From this webside ----->https://www.selenium.dev/downloads/

1.Download Selenium JARs

Java

Stable: 4.27.0 (November 25, 2024)

Changelog

API Docs

2. https://googlechromelabs.github.io/chrome-for-testing/

Stable

Version: 131.0.6778.85 (r1368529)

Binary	Platform	URL	HTTP stat
chrome	linux64	https://storage.googleapis.com/chrome-for-testing-public/131.0.6778.85/linux64/chrome-linux64.zip	200
chrome	mac-arm64	https://storage.googleapis.com/chrome-for-testing-public/131.0.6778.85/mac-arm64/chrome-mac-arm64.zip	200
chrome	mac-x64	https://storage.googleapis.com/chrome-for-testing-public/131.0.6778.85/mac-x64/chrome-mac-x64.zip	200
chrome	win32	https://storage.googleapis.com/chrome-for-testing-public/131.0.6778.85/win32/chrome-win32.zip	200
chrome	win64	https://storage.googleapis.com/chrome-for-testing-public/131.0.6778.85/win64/chrome-win64.zip	200
chromedriver	linux64	https://storage.googleapis.com/chrome-for-testing-public/131.0.6778.85/linux64/chromedriver-linux64.zip	200
chromedriver	mac-arm64	https://storage.googleapis.com/chrome-for-testing-public/131.0.6778.85/mac-arm64/chromedriver-mac-arm64.zip	200
chromedriver	mac-x64	https://storage.googleapis.com/chrome-for-testing-public/131.0.6778.85/mac-x64/chromedriver-mac-x64.zip	200
chromedriver	win32	https://storage.googleapis.com/chrome-for-testing-public/131.0.6778.85/win32/chromedriver-win32.zip	200
chromedriver	win64	https://storage.googleapis.com/chrome-for-testing-public/131.0.6778.85/win64/chromedriver-win64.zip	200

https://storage.googleapis.com/chrome-for-testing-public/131.0.6778.85/win64/chromedriver-win64.zip

Copy this cromedriver file and paste in new tab

Extract both zip file

Project Setup

Create a Java Project in your IDE (like Eclipse or IntelliJ IDEA).

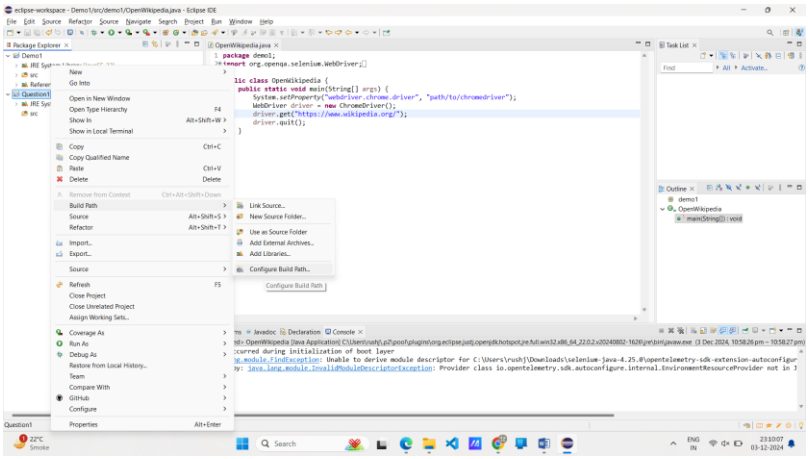
Add Selenium JARs to your project's build path:

