed test script? Write syntax.**
    - Use `TakesScreenshot` interface:

    ```
    File screenshot = ((TakesScreenshot)
    driver).getScreenshotAs(OutputType.FILE);
    FileUtils.copyFile(screenshot, new
    File("path/to/save/screenshot.png"));
    ```

24. **When do we go for `findElements` method and what is the return type?**
    - Use `findElements` when you expect multiple elements. It returns a list of WebElements.
25. **What is IllegalStateException? When do we get it?**
    - `IllegalStateException` is thrown when a method has been invoked at an illegal or inappropriate time. In Selenium, it can occur if the WebDriver executable is not set correctly.
26. **Why do we get WebDriverException?**
    - This exception is thrown when WebDriver is unable to interact with the browser. Possible reasons include incorrect WebDriver setup, browser crashes, or network issues.
27. **Does WebDriver have any constructor?**
    - No, WebDriver is an interface and does not have a constructor.
28. **Disadvantages of WebDriver?**
    - Does not support Windows-based application testing.
    - Limited support for mobile testing (use Appium for that).
    - Requires programming knowledge.
29. **We have 4 tabs in the browser currently I am in parent tab how do you switch into 4th tab?**
    - Use `getWindowHandles()` to get a list of window handles and switch:

    ```
    Set<String> handles = driver.getWindowHandles();
    String[] handlesArray = handles.toArray(new String[0]);
    driver.switchTo().window(handlesArray[3]);
    ```

30. **What is WebElement? Is it a class or interface or abstract class?**
    - WebElement is an interface in Selenium.
31. **Methods of WebElement?**
    - Some common methods are: `click()`, `submit()`, `sendKeys()`, `getText()`, `getAttribute()`, `getCssValue()`, `isDisplayed()`, `isEnabled()`, `isSelected()`, `getTagName()`, `getSize()`, `getLocation()`.

32. **Why do we have `findElement` and `findElements` methods in WebElement?**
    - To locate child elements within a parent element.
33. **Difference between `click` and `submit()`?**
    - `click()`: Clicks on an element.
    - `submit()`: Submits a form.
34. **What is the use of `clear()` method?**
    - Clears the text in a text input or textarea element.
35. **How will you get text from WebElement?**
    - Use the `getText()` method:

```
String text = element.getText();
```

36. **How do you get height and width of the WebElement?**
    - Use the `getSize()` method:

```
Dimension size = element.getSize();
int height = size.getHeight();
int width = size.getWidth();
```

37. **How will you get the location of the WebElement?**
    - Use the `getLocation()` method:

```
Point location = element.getLocation();
```

38. **How will you get the x- and y-coordinates of WebElement?**
    - Use the `getLocation()` method and extract x and y:

```
Point location = element.getLocation();
int x = location.getX();
int y = location.getY();
```

39. **Purpose of using `getRect()` method?**
    - `getRect()` provides the location and size of the element in a single method.
40. **How do you send data into a text field?**
    - Use the `sendKeys()` method:

```
element.sendKeys("text");
```

41. **How do you pass data into a text field without using `sendKeys()`?**
    - Use JavaScript:

```
JavascriptExecutor js = (JavascriptExecutor) driver;
js.executeScript("arguments[0].value='text';", element);
```

42. **How do you click on the WebElement without using `click()` method?**
    - Use JavaScript:

```
JavascriptExecutor js = (JavascriptExecutor) driver;
```

```
js.executeScript("arguments[0].click();", element);
```

43. **How will you check whether the element is displayed or not?**
    - Use the `isDisplayed()` method:

    ```
    boolean isDisplayed = element.isDisplayed();
    ```

44. **How will you check whether the element is enabled or not?**
    - Use the `isEnabled()` method:

    ```
    boolean isEnabled = element.isEnabled();
    ```

45. **How will you check whether the element is selected or not?**
    - Use the `isSelected()` method:

    ```
    boolean isSelected = element.isSelected();
    ```

46. **How do you get all the links of the current web page?**
    - Use the CSS selector `a`:

    ```
    List<WebElement> links =
    driver.findElements(By.cssSelector("a"));
    ```

47. **What is the return type of `findElements()` method?**
    - It returns a list of WebElements (`List<WebElement>`).
48. **Why do we have `findElement` and `findElements` methods on both WebDriver and WebElement?**
    - To allow locating elements both at the document level (using WebDriver) and within a specific parent element (using WebElement).
49. **How to capture color, height, width, font-size of the element?**
    - Use `getCssValue()`:

