**||QA Assessment Exam||**

**||ABEDALRAHEEM ALSAQQA||**

Q1- What is the advantage of using webdrivermanager package in selenium?

Q2- Name the most common methods of Rest API?

Q3- Name the wait methods in selenium?

Q4- What is the difference between Break and continue in Java?

Q5- Write two statement that allows class A to give its own properties to other class which is B?

Q6- Name three different maven repositories that we use commonly?

Q7- Call the methods to declare a variable in postman?

Q8 name three testing frameworks that you know ?

Q9 why we use the invocation in testng?

Q11 What is selenium

Q12 what is the different between selenium java and selenium python

Q13 how to locate elements in selenium.& what is the fastest way ?

Q14 . how manage window size in selenium?

Q15 what to do if the item can’t be shown directly

Q16 what is the different between thread.sleep and implicit wait

Q17 what is the different between class and id

Q18 can we locate the item by using more than way of locating , justify your answer ?

Q19 most important regular expression why we use it

Q20 what is the best way to do the same operation over and over again

Q21 what is the output here ( 21 Q and 22st Q )

for(int i = 30 ; i >10 ; i--) {

System.out.println(i);

break;

}

Q22 is there any different here ?

for(int i = 30 ; i >10 ; i++) {

if (i == 30) {

System.out.println(i + " equal to 30 ");

}

continue ;

System.out.println(i)

}

Q23 what is the error below

int arr2[] = {8,6,10,17,20};

System.out.println(arr1[1]);

System.out.println(arr2[2]);

System.out.println(arr3[3]);

System.out.println(arr4[4]);

System.out.println(arr5[5]);

System.out.println(arr6[6]);

Q24 for Q23 after you find the problem what is the suggested solution ?

Q25what is the correct sequence of post man starting from the request till the getting the response ?

Q26 what is the different between test and pre test in post man ?

**Answers**

**Automated Driver Management: WebDriverManager automates the process of downloading and managing WebDriver executables for different browsers. It can automatically download the appropriate version of the WebDriver executable based on your browser version and operating system, ensuring compatibility.**

**Simplified Setup: It simplifies the Selenium test setup by removing the need to manually download and configure WebDriver executables. This can save time and effort, especially when working with different browsers and versions.**

**Reduced Maintenance: WebDriverManager can automatically update WebDriver binaries when new versions are released, reducing the maintenance overhead of your Selenium tests. This ensures that your tests continue to work with the latest browser versions.**

**Cross-Platform Compatibility: WebDriverManager is compatible with various operating systems (Windows, macOS, Linux) and can be used with different Selenium WebDriver clients (e.g., Java, Python, C#).**

**Enhanced Readability: Your test code becomes cleaner and more readable because you don't need to specify the exact path to WebDriver executables. WebDriverManager takes care of locating and configuring the correct binary for your chosen browser.**

**Easy Integration: It can be easily integrated into popular testing frameworks like TestNG or JUnit.**



**GET: This method is used to retrieve data from the server. It should not have any side effects on the server; it's meant for read-only operations. When you make a GET request to a resource's URL, you are requesting information about that resource.**

**POST: POST is used to create a new resource on the server. It submits data to be processed to a specified resource. It can also be used for various other operations, but it's commonly associated with creating new resources.**

**PUT: PUT is used to update an existing resource on the server. It typically requires the client to send the entire updated representation of the resource. If the resource doesn't exist, PUT can create it. It is idempotent, meaning multiple requests will have the same effect as a single request.**

**PATCH: PATCH is used to partially update an existing resource. Instead of sending the entire resource, the client can send only the changes (differences) to the server, reducing the amount of data transferred. Like PUT, it's also idempotent.**

**DELETE: DELETE is used to remove a resource from the server. It performs the deletion of the specified resource. It should be used with caution, as it permanently removes the resource from the server.**



**Implicit Wait: Implicit waits are used to set a default waiting time for the entire script. If an element is not immediately available, Selenium will wait for a specified amount of time before throwing an exception. It's applied globally to all elements and actions in the script.**

