Instructions & Notes

Initial Setup:

The setup will need to be completed in order to run my tests. There are two ways to run my tests, the first method is to run it off Command Prompt, the second method is to manually create a new Eclipse Project and copy the code over. Please choose whichever method you find easier. If one doesn’t work, please try the other.

Also please note – I’ve uploaded two versions of my code to github. The first is KneatTests.java and uses Junit Annotations to run the tests. The second is KneatTestsMain.java – this uses the standard main method to call the test methods instead of annotations.

*Method 1:*

The ChromeDriver.zip file will be included in the files I have sent. Please create a new folder in your C: drive called ‘Chromedriver’ and copy/unzip ChromeDriver.exe to this location so that the address of the file is ‘C:\Chromedriver\chromedriver.exe’.

Or:

*Method 2:*

Wherever you install/unzip ChromeDriver to, please copy the path of this folder into my test file.

This is referenced on line 43 of my KneatTests java file:

System.setProperty("webdriver.chrome.driver", "C:\\Chromedriver\\chromedriver.exe");

The above C drive location – please change that to wherever you copy the Chrome driver to.

This method will prevent my jar file from running correctly, in which case, please jump to ‘Method 2 – Creating a new project and copying the code over’.

Method 1 – Running the JAR File:

After testing my code on different machines and witnessing the difficulties experienced in installing/setting up the tests, I have written the following steps to make the process of running my tests as easy as possible.

I created a runnable jar file that will run without issue (as long as you created the folder in the C: drive as mentioned in ‘Initial Setup Method 1’ above). Copy the jar file to a folder on your machine.

Open the Command Prompt (cmd) and change directory to where you copied my jar file to and enter the following command:

java -jar KneatTest.jar

That should run my tests without issue.

\*Note – please don’t start clicking things or selecting options in the test window while the tests are running. Scrolling up and down is fine though.

Method 2 – Creating a new project and copying the code over:

The following steps will ensure my tests can be run on Eclipse, though this will involve creating a new Maven Project in Eclipse and modifying the POM file slightly.

1. In Eclipse, go to File → New → Project → Maven Project → Next → Create simple Project & Use default workspace location (tick the boxes).

2. Enter ‘KneatProject’ into both Group ID and Artifact ID. Then hit ‘Finish’.

3. Open the pom.xml file and add all the code from my ‘Dependencies Code’ file after the ‘</version>’ tag and save.

4. Right-click on the src/main/java folder and select New → Package → Enter the name ‘kneatProject’ and Finish.

5. Right-click on the package you just created → New → Class → Name → KneatTests → Finish.

6. Highlight everything in the new class you just created, copy and paste all of my code from KneatTests.java.

7. Modify the Chrome driver location code as mentioned in the ‘Setup’ section and save.

8. Right-click on the KneatTests.java file in the Project view, or in the code itself, and select Run As → Junit Test.

(Optional) to run my alternate version of my code - KneatTestsMain, repeat steps 5 – 8 except use the name KneatTestsMain. To run it, select Run As → Java Application.

\*Note – please don’t start clicking things or selecting options in the test window while the tests are running. Scrolling up and down is fine though.

Walkthrough & Notes:

Before I began writing the code for these tests, I first navigated to Booking.com and performed the instructions manually. This gave me a rough idea of how many different elements on the webpage needed to be manipulated and what that may involve. Below are the steps I took to enter all the required information before coming to the results page containing all the filters.

I right-clicked while on the Booking.com page and selected ‘inspect’ to display the debug console and view the HTML tags and to locate the exact piece of code needed for every element that I used. The first thing I did was find the location/’Where are you going?’ box element - just driver.findElement(By.*id*("ss")); - and send the word ‘Limerick’ to it.

The second thing I did was write a piece of code to accept and dismiss that annoying cookie popup that was taking up a quarter of the page.

The ‘Check-In’ box was a bit more difficult. It would have been straightforward enough to just use the Calendar class in java, but I wanted to try and use Selenium as much as I could for this program. I found today’s date on the calendar by using ‘By.*cssSelector*("td[class='bui-calendar\_\_date bui-calendar\_\_date—today']")’ - this would always bring me to the element that was highlighted for today’s date. To return the full date, I had to use ‘getAttribute’.

I had to manipulate the date carefully. The format was ‘2020-07-16’ and even increasing the month by 3 (3 months from today) required me convert the substring ‘07’ to an int which became ‘7’, add 3, convert it back to a string and add it back to the date format. To further complicate matters, I had to perform checks on whether the month’s value was less than 7 and if so, to add on a ‘0’. This may sound pedantic, but the date needed to be ‘2020-07-16’ and not ‘2020-7-16’ which wouldn’t work. Also, if the current month was October, November or December (10, 11 or 12) I would have to roll the month into next year to become January, February or March – which also affected the ‘year’ part of the date.

The only part of the Calendar choosing function I haven’t coded is if the current date is the 31st and 3 months from now the last day of the month is the 30th – and don’t even get me started on February.

The 2 adults 1 room option was already preselected so I didn’t need to manipulate it, after that I just needed to click the search button.

I have written a number of tests for this challenge. The two mandatory ones are fiveStarFilter and saunaFilter.

I created an extra three tests to test more of the filters as well as test the flight booking option.

I have written assertions that will check whether some hotels are visible or not after filtering the results, but please take note of the bug I found on the Booking.com website.

Bug:

\*I did notice one thing - a bug on the Booking.com website. When running Chrome in the test window or incognito mode (which is similar to the test browser window), one of the filter links did not work properly. As shown in the attached screenshot ‘Discrepancy 2’, sometimes the ‘Show more’ link will be replaced by ‘Show all 13’ - this seems to occur intermittently and at random.

It is a requirement of the ‘Kneat Automation Code Challenge’ to select the ‘Sauna’ filter and check the resulting list for specific hotel names. This cannot be accomplished if the ‘Show all 13’ link has loaded as none of the 13 filter options presented are ‘Sauna’.

The ‘Sauna’ option does sometimes appear under the ‘Fun things to do’ filter heading, but this is also intermittent and at random.

For more detail on this bug – please see the Bug Report I wrote.

I have altered my program to detect the broken ‘Show all 13’ filter and to immediately close the test with a message.

If the ‘Show more’ link loads instead, my program should also detect this and click on it – and my test will continue.

I can only ask that whoever is testing my program to please re-run the ‘saunaFilter’ test a few times – until the ‘Show more’ link appears and the test works as expected.