    ```
    String color = element.getCssValue("color");
    String height = element.getCssValue("height");
    String width = element.getCssValue("width");
    String fontSize = element.getCssValue("font-size");
    ```

50. **How to delete all cookies?**
    - Use `deleteAllCookies()` method:

    ```
    driver.manage().deleteAllCookies();
    ```

51. **How will you handle disabled elements?**
    - Disabled elements cannot be interacted with directly. You may use JavaScript to interact with them:

    ```
    JavascriptExecutor js = (JavascriptExecutor) driver;
    ```

```
js.executeScript("arguments[0].removeAttribute('disabled');",
element);
```

52. **How will you handle scroll up and scroll down webpage?**
    - Use JavaScript:

    ```
    // Scroll down
    js.executeScript("window.scrollBy(0,1000)");
    // Scroll up
    js.executeScript("window.scrollBy(0,-1000)");
    ```

53. **How do you capture the screenshot of a webpage?**
    - Use `TakesScreenshot` interface:

    ```
    File screenshot = ((TakesScreenshot)
    driver).getScreenshotAs(OutputType.FILE);
    FileUtils.copyFile(screenshot, new
    File("path/to/save/screenshot.png"));
    ```

Synchronization:

54. **What is Synchronization?**
    - Synchronization refers to handling the timing issues when there are delays in loading elements or pages, ensuring that Selenium waits for elements to be available before performing actions on them.
55. **How to handle Synchronization wait available in WebDriver?**
    - Using implicit waits, explicit waits, and fluent waits.
56. **Which wait statement will be used to wait until the page gets loaded?**
    - You can use `WebDriverWait` with a condition like `wait.until(ExpectedConditions.visibilityOfElementLocated(By.id("someId")));`.
57. **How to handle dynamic object?**
    - Use explicit waits to wait for the object to appear, and use flexible locators like XPath or CSS selectors that can handle dynamic attributes.
58. **Difference between thread wait, implicitly wait, explicitly wait?**
    - Thread wait (`Thread.sleep()`): Pauses the execution for a specified time.
    - Implicit wait: Sets a default wait time for the WebDriver to wait for elements to appear.
    - Explicit wait: Waits for a specific condition to occur before proceeding.
59. **Do you have any idea of fluent wait? If yes, explain.**
    - Fluent wait is a type of explicit wait with polling intervals. It repeatedly checks for a condition at specified intervals, ignoring exceptions.
60. **If an element fails to load within implicit wait, what exception will you get?**
    - `NoSuchElementException`.

Locators:

61. **Locators in Selenium?**

  - ID, Name, Class Name, Tag Name, Link Text, Partial Link Text, CSS Selector, XPath.

62. **When do we go for CSS selector?**
    - When you need a fast and flexible locator that can handle complex CSS attribute matching.
63. **When do we go for ID?**
    - When the element has a unique ID attribute, which is the fastest and most reliable locator.
64. **When do we go for name?**
    - When elements have a unique name attribute.
65. **When do we go for linkText and partialLinkText locator?**
    - When locating links by their visible text.
66. **When do we go for XPath?**
    - When other locators are not suitable, especially for complex and dynamic elements.
67. **Difference between CSS selector and XPath?**
    - CSS selector is generally faster and more readable. XPath is more powerful and flexible, especially for complex element hierarchies.
68. **Advantages of XPath?**
    - Can navigate through elements and attributes in a document, supports complex queries, and can locate elements by text.
69. **Absolute and Relative XPath?**
    - Absolute XPath: Starts from the root and follows a complete path (e.g., `/html/body/div`).
    - Relative XPath: Starts from the middle of the HTML DOM structure (e.g., `//div[@id='example']`).
70. **How will you handle if the WebElement is partially dynamic?**
    - Use partial matching techniques in locators like `contains()` in XPath or `*=` in CSS.
71. **How to write XPath in IE & Chrome browser?**
    - You can use the browser's developer tools to inspect elements and generate XPath.
72. **What do you know about XPath by axes? Explain.**
    - XPath axes refer to the relationship between nodes in the document. Common axes include `parent`, `child`, `ancestor`, `descendant`, `following-sibling`, and `preceding-sibling`.
73. **What is your approach to find the location of WebElement if we do not have any attribute in the tag?**
    - Use the element's position relative to other elements, or use text-based XPath.
74. **How will you handle if the locator value is matching with multiple elements?**
    - Use `findElements` and work with the list of elements returned.
75. **Will locator value of the WebElement change from one browser to another? Justify.**
    - Locator values generally remain consistent across browsers, but rendering differences might cause certain elements to be present or absent.

