

# 软件测试上机报告



## 第五次上机作业 - Lab 5 Selenium

学 院 智能与计算学部

专 业 软件工程

姓 名 郎文翀

学 号 **3019244247**

年 级 2019 级

班 级 软件工程 5 班

## 1. Experimental requirements

1. Install Selenium with Eclipse or IntelliJ IDEA.
2. Install Firefox/Chrome (Chrome is recommended) and SeleniumIDE plugin.
3. Try to record and export scripts using SeleniumIDE.
4. Please complete the following task using Selenium Webdriver:  
“user\_info.csv” contains information about the students, and <http://118.178.137.170:8080/> can view someone's information after logging in (simplified student id). Please check each record in the csv file to make sure that each student's information is consistent with the information on the website.

### Requirements for the experiment:

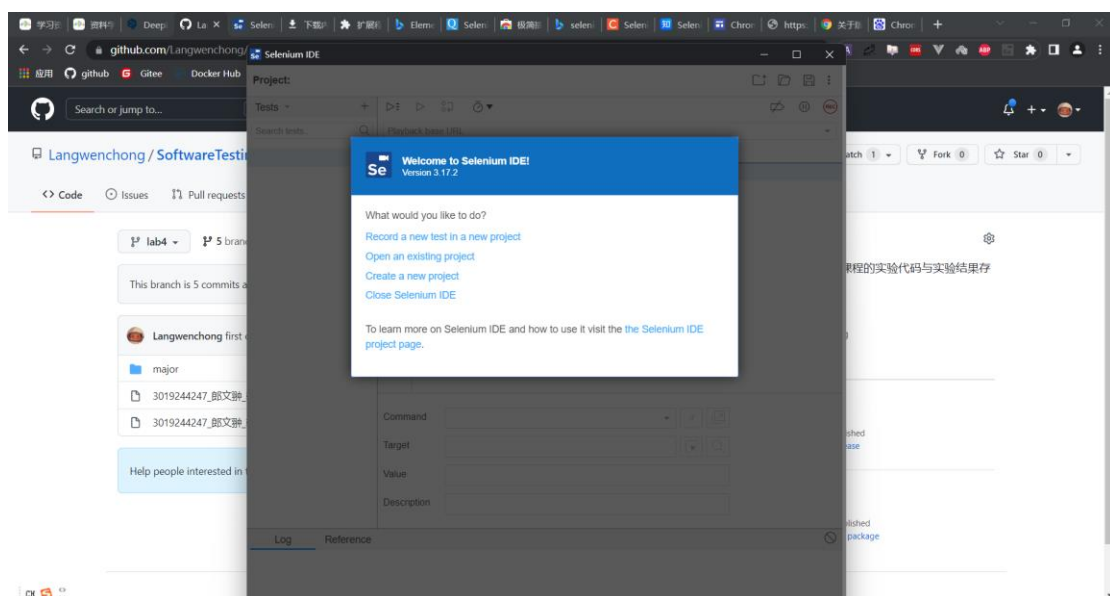
1. Finish the tasks above individually.
2. Check in your java code to github or gitee
3. Please send your experiment report to 智慧树, the following information should be included in your report:
  - a) The brief description that you install Selenium, Firefox/Chrome and SeleniumIDE.
  - b) Steps for recording and exporting scripts.
  - c) Steps for testing the website using Selenium Webdriver.

**Submission deadline:** 23:59 April 16, 2022.

## 2. Configuration

### 2.1 在 chrome 上安装 selenium ide 插件

我是在 chrome 上进行实验的，因此首先我们需要在 chrome 上安装 selenium ide 插件，点击链接 [website](#) 下载 crx 插件，由于插件是在 googles 上，需要科学上网，安装完成以后我们将插件固定在右上角，就可以看到安装成功的插件了如下图所示：

