Кратко ръководство за създаване на автоматизирани тестове със Selenium WebDriver, Eclipse, Java и TestNG

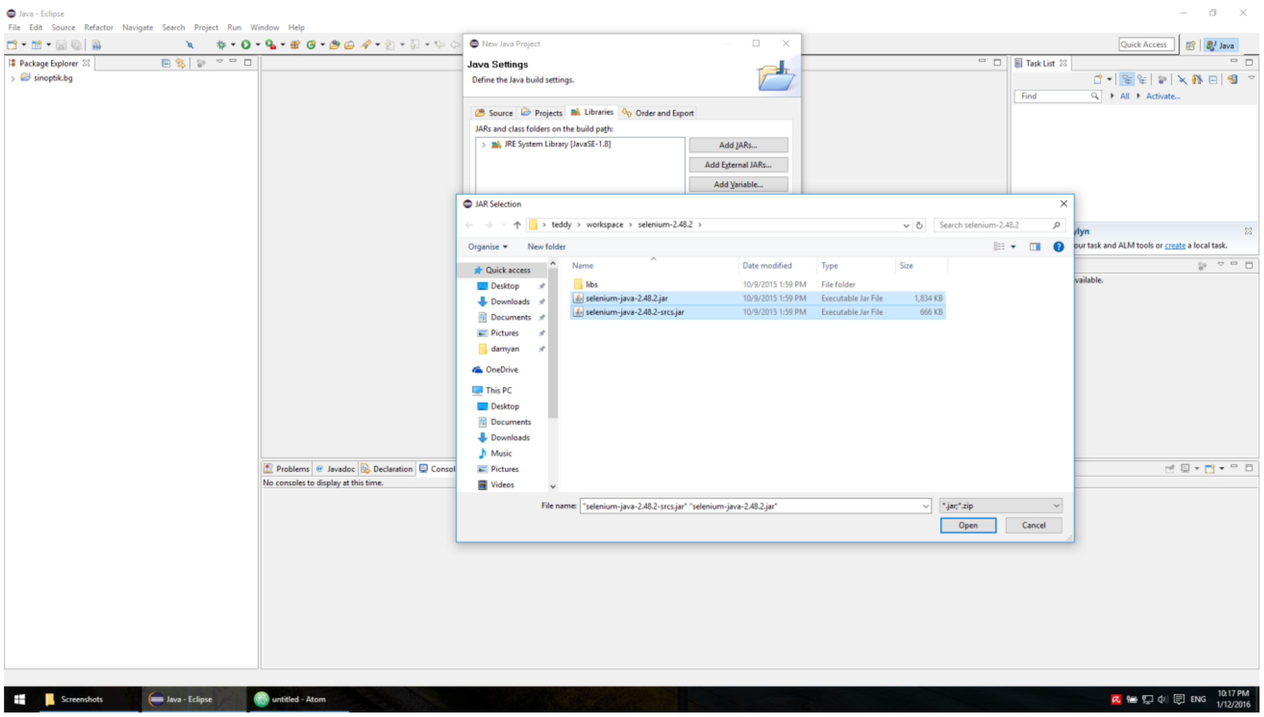
Това е кратко ръководство, което описва стъпките за създаване на примерни автоматизирани тестове с Selenium WebDriver, Eclipse, Java и TestNG, които бяха показани по време на бързото демо на 14 януари. Избрали сме да използваме сайта sinoptik.bg

За да можем да продължим напред трябва да имате инсталирани Java и Eclipse. Не би трябвало да има проблем от версиите, които ще се използват. Ние сме използвали последните актуални.