**Explicit Wait: Explicit waits are more specific and flexible than implicit waits. They allow you to wait for a particular condition to be met for a specific element. You can specify the maximum time to wait and the condition to wait for, such as the presence of an element, its visibility, or some custom condition. Common classes used for explicit waits are WebDriverWait and ExpectedConditions.**

**Fluent Wait: Fluent waits are an extension of explicit waits. They allow you to specify the polling interval and ignore certain exceptions. This provides greater control over how long and how often Selenium checks for a condition.**

**Thread.sleep(): While not a wait method provided by Selenium, Thread.sleep() is a general-purpose Java method that can be used to introduce pauses in your script. However, it's less efficient and less flexible than explicit waits, as it introduces fixed, often unnecessary delays.**



**Break:**

**break is used to exit a loop prematurely. When the break statement is encountered within a loop, the loop is terminated, and the program continues with the statement immediately following the loop.**

**It is often used to exit a loop when a certain condition is met, allowing you to terminate the loop early.**

**Continue:**

**continue is used to skip the current iteration of a loop and proceed to the next iteration. When the continue statement is encountered, the remaining code within the current iteration is skipped, and the loop proceeds with the next iteration.**

**It is often used when you want to skip a specific iteration of the loop based on a condition, but you don't want to exit the loop entirely.**



**class A {**

**int propertyA; }**

**class B {**

**A objectA = new A();**

**int propertyB; }**



**Maven Central Repository**

**JCenter Repository**

**The availability and usage of repositories may change over time, so it's important to verify the current status and repositories specific to your project's dependencies. Additionally, organizations often use private or custom repositories for their internal artifacts and libraries.**



**Using pm.environment for Environment Variables:**

**To declare an environment variable, you can use pm.environment.set() method.**

**Here's an example of how to declare and set an environment variable named "baseUrl" with a value:**

**pm.environment.set("baseUrl", "https://api.example.com");**

**Using pm.globals for Global Variables:**

**To declare a global variable, you can use pm.globals.set() method.**

**Here's an example of how to declare and set a global variable named "authToken" with a value:**

**pm.globals.set("authToken", "your\_token\_value");**

**Using pm.collectionVariables for Collection Variables:**

**To declare a collection variable, you can use pm.collectionVariables.set() method.**

**Here's an example of how to declare and set a collection variable named "apiKey" with a value:**

**pm.collectionVariables.set("apiKey", "your\_api\_key");**



**TestNG: TestNG is a popular testing framework for Java that is widely used for unit testing, integration testing, and end-to-end testing. It provides features like parallel test execution, test prioritization, and test parameterization.**

**JUnit: JUnit is another widely used testing framework for Java. It is primarily focused on unit testing and is often used in test-driven development (TDD). JUnit 5, the latest version, introduced several new features and improvements.**

**Selenium WebDriver: While Selenium is primarily known as an automation tool for web testing, it also offers a testing framework to structure and manage test cases. It's particularly popular for web application testing.**



**Repeating Tests: It allows you to run the same test method multiple times with the same set of data. This can be useful for testing the stability and consistency of your application under different conditions.**

**Load Testing: It can be used for simulating load or stress testing by running the same test multiple times concurrently, which can help identify performance and concurrency issues.**

**Testing with Varied Data: When combined with data providers, you can run the same test with different data inputs, making it effective for parameterized testing and testing multiple scenarios**

**Q11: Selenium is an open-source framework and a suite of tools used for automating web browsers. It is primarily used for automating web application testing but can also be used for various other web-related tasks. Selenium allows you to interact with web elements, perform actions on web pages, and validate the behavior of web applications. It provides support for various programming languages, including Java, Python, C#, and more.**

**Q12: The main difference between Selenium with Java and Selenium with Python lies in the programming language used for test automation. Both versions of Selenium offer similar functionality, but they are written in different languages and provide different language-specific APIs. Some key differences include:**

**- Language: Selenium with Java uses the Java programming language for test scripting, while Selenium with Python uses Python.**

**- Syntax: The syntax and coding style differ between Java and Python, so your test scripts will be written in the chosen language.**