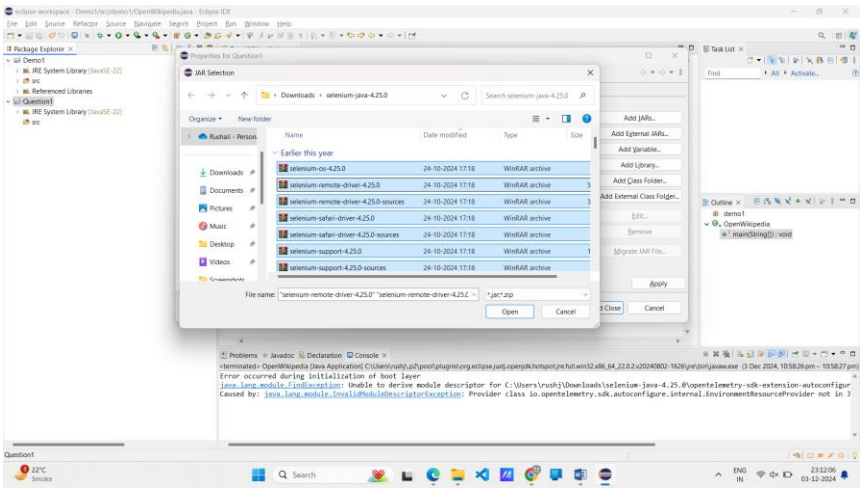
Right-click on your project.

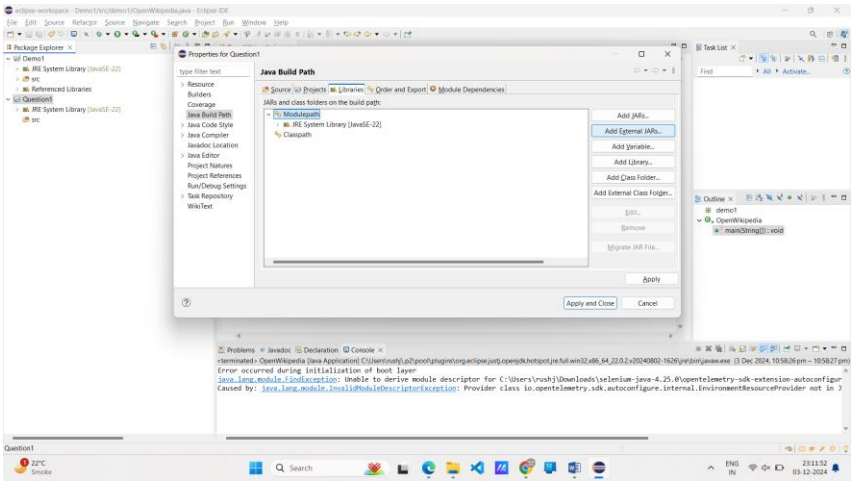
Go to Build Path -> Configure Build Path.

Add external JARs and include the Selenium JAR files.

Screenshot:







Open-> Apply and Close

1.Write a Selenium script to open the URL https://www.wikipedia.org/ in google browsers(Chrome). Make sure to set up the required WebDriver for google browser.

Answer:

```
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;

public class OpenWikipedia {
    public static void main(String[] args) {
        System.setProperty("webdriver.chrome.driver", "path/to/chromedriver");
        WebDriver driver = new ChromeDriver();
        driver.get("https://www.wikipedia.org/");
        driver.quit();
    }
}
```

2. Open the URL https://www.amazon.com/ in Chrome. Then, perform the following actions:

- **Click on the "Best Sellers" link.**
- **Go back to the previous page.**
- **Refresh the current page.**

```
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;

public class AmazonNavigation {
    public static void main(String[] args) {
        System.setProperty("webdriver.chrome.driver", "path/to/chromedriver");
        WebDriver driver = new ChromeDriver();
        driver.get("https://www.amazon.com/");

        WebElement bestSellers = driver.findElement(By.linkText("Best Sellers"));
        bestSellers.click();

        driver.navigate().back();
        driver.navigate().refresh();

        driver.quit();
    }
}
```

3. Visit the URL https://www.w3schools.com/html/html_forms.asp.

- **Locate the checkbox for "I agree to receive email notifications" and check it.**
- **Uncheck the checkbox.**
- **Locate and select the radio button for "Male".**

```
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;

public class W3SchoolsForm {
   public static void main(String[] args) {
      System.setProperty("webdriver.chrome.driver", "path/to/chromedriver");
      WebDriver driver = new ChromeDriver();
      driver.get("https://www.w3schools.com/html/html_forms.asp");

      WebElement checkBox = driver.findElement(By.xpath("//input[@type='checkbox' and
@name='subscribe']"));
      checkBox.click();
      checkBox.click();

      WebElement radioButton = driver.findElement(By.xpath("//input[@type='radio' and
@value='male']"));
      radioButton.click();

      driver.quit();
   }
}
```