Pop-Ups:

76. **How do you identify whether it is an alert pop-up or not? How do you handle it?**
    - Selenium automatically switches to alert pop-ups. Use `driver.switchTo().alert()` to handle alerts.
77. **Can I inspect Alert pop-up?**
    - No, JavaScript alerts cannot be inspected.
78. **Is Alert an interface or class? Explain the methods of it.**
    - Alert is an interface. Methods include `accept()`, `dismiss()`, `getText()`, and `sendKeys()`.
79. **How do you identify whether it is a hidden-division pop-up or not? How do you handle it?**
    - Hidden-division pop-ups are regular DOM elements. Use WebDriver methods to interact with them.
80. **How do you identify whether it is a notification pop-up or not? How do you handle it?**
    - Notification pop-ups are handled by the browser. You can use WebDriver to interact with the notification permissions.
81. **How do you handle authentication pop-up?**
    - Pass credentials in the URL:

      ```
      driver.get("http://username:password@url.com");
      ```
82. **How do you handle file upload pop-up?**
    - Use `sendKeys()` to input the file path into the file input element.
83. **If alert pop has text field how will you enter the value to the field?**
    - Use `sendKeys()` method on the alert:

      ```
      driver.switchTo().alert().sendKeys("text");
      ```
84. **Can I inspect file upload pop-up?**
    - No, OS-level file upload pop-ups cannot be inspected.
85. **Can I handle file upload pop-up with the help of Robot class? If yes, explain.**
    - Yes, Robot class can simulate keyboard and mouse actions to handle file upload pop-ups.
86. **How do you handle file download pop-up?**
    - Configure browser settings to auto-download files, or use AutoIT/Robot class.
87. **How will you handle SSL pop-up in IE?**
    - Use desired capabilities to accept SSL certificates:

      ```
      DesiredCapabilities capabilities =
      DesiredCapabilities.internetExplorer();
      capabilities.setCapability(CapabilityType.ACCEPT_SSL_CERTS,
      true);
      ```
88. **How to handle File Upload Pop-up using AutoIT?**
    - Write an AutoIT script to handle the pop-up and execute it using:

      ```
      Runtime.getRuntime().exec("path/to/autoit/script.exe");
      ```

89. **How do you get text from alert pop-up?**
    - Use `getText()` method on the alert:

      ```
      String alertText = driver.switchTo().alert().getText();
      ```

Drop Down:

90. **How will you identify if a dropdown is single select or multi select?**
    - Use `isMultiple()` method of the `Select` class:

      ```
      Select select = new Select(element);
      boolean isMultiple = select.isMultiple();
      ```

91. **How do you handle dropdown?**
    - Use the `Select` class:

      ```
      Select select = new Select(element);
      select.selectByVisibleText("Option");
      ```

92. **WebDriver driver = new ChromeDriver(); Select select = new Select(driver); True or false? Justify.**
    - False. `Select` constructor requires a WebElement, not a WebDriver instance.
93. **How do you handle if the dropdown options are dynamic?**
    - Use explicit waits to wait for options to be populated.
94. **How will you handle dropdown without using Select class?**
    - Click on the dropdown and then click on the desired option using WebDriver methods.
95. **How will you get all the options from dropdown?**
    - Use `getOptions()` method of the `Select` class:

      ```
      List<WebElement> options = select.getOptions();
      ```

96. **How do you select only alternative options from dropdown?**
    - Loop through the options and select every alternate one using `Select` methods.
97. **How will you get only selected options from dropdown?**
    - Use `getAllSelectedOptions()` method of the `Select` class.
98. **Get all even index options from dropdown?**
    - Loop through the options list and select options with even indices.
99. **How will you get the first selected option from dropdown?**
    - Use `getFirstSelectedOption()` method of the `Select` class.
100. **How will you check if the content in dropdown is not duplicate?** - Store options in a Set and compare its size with the original list.
101. **Why do we use `getWrappedElement()` method?** - To get the WebElement associated with the `Select` object.
102. **What is the return type of `getWrappedElement()` method?** - It returns a WebElement.

