**Task-15**

**Automation tesing**

1. Difference between Selenium IDE, Selenium Webdriver, Selenium Grid

**Selenium IDE**

**Overview:** Selenium Integrated Development Environment (IDE) is a browser plugin (available for Chrome and Firefox) that allows users to record and playback scripts to automate browser interactions.

**Key Features:**

* **Record and Playback:** Allows users to record their actions on a web application and playback these actions later.
* **Ease of Use:** No programming knowledge is required, making it accessible for beginners.
* **Script Export:** Can export recorded scripts to various programming languages (Java, C#, Python, etc.) for further customization and use with Selenium WebDriver.
* **Debugging and Editing:** Provides features to debug and edit scripts directly within the IDE.

**Best Suited For:**

* Beginners who need to quickly create automation scripts without writing code.
* Simple test cases and quick prototyping.

**Selenium WebDriver**

**Overview:** Selenium WebDriver is a more advanced and flexible framework compared to Selenium IDE. It allows for the creation of more complex and robust test scripts using various programming languages (Java, C#, Python, Ruby, etc.).

**Key Features:**

* **Programming Language Support:** Allows writing scripts in multiple languages.
* **Browser Control:** Interacts with browsers at a deeper level, offering more control and precision over browser actions.
* **Support for Multiple Browsers:** Supports all major browsers (Chrome, Firefox, Safari, Edge, etc.).
* **Dynamic Web Elements:** Can handle dynamic web content and elements that change over time.
* **Integration with Testing Frameworks:** Easily integrates with testing frameworks like JUnit, TestNG, NUnit, etc.

**Best Suited For:**

* Developers and testers who need to write detailed and customizable test scripts.
* Complex test scenarios requiring conditional logic, loops, and error handling.
* Integration with CI/CD pipelines for continuous testing.

**Selenium Grid**

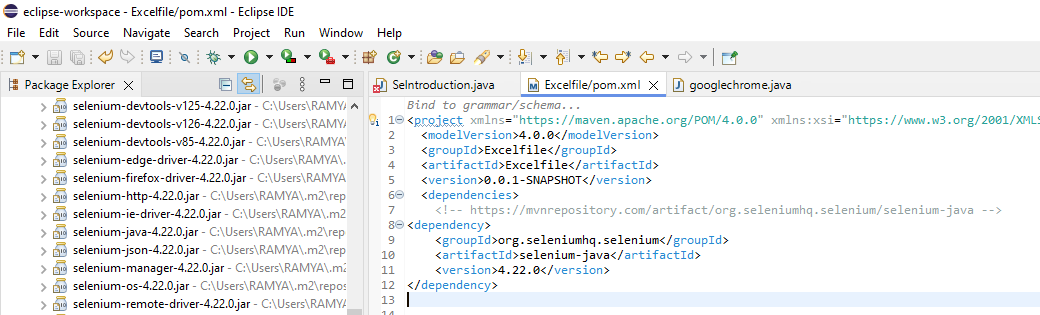
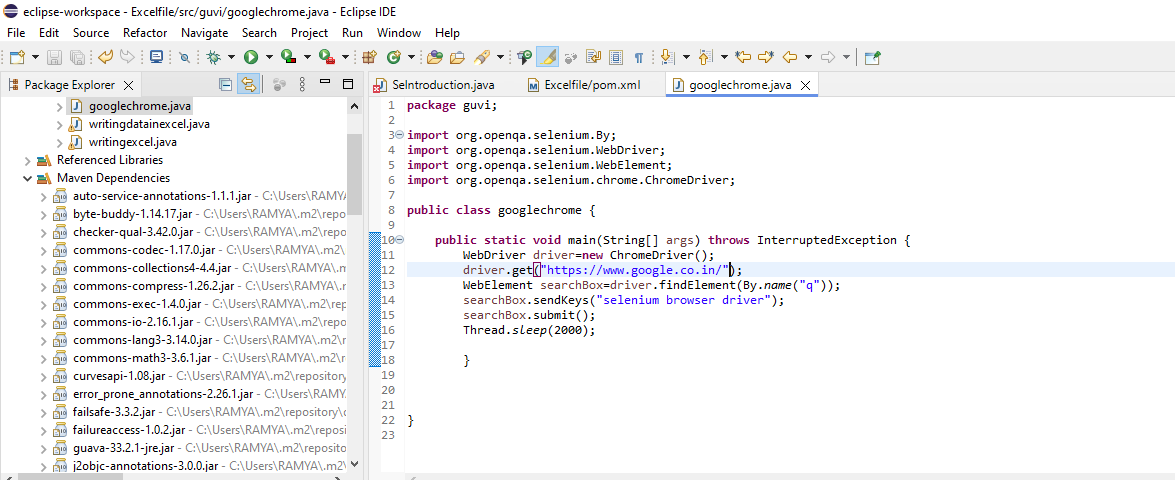
**Overview:** Selenium Grid is used to run tests in parallel across multiple machines and browsers, facilitating distributed test execution.