4. Write a Selenium script to:(any website)

- **Select a dropdown option by visible text.**
- **Select a dropdown option by value.**

```
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.support.ui.Select;

public class SelectDropdown {
    public static void main(String[] args) {
        System.setProperty("webdriver.chrome.driver", "path/to/chromedriver");
        WebDriver driver = new ChromeDriver();
        driver.get("https://www.w3schools.com /");

        WebElement dropdown = driver.findElement(By.id("dropdown_id"));
        Select select = new Select(dropdown);

        select.selectByVisibleText("Visible Text Option");
        select.selectByValue("value_option");

        driver.quit();
    }
}
```

5. **Write a Selenium script that performs the following actions: Open the URL https://www.example.com.**
   - **Navigate to a new URL https://www.google.com using the get() method.**
   - **Navigate back to the previous page.**
   - **Refresh the page.**

```
import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;



public class NavigationActions {

    public static void main(String[] args) {

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

        WebDriver driver = new ChromeDriver();

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



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

        driver.navigate().back();

        driver.navigate().refresh();



        driver.quit();

    }

}
```

6. **Write a Selenium script to upload a file using the sendKeys() method for an input element with type="file".**

```
import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.chrome.ChromeDriver;


public class FileUpload {

    public static void main(String[] args) {

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

        WebDriver driver = new ChromeDriver();

        driver.get("URL_OF_YOUR_CHOICE");


        WebElement uploadElement = driver.findElement(By.id("file_upload_id"));

        uploadElement.sendKeys("C:\\path\\to\\your\\file.txt");


        driver.quit();

    }

}
```

7. Write a Selenium WebDriver script to open the URL https://twitter.com/settings/account.

- **Select a checkbox if it is not already selected.**
- **Deselect it if it is selected.**

```
import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.chrome.ChromeDriver;



public class TwitterSettings {

    public static void main(String[] args) {

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

        WebDriver driver = new ChromeDriver();

        driver.get("https://twitter.com/settings/account");



        WebElement checkbox =
driver.findElement(By.cssSelector("input[type='checkbox']"));

        if (!checkbox.isSelected()) {

            checkbox.click();

        } else {

            checkbox.click();

        }



        driver.quit();

    }

}
```

8. **Write a Selenium script that: (Any Website)**
    - **Selects an option from a dropdown by visibleText using the Select class.**
    - **Selects multiple options from a multi-select dropdown using the Select class.**

```
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.support.ui.Select;

public class DropdownInteraction {
   public static void main(String[] args) {
      System.setProperty("webdriver.chrome.driver", "path/to/chromedriver");
      WebDriver driver = new ChromeDriver();
      driver.get("URL_OF_YOUR_CHOICE");

      WebElement singleSelectDropdown =
driver.findElement(By.id("single_select_id"));
      Select singleSelect = new Select(singleSelectDropdown);
      singleSelect.selectByVisibleText("Option Text");

      WebElement multiSelectDropdown =
driver.findElement(By.id("multi_select_id"));
      Select multiSelect = new Select(multiSelectDropdown);
      multiSelect.selectByVisibleText("Option 1");
      multiSelect.selectByVisibleText("Option 2");

      driver.quit();
   }
}
```

9. **Write a Selenium WebDriver script that opens a browser and closes it based on user input. The script should prompt the user for input to either open or close the browser**

```
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import java.util.Scanner;

public class OpenCloseBrowser {
   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);
      System.out.println("Enter 'open' to open the browser or 'close' to close the
browser:");
      String input = scanner.nextLine();

      System.setProperty("webdriver.chrome.driver", "path/to/chromedriver");
      WebDriver driver = null;

      if ("open".equalsIgnoreCase(input)) {
         driver = new ChromeDriver();
         driver.get("https://www.example.com");
      }

      System.out.println("Enter 'close' to close the browser:");
      input = scanner.nextLine();

      if ("close".equalsIgnoreCase(input) && driver != null) {
         driver.quit();
      }

      scanner.close();
   }
}
```

10. How do you set up the environment for running JMeter tests on your machine?

Download JMeter:

Go to the JMeter official website.
Download the binary archive (ZIP file).
Install JMeter:

Extract the downloaded ZIP file to a directory of your choice (e.g., C:\JMeter).
Set Environment Variables:

Open Control Panel > System and Security > System > Advanced system settings.
Click on Environment Variables.