103. **What is the return type of `selectByVisibleText()`, `selectByIndex()`, `selectByValue()` method?** - These methods have a void return type.
104. **What is the return type of `getAllSelectedOptions()` method?** - It returns a List of WebElements.
105. **How do you de-select the selected option from the dropdown?** - Use `deselectByVisibleText()`, `deselectByIndex()`, or `deselectByValue()` methods of the `Select` class.
106. **How do you de-select all the selected options from the dropdown?** - Use `deselectAll()` method of the `Select` class.
107. **We have two dropdowns in a web page. How will you compare all the options of one dropdown with all the options of another dropdown?** - Get the options of both dropdowns using `getOptions()` method and compare their text values.

Actions Class:

109. **How to perform double click on the WebElement?** - Use `Actions` class:

```
Actions actions = new Actions(driver);
actions.doubleClick(element).perform();
```

110. **How do you right-click on the webpage?** - Use `Actions` class:

```
Actions actions = new Actions(driver);
actions.contextClick(element).perform();
```

111. **How will you perform drag and drop action?** - Use `Actions` class:

```
Actions actions = new Actions(driver);
actions.dragAndDrop(sourceElement, targetElement).perform();
```

112. **How do you move the cursor to a specific WebElement?** - Use `Actions` class:

```
Actions actions = new Actions(driver);
actions.moveToElement(element).perform();
```

113. **Can I perform keyboard actions with the help of Actions class?** - Yes, use methods like `sendKeys()` in the `Actions` class.
114. **What is the difference between `perform()` and `build()`?** - `build()`: Compiles the actions into a single step. - `perform()`: Executes the actions.
115. **What happens if I do not call `perform()` method?** - The actions will not be executed.
116. **Write a syntax to double-click on WebElement using Actions class.** Actions actions = new Actions(driver); actions.doubleClick(element).perform();
117. **How will you perform keyboard action? Write syntax.** - Actions actions = new Actions(driver); actions.sendKeys(Keys.ENTER).perform();

Frames:

118. **How to work with frame-window?** - Use `switchTo().frame()` method:

```
driver.switchTo().frame(frameElement);
```

119. **How to work with nested frames?** - Switch to the outer frame first, then to the inner frame.
120. **How to work with multiple frames of a web page?** - Use `switchTo().frame()` with different frame references.
121. **How many ways to work with frames?** - By index, by name or ID, and by WebElement.
122. **How to work with frame, when frame does not have id & @name attribute?** - Use WebElement reference.
123. **How do you identify frame in a web page?** - Inspect the page to find the `<iframe>` or `<frame>` tags.
124. **What is the action of `defaultContent()`?** - Switches the context to the main document from a frame.
125. **What is the action of `parentFrame()`?** - Switches to the parent frame of the current frame.

Data-driven:

126. **What is data-driven framework?** - A framework where test data is driven from external data sources like Excel, CSV, or databases.
127. **Why do we go for Excel?** - Excel is user-friendly, supports complex data structures, and is easy to manipulate.
128. **What is POI?** - Apache POI is a Java API for manipulating Microsoft documents like Excel.
129. **How do you get the data from Excel sheet? Write syntax.** - FileInputStream fis = new FileInputStream("path/to/excel"); Workbook workbook = WorkbookFactory.create(fis); Sheet sheet = workbook.getSheetAt(0); Row row = sheet.getRow(0); Cell cell = row.getCell(0); String data = cell.getStringCellValue();

130. **How do you set the data into Excel sheet? Explain with syntax.** - FileOutputStream fos = new FileOutputStream("path/to/excel"); Cell cell = row.createCell(1); cell.setCellValue("data"); workbook.write(fos);

131. **Is WorkBookFactory an interface or class?** - It is a class.
132. **Why do we call `create()` method?** - To create a Workbook instance from a file.
133. **What is the function of `FileInputStream` object in data-driven?** - It reads data from a file.
134. **What is the function of `FileOutputStream` object in data-driven?** - It writes data to a file.
135. **What is the return type of `create()` method?** - It returns a `Workbook` object.
136. **What is the return type of `getSheet()`?** - It returns a `Sheet` object.
137. **What is the return type of `getRow()`?** - It returns a `Row` object.
138. **What is the return type of `getCell()`?** - It returns a `Cell` object.