**- Ecosystem: Java has a larger ecosystem and more libraries available for various tasks, but Python is known for its simplicity and readability.**

**- Community and Documentation: Both Java and Python Selenium bindings have active communities and extensive documentation, but the way you access and use documentation differs based on the language.**

**- Personal Preference: The choice between Java and Python often comes down to personal preference and the specific needs of your project.**

**Q13: To locate elements in Selenium, you can use various locators, such as ID, name, class name, tag name, XPath, CSS selectors, and more. The fastest way to locate elements depends on the specific scenario and the web page you are interacting with. Generally, using IDs and CSS selectors are considered faster methods because they allow Selenium to locate elements efficiently. XPath can be slower due to its complexity, especially when using absolute XPath.**

**Here are examples of element locators:**

**- By ID:**

**driver.findElement(By.id("elementId"));**

**- By CSS Selector:**

**driver.findElement(By.cssSelector("#elementId"));**

**- By XPath:**

**driver.findElement(By.xpath("//input[@id='elementId']"));**

**The choice of locator depends on the uniqueness and stability of the element on the page.**

**Q14: To manage the window size in Selenium, you can use the `manage().window()` method from the `WebDriver` instance. Here's how to set the window size:**

**WebDriver driver = new ChromeDriver();**

**driver.manage().window().setSize(new Dimension(width, height));**

**Replace `width` and `height` with the desired width and height values in pixels. For example, to set the window size to 1200x800 pixels:**

**driver.manage().window().setSize(new Dimension(1200, 800));**

**This will set the browser window's size to the specified dimensions. You can adjust the size according to your testing needs.**

**Q15: If an item cannot be shown directly in a web page, you can consider using various techniques to locate it. These include:**

**- XPath:Using XPath expressions to navigate the HTML structure and locate elements, especially when there are no specific attributes.**

**- CSS Selectors: Utilizing CSS selectors that target specific element properties, classes, or attributes to find elements that cannot be located using other methods.**

**- Relative Locators: Tools like Selenium 4's Relative Locators can help you locate elements in relation to other elements (e.g., above, below, near).**

**- Implicit Wait: Configuring implicit waits, allowing the web driver to wait for a certain period for an element to appear. It's a good practice to use waits when elements are loaded asynchronously.**

**- Explicit Wait: Using explicit waits to wait for elements to meet specific conditions. Explicit waits provide more control and flexibility when dealing with dynamic web pages.**

**Q16: The main differences between `Thread.sleep` and implicit wait in Selenium are:**

**- Purpose: `Thread.sleep` is a general-purpose pause that adds a fixed delay, while an implicit wait is specifically designed for waiting for web elements to appear or become interactable.**

**- Efficiency: Implicit wait is more efficient because it waits only as long as necessary, whereas `Thread.sleep` introduces fixed and often unnecessary delays.**

**- Granularity: Implicit wait can be set at a fine-grained level, applying only to specific elements, while `Thread.sleep` applies globally and halts the entire test execution.**

**- Handling Conditions: Implicit wait is dynamic, checking for element presence at regular intervals until a timeout is reached, while `Thread.sleep` doesn't check conditions and blindly waits for the specified time.**

**- Wait Customization: With implicit wait, you can set different timeouts for different elements or conditions, making it more adaptable to dynamic page loading.**

**Q17: The difference between class and ID in HTML and CSS is as follows:**

**- Class:In HTML, the `class` attribute is used to apply a class to one or more elements. It allows multiple elements to share the same class and style. In CSS, you can target elements with a specific class using a dot notation (e.g., `.classname`).**

**- ID:In HTML, the `id` attribute is used to uniquely identify a single element on the page. Each element should have a unique ID. In CSS, you can target elements with a specific ID using a hash notation (e.g., `#elementid`).**

**Q18: Yes, you can locate an item using multiple ways of locating in Selenium. This approach is known as "fallback" or "chained" locating. It's a good practice for handling different scenarios or when the primary locator fails.**