In the System variables section, find the Path variable and click Edit.
Add the path to the JMeter bin directory (e.g., C:\JMeter\apache-jmeter-5.4.1\bin).
Verify Installation:

Open Command Prompt.
Type jmeter and press Enter. The JMeter GUI should start.

11. In JMeter, how would you configure a Thread Group **to simulate 100 users over 10 minutes with a ramp-up time of 1 minute?**

- **Open JMeter**:
  - Run `jmeter` from Command Prompt to open the JMeter GUI.
- **Add a Thread Group**:
  - Right-click on `Test Plan`.
  - Select `Add > Threads (Users) > Thread Group`.
- **Configure the Thread Group**:
  - Set `Number of Threads (users)` to `100`.
  - Set `Ramp-Up Period (in seconds)` to `60`.
  - Check `Scheduler` and set `Duration (seconds)` to `600`.

12 .Create a simple web test plan in JMeter to test a website's homepage. Include HTTP request and a listener.

- **Open JMeter and Add a Thread Group**:
  - Right-click on `Test Plan`.
  - Select `Add > Threads (Users) > Thread Group`.
- **Add an HTTP Request**:
  - Right-click on `Thread Group`.
  - Select `Add > Sampler > HTTP Request`.
  - Configure the HTTP request:
    - Server Name or IP: `example.com`
    - Path: `/`
- **Add a Listener**:
  - Right-click on `Thread Group`.
  - Select `Add > Listener > View Results Tree`.
- **Save and Run the Test Plan**:
  - Save your test plan (`File > Save`).
  - Click the green `Start` button to run the test.

13. How would you configure a JMeter test plan for testing a RESTful API?

- **Open JMeter and Add a Thread Group**:
  - Right-click on `Test Plan`.
  - Select `Add > Threads (Users) > Thread Group`.
- **Add an HTTP Request**:
  - Right-click on `Thread Group`.
  - Select `Add > Sampler > HTTP Request`.
  - Configure the HTTP request for the RESTful API:
    - Method: `GET`, `POST`, `PUT`, etc.
    - Server Name or IP: `api.example.com`
    - Path: `/endpoint`
    - Add any necessary parameters in the `Parameters` section.
- **Add Headers** (if needed):
  - Right-click on `Thread Group`.
  - Select `Add > Config Element > HTTP Header Manager`.
  - Add necessary headers (e.g., `Content-Type: application/json`).
- **Add a Listener**:
  - Right-click on `Thread Group`.
  - Select `Add > Listener > View Results Tree`.
- **Save and Run the Test Plan**:
  - Save your test plan.
  - Click the green `Start` button to run the test.

14. How do you configure a JMS Point-to-Point Test Plan in JMeter?

- **Open JMeter and Add a Thread Group**:
  - Right-click on `Test Plan`.
  - Select `Add > Threads (Users) > Thread Group`.
- **Add JMS Point-to-Point Sampler**:
  - Right-click on `Thread Group`.
  - Select `Add > Sampler > JMS Point-to-Point`.
  - Configure the JMS Point-to-Point sampler:
    - JNDI Initial Context Factory: `org.apache.activemq.jndi.ActiveMQInitialContextFactory`
    - Provider URL: `tcp://localhost:61616`
    - Connection Factory: `ConnectionFactory`
    - Destination: `queue/TestQueue`
- **Add a Listener**:
  - Right-click on `Thread Group`.
  - Select `Add > Listener > View Results Tree`.
- **Save and Run the Test Plan**:
  - Save your test plan.
  - Click the green `Start` button to run the test.