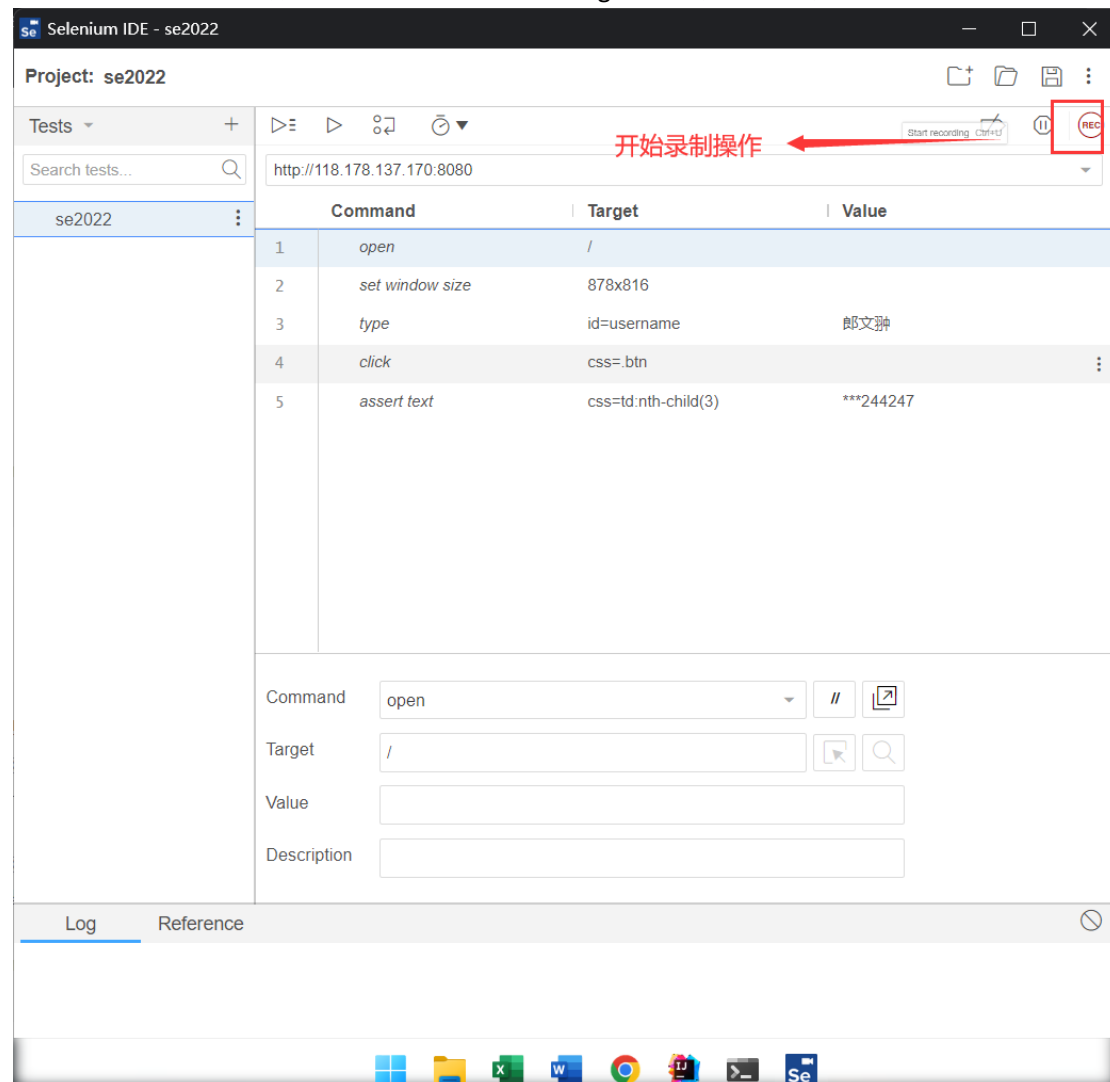


然后我们运行这个插件，首先我们需要新建一个项目，我将其命名为了 se2022 项目，然后输入网页的根路径是 <http://118.178.137.170:8080/>，然后我们就可以开始录制我们的操作了。