139. **Difference between `getSheet()` and `createSheet()` methods?** - `getSheet()`: Retrieves an existing sheet. - `createSheet()`: Creates a new sheet.
140. **Difference between `getStringCellValue()` and `setStringCellValue()`?** - `getStringCellValue()`: Retrieves the value from a cell. - `setStringCellValue()`: Sets a value in a cell.
141. **How do you set multiple data into Excel sheet dynamically?** - Use loops to iterate through rows and cells, setting data dynamically.
142. **How do you get data from all the columns of the last row in an Excel sheet?** - Use `getLastRowNum()` to find the last row and then iterate through columns.
143. **What are the jars/dependencies we need to add to get/set the data from Excel sheet?** - Apache POI dependencies: `poi`, `poi-ooxml`, `poi-ooxml-schemas`, `xmlbeans`.

TestNG:

144. **What is TestNG?** - TestNG is a testing framework inspired by JUnit and NUnit, designed for testing needs in Java.
145. **Why do we go for TestNG?** - It provides powerful features like annotations, parallel execution, and test configuration.
146. **Why do we call TestNG as unit testing tool?** - Because it allows writing and running unit tests.
147. **What are the ways we have to install TestNG?** - Install as a plugin in Eclipse or add it as a dependency in Maven/Gradle.
148. **How do you install TestNG? Explain with steps.** - In Eclipse: Go to Help > Eclipse Marketplace > Search for TestNG > Install. - Maven: Add the TestNG dependency to `pom.xml`.
149. **When do we get `TestNGException`? Why?** - When there are issues with TestNG configuration or when tests fail to execute properly.
150. **Uses of TestNG?** - Organizing tests, setting test priorities, data-driven testing, generating reports, and parallel execution.
151. **Is TestNG an interface or class?** - TestNG is a class.
152. **Explain the annotations of TestNG?** - Common annotations include `@Test`, `@BeforeMethod`, `@AfterMethod`, `@BeforeClass`, `@AfterClass`, `@BeforeSuite`, `@AfterSuite`.
153. **Explain the order of execution of all the annotations?** - `@BeforeSuite`, `@BeforeTest`, `@BeforeClass`, `@BeforeMethod`, `@Test`, `@AfterMethod`, `@AfterClass`, `@AfterTest`, `@AfterSuite`.
154. **How do you perform group execution using TestNG?** - Use `groups` attribute in `@Test` and configure groups in `testng.xml`.
155. **How do you perform batch execution using TestNG?** - Define tests in `testng.xml` and execute them together.
156. **How do you perform controlled batch execution using TestNG?** - Use `dependsOnMethods` and `dependsOnGroups` attributes to control test execution order.
157. **What do you mean by parallel execution?** - Running multiple tests concurrently to save time.
158. **What do you mean by thread-count?** - Specifies the number of threads to be used for parallel execution.

159. **How do you perform parallel execution using TestNG?** - Set `parallel` and `thread-count` attributes in `testng.xml`.
160. **I have 200 test cases, what is your approach if I want to execute them parallelly in two different browsers?** - Define two `<test>` tags in `testng.xml`, each specifying a different browser.
161. **I have 200 test cases, what is your approach if I want to execute them parallelly in the same browser?** - Use the `parallel="methods"` attribute in `testng.xml`.
162. **How do you run test cases at the class level?** - Use `@Test` annotation at the class level.
163. **How do you run test cases at the package level?** - Define the package in `testng.xml`.
164. **How do you run all the test cases at the project level?** - Use the `testng.xml` file at the project root or use Maven/Gradle build tools.
165. **How do you run a single test case?** - Use `testng.xml` to define a single test or run it directly from the IDE.
166. **Why do we go for `@DataProvider` annotation?** - For parameterizing tests and providing multiple sets of data to a test method.
167. **How do you perform cross-browser testing?** - Use WebDriver capabilities and configure browsers in `testng.xml`.
168. **How do you run the same test case 100 times?** - Use `invocationCount` attribute in the `@Test` annotation:

```
@Test(invocationCount = 100)
```

169. **How will you create dependency between two test cases?** - Use `dependsOnMethods` attribute in the `@Test` annotation.
170. **How do you skip a test case?** - Use `enabled=false` attribute in the `@Test` annotation:

```
@Test(enabled = false)
```

171. **Is it possible to skip a test case without using `enabled`?** - Yes, use `throw new SkipException("Skipping test")` in the test method.
172. **How do you prioritize the test cases?** - Use `priority` attribute in the `@Test` annotation:

```
@Test(priority = 1)
```

173. **Why do we need to prioritize the test cases?** - To define the order of test execution.
174. **How do you convert all the test cases into `testNG.xml` file?** - Use the TestNG Eclipse plugin to generate `testng.xml` or manually create the file.
175. **What is the use of `testNG.xml` file?** - It is used to configure and organize the execution of test cases.