**Here's an example of locating an element with a fallback strategy:**

**By primaryLocator = By.id("elementId");**

**By secondaryLocator = By.name("elementName");**

**WebElement element = null;**

**try {**

**element = driver.findElement(primaryLocator);**

**} catch (NoSuchElementException e) {**

**element = driver.findElement(secondaryLocator);**

**}**

**This approach improves the robustness of your tests, as it can adapt to changes in the web page structure and ensure that the test can proceed even if one locator fails.**

**Q19: Regular expressions (regex) are used in software testing and many other fields for various reasons. In testing, regular expressions are important because they allow you to:**

**- Match Patterns:You can use regex to define and match patterns within text or data. This is helpful for validating and extracting information from test results, log files, and web page content.**

**- Data Validation:Regular expressions enable you to validate user input and ensure it conforms to specific formats or constraints, such as email addresses, phone numbers, and passwords.**

**- Test Data Generation: You can use regex to generate test data or simulate specific patterns, which is valuable for testing various scenarios.**

**- Text Manipulation: Regular expressions are helpful for text manipulation tasks, such as replacing, searching, and formatting text in test scripts or test data.**

**- Dynamic Element Locators: When working with dynamic elements on web pages, you can use regex to create flexible element locators, especially when the attributes have varying values.**

**Q20: The best way to perform the same operation over and over again in software testing and automation is by using loops. In testing, you often need to execute the same set of actions or test steps multiple times, and loops provide a way to do this efficiently. Common loop constructs used in testing include `for` loops, `while` loops, and test case parameterization. These loops allow you to iterate through test data, perform repetitive tasks, and validate your application's behavior across different scenarios, reducing the need for redundant code.**

**Q21: The code you provided will only print the number 30 and then break out of the loop. Therefore, the output will be:**

**Output : 30**

**The loop iterates just once, printing 30, and then the `break` statement is encountered, causing the loop to exit.**

**Q22: In the second code snippet, it has a `continue` statement after printing "30 equal to 30." This `continue` statement means that the loop will continue to the next iteration without executing any of the code that follows it. Therefore, the loop will execute repeatedly but never print anything other than "30 equal to 30."**

**The output of the second code will be:**

**```**

**30 equal to 30**

**30 equal to 30**

**30 equal to 30**

**...**

**```**

**It will print "30 equal to 30" repeatedly in an infinite loop.**

**Q23:**

**- You are trying to print elements from arrays (`arr1`, `arr3`, `arr4`, `arr5`, and `arr6`) that have not been declared or initialized in your code.**

**- You have declared `arr2`, but you are attempting to print elements from `arr1`, which doesn't exist in your code.**

**- You are trying to access elements at indices that are out of bounds for the arrays. Array indices start from 0, so `arr2[2]` would be the third element, but your array only contains 5 elements.**

**Q24:**

**- Declare and initialize arrays properly.**

**- Ensure that you access valid indices within the bounds of the arrays.**

**- Make sure that the arrays you are trying to access actually exist and are properly defined in your code.**

**For example, to fix the issues with `arr2`, you can initialize it as follows:**

**int arr2[] = {8, 6, 10, 17, 20};**

**System.out.println(arr2[2]);**

**Q25:**

**1. Create a new request in Postman.**

**2. Set the request method (e.g., GET, POST) and the URL.**

**3. Add request headers, parameters, and request body if necessary.**

**4. Click the "Send" button to make the request to the specified URL.**

**5. Postman sends the request to the server, and the server processes it.**

**6. Postman receives the response from the server.**

**7. The response details, including status code, headers, and response body, are displayed in the Postman interface.**

**Q26: In Postman, "Pre-request Script" and "Test Script" are script blocks that allow you to run custom code before and after a request is sent, respectively.**

**- Pre-request Script: This script is executed before the request is sent to the server. It's useful for tasks like setting environment variables, preparing data, or performing authentication before making the actual request.**

**- Test Script: This script is executed after the response is received from the server. It's used for validating the response, extracting data from it, and performing various tests to ensure the response meets your expectations.**

**In summary, the primary difference is the timing of execution: "Pre-request Script" runs before the request, and "Test Script" runs after the response.**