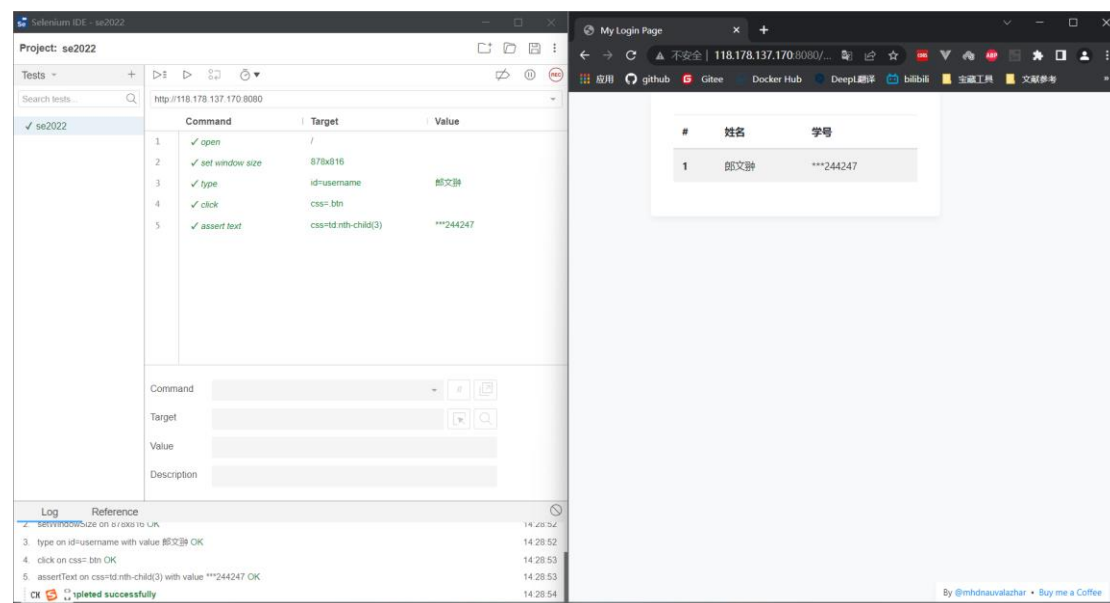
### 2.2 录制操作

首先我们简述一下 selenium 是干什么的？他可以捕捉我们对于页面的操作，比如输入、点击、获取元素值等然后记录成功以后保存为 side 文件，再次点击运行就可以自动执行我们之前记录的操作完成对页面的操作。这里我们实验要求要将网站中存储得每一个键值对 学号-姓名信息获取下来和本地的 user\_info.csv 文件中的信息进行比对，因此我们需要完成一个自动化在网页上输入学生姓名并获取学生学号的操作，这就用到了 selenium。首先我们需要开启 selenium 录制获取某一个学生的信息，这里我已获取我的信息为例，首先我们需要