176. **Can I create multiple `testNG.xml` files?** - Yes, you can create multiple `testNG.xml` files for different test suites.
177. **Can I create multiple `testNG.xml` files with the same name?** - No, file names must be unique within the same directory.
178. **Can I run `testNG.xml` file?** - Yes, you can run it directly from the IDE or through the command line.
179. **Can I run multiple `.xml` files at the same time?** - Yes, you can specify multiple XML files to be run together.
180. **What is the advantage of TestNG report?** - It provides detailed HTML reports of test execution with logs and results.
181. **How do you execute only failed test cases?** - Use `testng-failed.xml` generated by TestNG after a test run.
182. **What do you know about include and exclude?** - Used in `testng.xml` to include or exclude specific test methods or groups.
183. **How do you configure `testng.xml` file to perform parallel execution?** - Set the `parallel` attribute in `testng.xml` to `tests`, `classes`, or `methods`.
184. **Can I run `testng.xml` file through command prompt?** - Yes, use the command:

```
java -cp <classpath> org.testng.TestNG testng.xml
```

185. **Why do we use `log()` method?** - To log messages during test execution.
186. **I will run 100 test cases but 10 got failed. How do you run only those failed test cases?** - Use `testng-failed.xml` generated after the initial run.
187. **Advantages of TestNG over JUnit?** - More annotations, parallel execution, flexible test configuration, detailed reports, and data-driven testing support.

POM:

188. **What is POM?** - Page Object Model (POM) is a design pattern to create object repositories for web UI elements.
189. **Why do we go for POM?** - To enhance code reusability, maintainability, and readability by encapsulating page elements and actions.
190. **Explain the rule of POM class design?** - Each page of the application should have a corresponding class. This class should contain WebElements and methods to interact with those elements.
191. **Why/when do we get `StaleElementReferenceException`?** - When a WebElement is no longer attached to the DOM.
192. **How do you handle `StaleElementReferenceException`?** - Re-locate the element, use retry logic, or wait until the DOM is stable.
193. **Is it possible to handle `StaleElementReferenceException` without POM class?** - Yes, by implementing retry logic or re-locating elements.
194. **Difference between `findBy` and `findElement`?** - `findBy` is used in POM to define locators; `findElement` is used to locate elements in WebDriver.

195. **Difference between `findBy`, `findBys` and `findAll`?** - `findBy`: Finds a single element. - `findBys`: Finds elements matching all specified criteria. - `findAll`: Finds elements matching any of the specified criteria.
196. **Why do we need to follow encapsulation rule to design POM class?** - To ensure that the WebElements are accessed only through methods in the POM class, enhancing maintainability and readability.
197. **Why do we need to call `initElements()` method only inside the constructor?** - To initialize the elements as soon as the object of the POM class is created.

General:

198. **Why do we go for automation?** - To save time, reduce human errors, increase test coverage, and perform repetitive tasks efficiently.
199. **What is automation testing?** - Automation testing involves using tools and scripts to perform tests on software applications automatically.
200. **Can I automate hardware-related test cases?** - Generally, no. Automation testing focuses on software applications.
201. **What is the prerequisite for automation?** - A stable application, test cases, automation tools, and a testing environment.
202. **What is not automatable?** - Highly dynamic UI changes, complex human interactions, and non-deterministic tasks.
203. **Advantages and disadvantages of Selenium IDE?** - Advantages: Easy to use, record-and-playback, no programming required. - Disadvantages: Limited functionality, not suitable for complex test cases, no support for programming logic.
204. **Advantages and disadvantages of Selenium RC?** - Advantages: Supported multiple browsers and platforms, allowed programming logic. - Disadvantages: Requires server to start, slower than WebDriver, outdated.
205. **Any idea of GitHub?** - GitHub is a web-based platform for version control and collaboration using Git. It allows multiple developers to work on projects simultaneously, track changes, and manage repositories.
206. **Any idea of Maven?** - Maven is a build automation tool used primarily for Java projects. It helps manage project dependencies, build processes, and project documentation.
207. **Any idea of Jenkins?** - Jenkins is an open-source automation server that helps in continuous integration and continuous delivery (CI/CD) of software projects. It automates the building, testing, and deployment of applications.