**Automate the Web Application**

**Writeup:**

Step 1: Import necessary Selenium classes.

In both `Login.java` and `Register.java`, you need to import the required classes to work with Selenium WebDriver. The import statements are present at the beginning of the code, and they ensure that you can use WebDriver functionalities.

Step 2: Set up ChromeDriver path.

Both `Login.java` and `Register.java` set the system property to specify the path to the `chromedriver.exe`. This is essential to tell Selenium which browser driver to use.

Step 3: Create a WebDriver instance.

In both cases, the code creates a new instance of `ChromeDriver`, which represents the Chrome browser. This instance will be used to interact with the browser.

Step 4: Open the web page.

In both cases, the code uses the `driver.get()` method to navigate to the specified URL, which opens the web page.

Step 5: Locate and interact with elements.

Both classes locate various web elements on the page and interact with them. They use different locators like `By.name()`, `By.id()`, `By.xpath()`, `By.cssSelector()`, etc., to find elements.

Step 6: Performing actions on elements.

Once the elements are located, they perform different actions like sending keys (entering data into input fields), clicking checkboxes, selecting options from dropdowns, etc.

Step 7: Handle dynamic wait using `WebDriverWait`.

In `Register.java`, you have used `WebDriverWait` to wait for the dropdown options to be visible before selecting an option. This ensures that the script waits for the elements to be loaded before interacting with them.

Step 8: Use `JavascriptExecutor` to perform certain actions.

In `Register.java`, you have used `JavascriptExecutor` to click on an element. This allows you to execute JavaScript code to interact with elements that might not be directly clickable using the standard `click()` method.

Step 9: Run the test script.

You can run both `Login.java` and `Register.java` as Java applications. They will launch the Chrome browser, open the specified web pages, interact with elements, and perform the actions as specified in the code.

**Note: Ensure that you have the appropriate Selenium WebDriver JAR files in your project's build path, and the ChromeDriver executable is accessible at the specified path. Also, make sure to use the correct locators for the elements you want to interact with.**