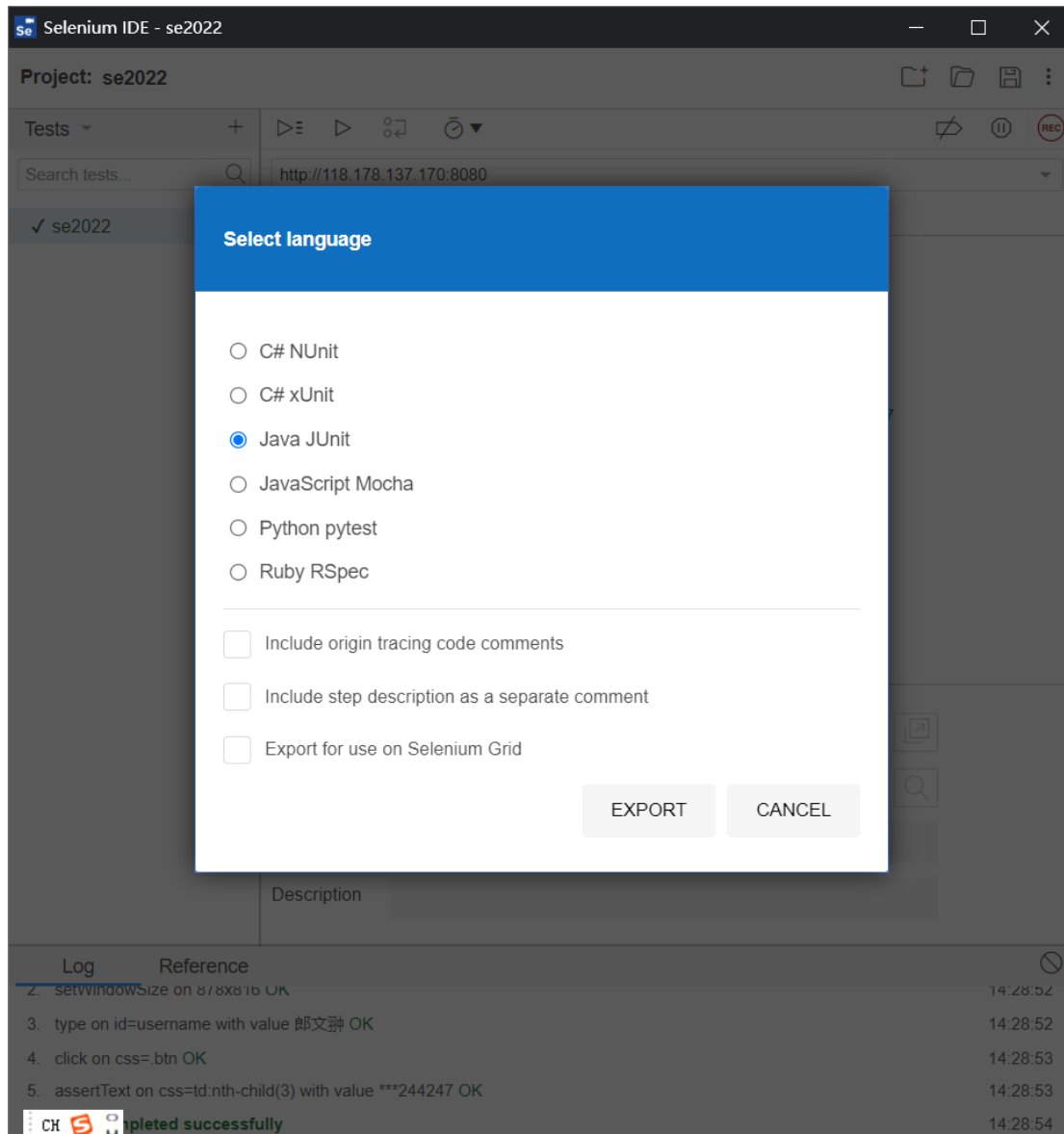
进入网站以后点击插件的可视化页面中的 starting record 按钮进行操作的录制：



然后在输入框中输入郎文翀姓名，紧接着我们就来到了具体的信息页面，这里我们需要选中学号右键选择 store 属性将学号存储，这样我们就完成了一次信息的获取操作如下图所示：



我们可以看到左侧的框中记录了我们的每一次操作，并且只有全为绿色时说明所有的操作都执行成功了。接下来我们将这个记录保存为 `side` 文件，并且将这个操作导出为 `junit` 一个 `java` 的代码，方便我们后面进行修改改为多次获取不同姓名学生对应的学号并进行对比，将导出的 `java` 代码命名为 `se2022Tes.java`:



然后我们新建一个项目，打开这个 `se2022Test.java` 文件可以看到具体的代码如下所示：

```
public class Se2022Test {
    private WebDriver driver;
    private Map<String, Object> vars;
    JavascriptExecutor js;
    @Before
    public void setUp() {
        driver = new ChromeDriver();
        js = (JavascriptExecutor) driver;
        vars = new HashMap<String, Object>();
    }
    @After
```

```

public void tearDown() {
    driver.quit();
}
@Test
public void se2022() {
    driver.get("http://118.178.137.170:8080/");
    driver.manage().window().setSize(new Dimension(878, 816));

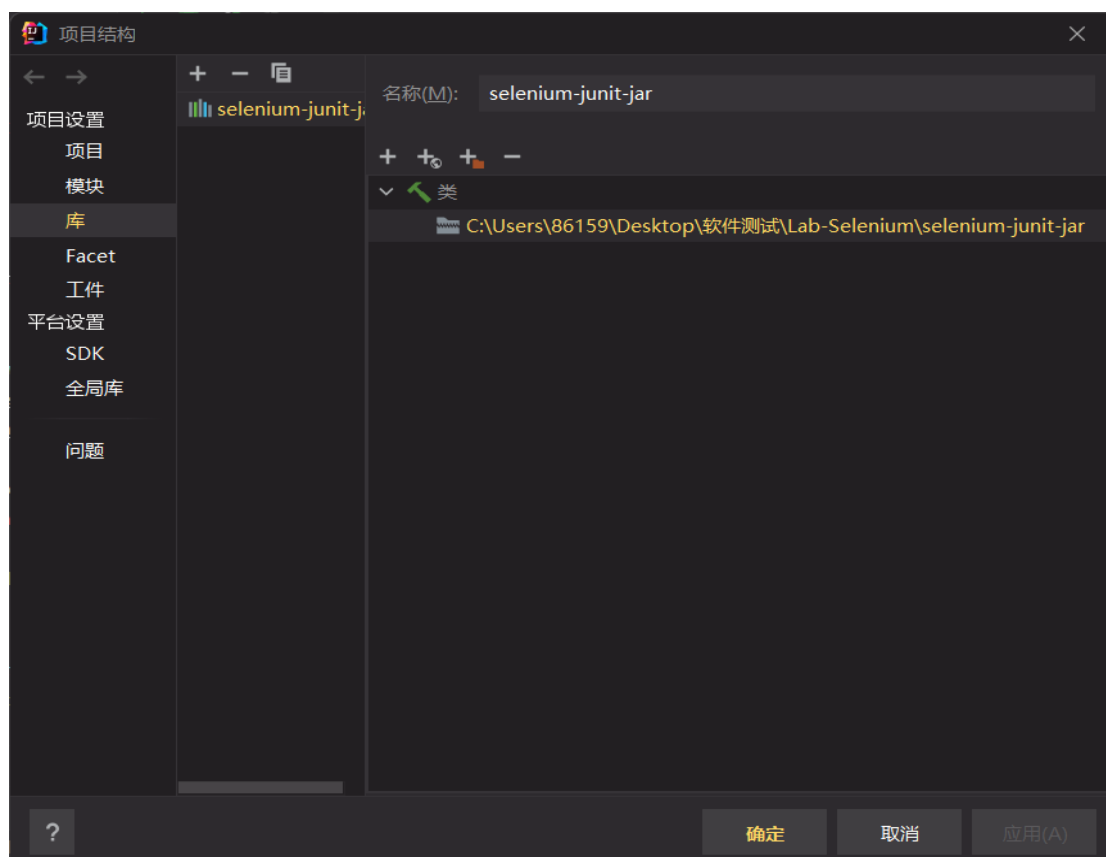
    driver.findElement(By.id("username")).sendKeys("郎文翀");
    driver.findElement(By.cssSelector(".btn")).click();
    vars.put("sNumber",
driver.findElement(By.cssSelector("td:nth-child(3)")).getText());
}
}

```

我们可以看到这个就是我们之前记录的操作的对应的 java 代码，他是在@before 函数中初始化了一个 driver 驱动实例，然后定位到 baseUrl,接下来就是具体的操作对应的代码了，这里都是写死的获取郎文翀对应的学号。

## 2.3 导入依赖与驱动

我们发现这个时候的代码中还有许多报错，这是因为我们还需要导入所有的依赖比如 junit，driver，同时还需要安装 chromeDriver 驱动，首先我们导入老师给的依赖包到 idea 中，具体的操作就是打开设置->项目结构->库导入新的 java 依赖如下所示：



接下来我们还需要加载 chrome driver 驱动，首先我们还是需要前往链接 <https://chromedriver.chromium.org/downloads> 下载驱动，这里我们要确保下载的版本与 chrome 一致，我们可以在浏览器中输入 chrome://version 查看具体的版本信息，我这里是 100，因此下载 win32 位 version 为 100.x 的版本。然后下载下来后解压，将存有 chromeDriver.exe 的文件夹加入到系统变量 Path 中即可。