**Key Features:**

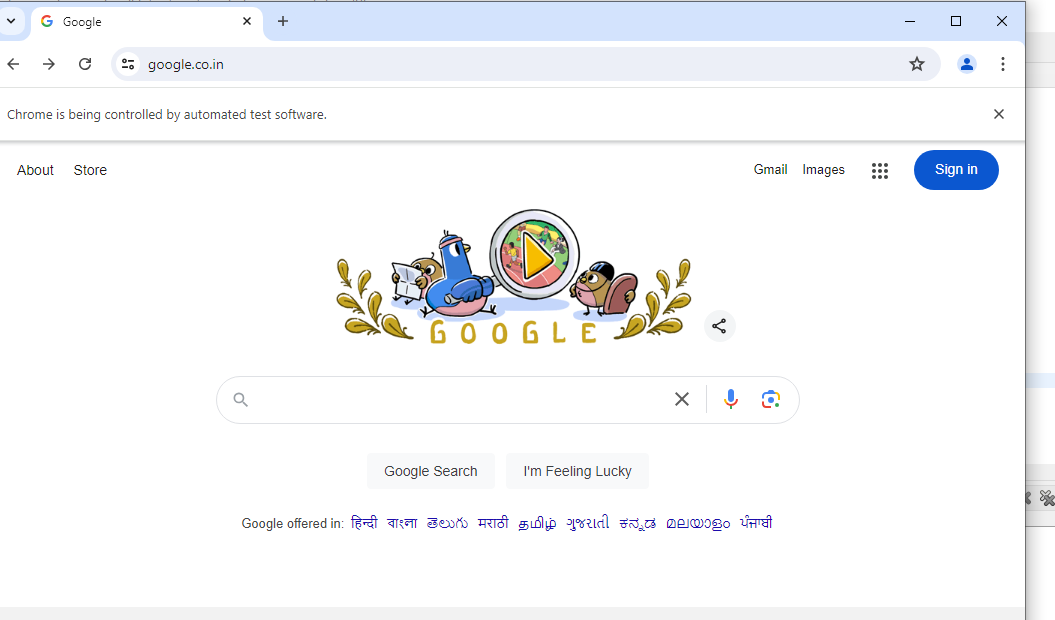
* **Parallel Execution:** Allows running multiple tests concurrently, reducing test execution time.
* **Distributed Testing:** Can distribute tests across different machines (nodes) running different browsers and operating systems.
* **Centralized Management:** Uses a central hub to manage the distribution of tests to different nodes.
* **Scalability:** Easily scales to run tests on a large number of browsers and machines.

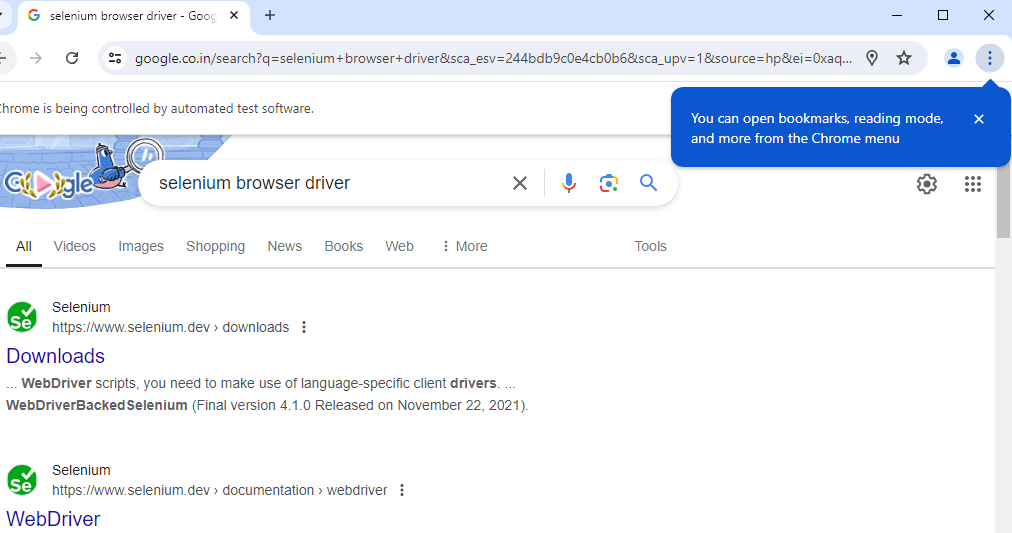
**Best Suited For:**

* Large test suites that need to be executed quickly.
* Environments where tests need to be run across various browsers and operating systems.
* Continuous integration environments requiring fast feedback on test results.