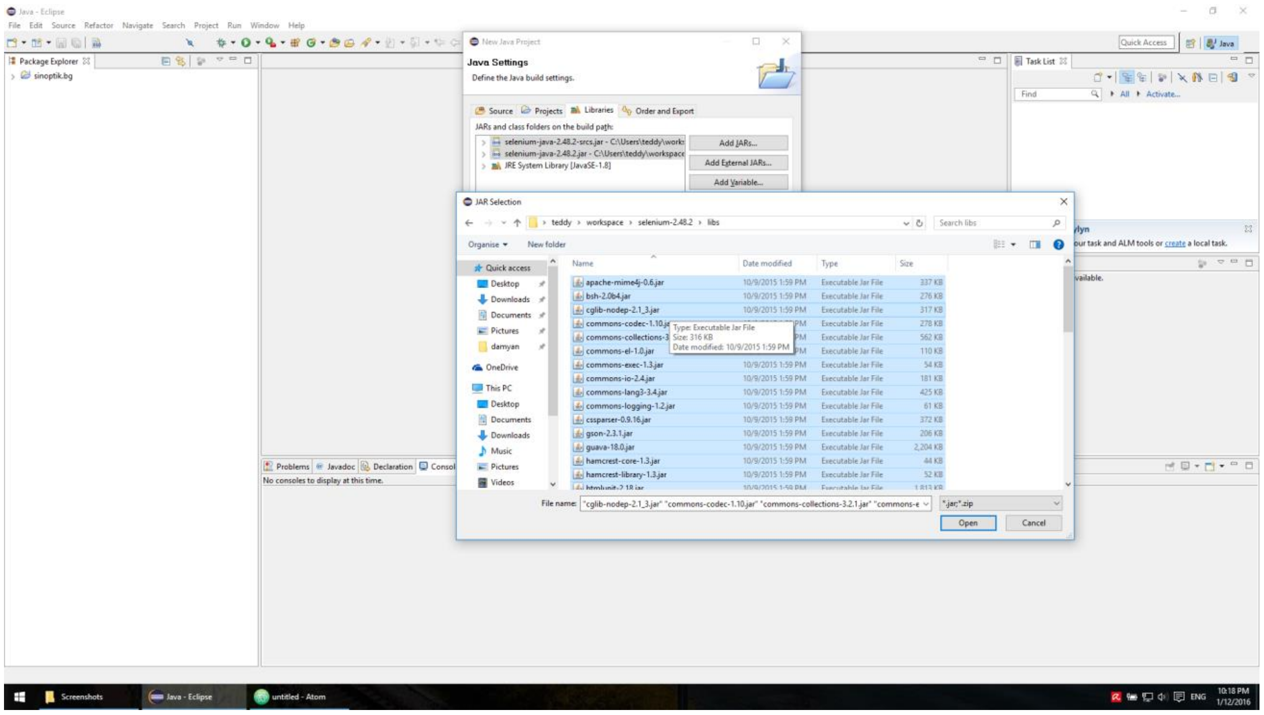
Също така ни трябват и Selenium и TestNG. За Selenium трябва да си свалите selenium-java пакет, което може да стане от тук - http://selenium-release.storage.googleapis.com/2.48/selenium-java-2.48.2.zip, като е необходимо само да си го разархивирате някъде.

На download страницата на TestNG е описано как да си инсталираме приставката(plugin) за Eclipse - http://testng.org/doc/download.html

Следващата стъпка е да си направим в Eclipse нов Java проект. Ние сме избрали името sinoptik.bg. Тук много важно е при създаването на проекта да добавим необходимите библиотеки. На 3тия таб на екрана Libraries избираме Add External JARs и добавяме 2та jar файла от директорията на Selenium и всички jar файлове от поддиректорията libs.



File name: "selenium-java-2.48.2-srcs.jar" "selenium-java-2.48.2.jar"