15. Create a Database Test Plan in JMeter to test performance when executing SQL queries against a database.

1. **Open JMeter and Add a Thread Group**:
   - Right-click on `Test Plan`.
   - Select `Add > Threads (Users) > Thread Group`.
2. **Add JDBC Connection Configuration**:
   - Right-click on `Thread Group`.
   - Select `Add > Config Element > JDBC Connection Configuration`.
   - Configure the JDBC connection:
     - Database URL: `jdbc:mysql://localhost:3306/testdb`
     - JDBC Driver class: `com.mysql.cj.jdbc.Driver`
     - Username: `your-username`
     - Password: `your-password`
3. **Add JDBC Request**:
   - Right-click on `Thread Group`.
   - Select `Add > Sampler > JDBC Request`.
   - Configure the JDBC request:
     - Variable Name: `mydb`
     - Query Type: `Select Statement`
     - Query: `SELECT * FROM users`
4. **Add a Listener**:
   - Right-click on `Thread Group`.
   - Select `Add > Listener > View Results Tree`.
5. **Save and Run the Test Plan**:
   - Save your test plan.
   - Click the green `Start` button to run the test.

16. How would you monitor real-time server performance metrics while running a JMeter test? Describe how to integrate Server Monitoring with JMeter.

- **Install Plugins**:
  - Download the JMeter Plugins Manager.
  - Place the JAR file in the `lib/ext` directory of your JMeter installation.
- **Open JMeter and Install Server Monitoring Plugins**:
  - Open JMeter.
  - Go to `Options > Plugins Manager`.
  - Install the necessary server monitoring plugins (e.g., PerfMon).
- **Add PerfMon Metrics Collector**:
  - Right-click on `Test Plan`.
  - Select `Add > Listener > jp@gc - PerfMon Metrics Collector`.
- **Add a Server Agent**:
  - Download the Server Agent.
  - Start the Server Agent on the server you want to monitor.
- **Configure PerfMon Metrics Collector**:
  - Set the `Host` and `Port` for the Server Agent.
  - Add the desired metrics (e.g., CPU, Memory).
- **Save and Run the Test Plan**:
  - Save your test plan.
  - Click the green `Start` button to run the test and monitor real-time server performance metrics.

17. How do Listeners impact JMeter's performance? What are some best practices for using listeners when performing large-scale load tests?

Impact:

- Listeners can significantly impact JMeter's performance, especially when dealing with large-scale load tests.
- They consume memory and CPU resources, which can slow down the test execution and lead to inaccurate results.

Best Practices:

- Use listeners sparingly and only when necessary.
- Prefer listeners that write results to a file rather than displaying them in the GUI (e.g., Simple Data Writer).
- Disable or remove listeners during large-scale load tests.
- Use non-GUI mode for large-scale tests to minimize resource consumption.

18. Implement a User-Defined Function in JMeter to generate a dynamic value for each user in a load test. Explain how you can use the __UUID() function to generate unique IDs for each request.

```
import org.apache.jmeter.config.Arguments;
import org.apache.jmeter.config.gui.ArgumentsPanel;
import org.apache.jmeter.functions.AbstractFunction;
import org.apache.jmeter.functions.Function;
import org.apache.jmeter.functions.FunctionParameter;
import org.apache.jmeter.functions.FunctionUtils;
import org.apache.jmeter.testelement.property.JMeterProperty;
import org.apache.jmeter.testelement.property.StringProperty;
import org.apache.jorphan.logging.LoggingManager;
import org.apache.log.Logger;

import java.util.Collection;
import java.util.LinkedList;
import java.util.List;
import java.util.UUID;

public class UUIDFunction extends AbstractFunction {
```

```
    private static final List<String> desc = new LinkedList<>();
    private static final String KEY = "__UUID";

    static {
        desc.add("Generates a unique identifier (UUID)");
    }

    private static final Logger log =
LoggingManager.getLoggerForClass();

    @Override
    public String execute(SampleResult previousResult, Sampler
currentSampler) {
        return UUID.randomUUID().toString();
    }

    @Override
    public void setParameters(Collection<FunctionParameter> parameters)
throws InvalidVariableException {
        checkParameterCount(parameters, 0);
    }

    @Override
    public String getReferenceKey() {
        return KEY;
    }

    @Override
    public List<String> getArgumentDesc() {
        return desc;
    }
}
```