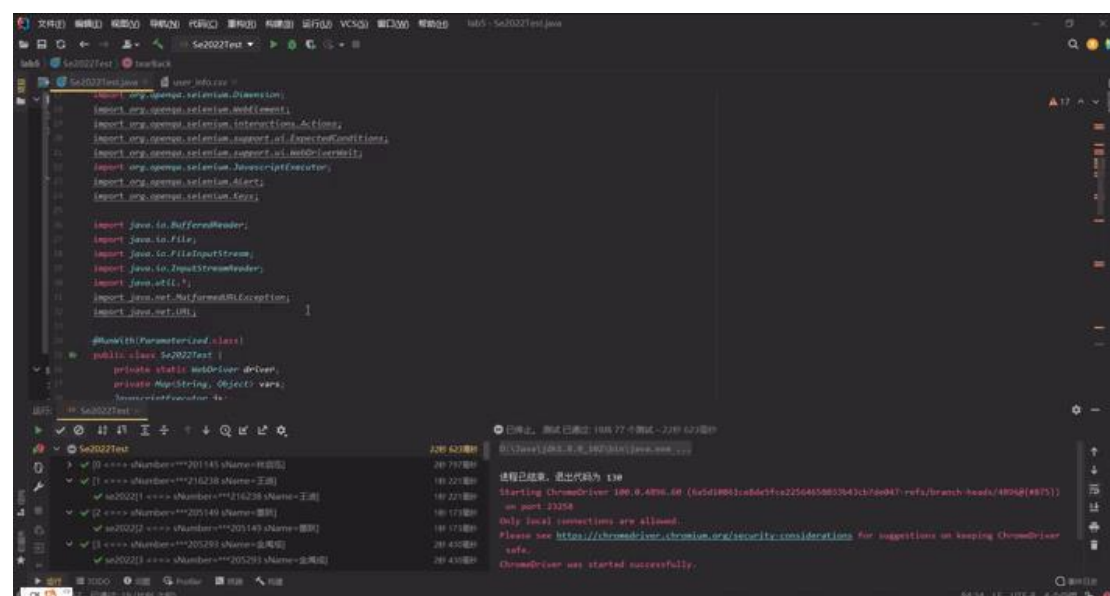
## 2.4 修改代码为获取所有信息并比对

接下来我们还需要进行代码修改，将其改成可以自动获取每一个学生对应的学号并与本地读取的 user\_info.csv 数据进行比对。首先我们需要使用 BufferedReader 将 user\_info.csv 中的数据按行读出并存储为键值对形式存储到一个 List 中，然后由于是自动化进行 test，因此我们需要使用到 lab2 中学习的 @runwith 进行自动化测试，使用 @parameters 进行测试变量的填充，具体的代码请查看第四部分。同时我们还发现代码中将我们爬取的学号信息 sNumber 存储到了一个 vars 的 map 中，因此我们还需要将自动填充的 user\_info.csv 信息与爬取的 sNumber 进行对比，这里使用 lab1 学习的 assertEquals 函数即可。

自此我们就完成了代码的完善，他可以自动多次调用 driver 进行不同学生的学号爬取并与本地的数据进行对比，但是我们这样存在一个问题，即每一次测试都需要重新启动一个浏览器，这显然很低效并且消耗计算机性能，因此我们需要进行修改。

使用 @before 将初始化变量，使用 @after 进行页面的回退与状态刷新，而 driver 的创建与初始化只进行一次，这样我们就可以仅启动一个页面进行测试，具体的代码请查看第四部分。同时我们还要注意一个问题，即 user\_info.csv 的第一行并不是学生信息，需要跳过读取。

一下是我的运行截图（gif）：



The screenshot shows an IDE with a Java file named 'Se2022Test.java'. The code imports Selenium WebDriver classes and Java IO classes. It defines a class 'Se2022Test' with a 'driver' field and a 'vars' map. The code includes a 'runTest' method that reads data from 'user\_info.csv' and compares it with the data from the web. The console output shows the test results, including the number of tests passed and failed, and the time taken to execute the tests.

```
import org.openqa.selenium.Dimension;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.interactions.Actions;
import org.openqa.selenium.support.ui.ExpectedConditions;
import org.openqa.selenium.support.ui.WebDriverWait;
import org.openqa.selenium.JavascriptExecutor;
import org.openqa.selenium.Alert;
import org.openqa.selenium.Keys;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;
import java.io.InputStreamReader;
import java.util.*;
import java.net.MalformedURLException;
import java.net.URL;

@RunWith(Parameterized.class)
public class Se2022Test {
    private static WebDriver driver;
    private Map<String, Object> vars;
    private int testCount = 0;

    @Before
    public void setUp() throws MalformedURLException {
        driver = new WebDriver(new URL("http://localhost:4242/"));
    }

    @Test
    public void test() {
        // ... test logic ...
    }
}
```