2.

Output:





3.Selenium is an open-source tool and library that automates web browsers. It is widely used for automating web applications for testing purposes, but it can also be used for web scraping, automating repetitive web-based administrative tasks, and more.

**How Selenium is Useful in Automation Testing:**

1. **Cross-Browser Testing**:
   * Selenium supports multiple browsers (e.g., Chrome, Firefox, Safari, Edge), allowing testers to ensure their web application works across different browsers.
2. **Cross-Platform Testing**:
   * Selenium supports multiple operating systems such as Windows, macOS, and Linux, ensuring that applications function correctly on different platforms.
3. **Support for Multiple Languages**:
   * Selenium supports various programming languages (Java, C#, Python, Ruby, JavaScript), making it flexible and accessible to a broad range of developers and testers.
4. **Integration with Testing Frameworks**:
   * Selenium can be integrated with frameworks like TestNG and JUnit for managing and organizing test cases. It can also work with CI/CD tools like Jenkins for continuous testing.
5. **Parallel Test Execution**:
   * Using Selenium Grid, tests can be executed in parallel, significantly reducing the time required for test execution and improving efficiency.
6. **Community and Support**:
   * Being an open-source tool, Selenium has a vast community of developers and testers who contribute to its development and provide support through forums, blogs, and documentation.
7. **Reusable Test Scripts**:
   * Selenium allows for the creation of reusable and maintainable test scripts. Tests can be written once and run across different environments and browsers.

4. Selenium WebDriver supports several browser drivers, each specific to a particular web browser. These drivers act as a bridge between Selenium and the web browser, allowing Selenium to control the browser and automate web tasks. Here are the primary browser drivers used in Selenium:

Chrome driver, Jecko Driver, Internet Explorer Driver, Edge Driver, Safari Driver,Opera Drivver , Htmlunit Driver.

### 5. Steps:

1. **Set Up Your Development Environment:**
   * Ensure you have JDK installed.
   * Set up a project in your preferred IDE (e.g., IntelliJ IDEA, Eclipse).
2. **Add Selenium Dependency:**
   * If using Maven, add Selenium dependency to your pom.xml file.
3. **Download WebDriver:**
   * Download the appropriate WebDriver executable for your browser
4. **Import necessary packages:**
   * Import the required packages from Selenium.
5. **Initialize WebDriver:**
   * Create an instance of the WebDriver for your browser.
6. **Open a webpage:**
   * Use the get method to navigate to a specific URL.
7. **Interact with the webpage:**
   * Perform actions like finding elements, clicking buttons, entering text, etc.
8. **Close the browser:**
   * Properly close the browser using the quit method.

### Example Code:)

Include the Selenium dependency in your pom.xml file: <dependencies>

<dependency>

<groupId>org.seleniumhq.selenium</groupId>

<artifactId>selenium-java</artifactId>

<version>4.2.0</version>

</dependency>

</dependencies>

Sample code:

**package** guvi;

**import** org.openqa.selenium.By;

**import** org.openqa.selenium.WebDriver;

**import** org.openqa.selenium.WebElement;

**import** org.openqa.selenium.chrome.ChromeDriver;

**public** **class** googlechrome {

**public** **static** **void** main(String[] args) **throws** InterruptedException {

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

driver.get("https://www.google.co.in/");

WebElement searchBox=driver.findElement(By.*name*("q"));

searchBox.sendKeys("selenium browser driver");

searchBox.submit();

Thread.*sleep*(2000);

}

}

**Explanation:**

1. **Importing Packages:**

import org.openqa.selenium.Keys;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.chrome.ChromeDriver;

* + These imports bring in the necessary classes from the Selenium library to work with WebDriver.

1. **Initializing WebDriver:**

WebDriver driver = new ChromeDriver();

* + This initializes the Chrome WebDriver.

1. **Opening a Webpage:**

driver.get("https://www.google.com");

* + This navigates the browser to the specified URL.

1. **Interacting with the Webpage:**

WebElement searchBox = driver.findElement(By.name("q"));

searchBox.sendKeys("Selenium WebDriver");

searchBox.sendKeys(Keys.RETURN);

* + This finds the search box element by its name attribute (q), enters the text "Selenium WebDriver", and simulates pressing the ENTER key.

1. **Adding a Delay:**

Thread.sleep(5000);

* + This adds a delay of 5 seconds to allow you to see the search results before the browser closes.

1. **Closing the Browser:**

driver.quit();

* + This closes the browser and ends the WebDriver session.