控制台输出：

```
已停止。测试已通过: 100.77 个测试 - 2.19 6.2 秒
Starting ChromeDriver 100.0.4896.48 (62457641c0b54e32254435803343c2e947-refa/branch-heads/4896@14875)
on port 4242
Only local connections are allowed.
Please see https://chromedriver.chromium.org/security-considerations for suggestions on keeping Chromedriver
safe.
ChromeDriver was started successfully.
```

## 3. Result analysis

经过本次实验，我们完成了使用 selenium ide 插件记录获取数据的操作并导出为 java 代码后，进一步进行修改，并使用 chromeDrover 驱动完成了自动获取所有学生的数据并与本地进行对比测试的操作，从中体会到了 selenium 的强大之处。这里一定要注意保证每一次测试都

是基于一个浏览器实现，这就需要使用@before 和@after 的合理使用，同时使用 runwith 进行变量的自动填充。

## 4. Source code

```
// Generated by Selenium IDE

import org.junit.*;

import static org.junit.Assert.*;
import static org.hamcrest.CoreMatchers.is;
import static org.hamcrest.core.IsNot.not;

import org.junit.runner.RunWith;
import org.junit.runners.Parameterized;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.remote.RemoteWebDriver;
import org.openqa.selenium.remote.DesiredCapabilities;
import org.openqa.selenium.Dimension;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.interactions.Actions;
import org.openqa.selenium.support.ui.ExpectedConditions;
import org.openqa.selenium.support.ui.WebDriverWait;
import org.openqa.selenium.JavascriptExecutor;
import org.openqa.selenium.Alert;
import org.openqa.selenium.Keys;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;
import java.io.InputStreamReader;
import java.util.*;
import java.net.MalformedURLException;
import java.net.URL;

@RunWith(Parameterized.class)
public class Se2022Test {
    private static WebDriver driver;
    private Map<String, Object> vars;
    JavascriptExecutor js;

    @Parameterized.Parameter(0)
    public String sNumber;
```



```

    @Parameterized.Parameter(1)
    public String sName;

    @Parameterized.Parameters(name = "{index} <==> sNumber={0}
sName={1}")
    public static List<String[]> getParameters() {
        List<String[]> list = new ArrayList<>();
        File inputFile = new File("user_info.csv");
        if (!inputFile.exists()) {
            System.out.println("error");
            System.exit(-1);
        }
        try {
            BufferedReader br = new BufferedReader(new
InputStreamReader((new FileInputStream(inputFile)), "UTF-8"));
            String line = br.readLine();
            while ((line = br.readLine()) != null) {
                String[] tmp = line.split(",");
                if (tmp[0].equals("学号")) {
                    continue;
                }
                list.add(tmp);
            }
            br.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
        return list;
    }

    @BeforeClass
    public static void preparation(){
        driver = new ChromeDriver();
    }

    @Before
    public void setUp() {
        js = (JavascriptExecutor) driver;
        vars = new HashMap<String, Object>();
    }

    @After
    public void tearBack() {
        driver.navigate().back();
        driver.navigate().refresh();
//        driver.quit();
    }

    @Test
    public void se2022() {

```

```
driver.get("http://118.178.137.170:8080/");
driver.manage().window().setSize(new Dimension(878, 816));
driver.findElement(By.id("username")).sendKeys(this.sName);
driver.findElement(By.cssSelector(".btn")).click();
vars.put("sNumber",
driver.findElement(By.cssSelector("td:nth-child(3)")).getText());
assertEquals(this.sNumber, vars.get("sNumber"));
    }
}
```