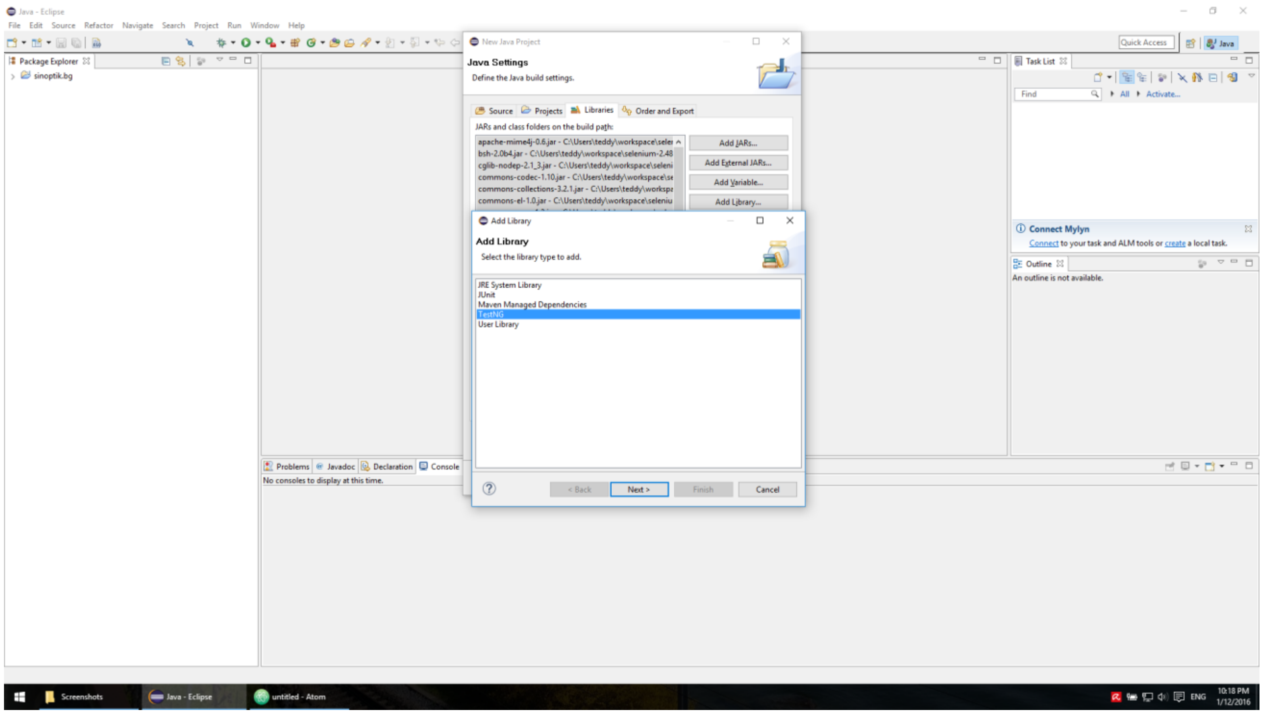
JARs and class folders on the build path:
- selenium-java-2.48.2-srcs.jar - C:\Users\teddy\work
- selenium-java-2.48.2.jar - C:\Users\teddy\workspace
- JRE System Library [JavaSE-1.8]

Name	Date modified	Type	Size
apache-mime4j-0.6.jar	10/9/2015 1:59 PM	Executable Jar File	337 KB
bsh-2.0b4.jar	10/9/2015 1:59 PM	Executable Jar File	276 KB
cglib-nodep-2.1_3.jar	10/9/2015 1:59 PM	Executable Jar File	317 KB
commons-codec-1.10.jar	10/9/2015 1:59 PM	Executable Jar File	278 KB
commons-collections-3	10/9/2015 1:59 PM	Executable Jar File	562 KB
commons-el-1.0.jar	10/9/2015 1:59 PM	Executable Jar File	110 KB
commons-exec-1.3.jar	10/9/2015 1:59 PM	Executable Jar File	54 KB
commons-io-2.4.jar	10/9/2015 1:59 PM	Executable Jar File	181 KB
commons-lang3-3.4.jar	10/9/2015 1:59 PM	Executable Jar File	425 KB
commons-logging-1.2.jar	10/9/2015 1:59 PM	Executable Jar File	61 KB
cssparser-0.9.16.jar	10/9/2015 1:59 PM	Executable Jar File	372 KB
gson-2.3.1.jar	10/9/2015 1:59 PM	Executable Jar File	206 KB
guava-18.0.jar	10/9/2015 1:59 PM	Executable Jar File	2,204 KB
hamcrest-core-1.3.jar	10/9/2015 1:59 PM	Executable Jar File	44 KB
hamcrest-library-1.3.jar	10/9/2015 1:59 PM	Executable Jar File	52 KB
htmlunit-2.18.jar	10/9/2015 1:59 PM	Executable Jar File	1,813 KB

File name: "cglib-nodep-2.1_3.jar" "commons-codec-1.10.jar" "commons-collections-3.2.1.jar" "commons-e

.jar;.zip

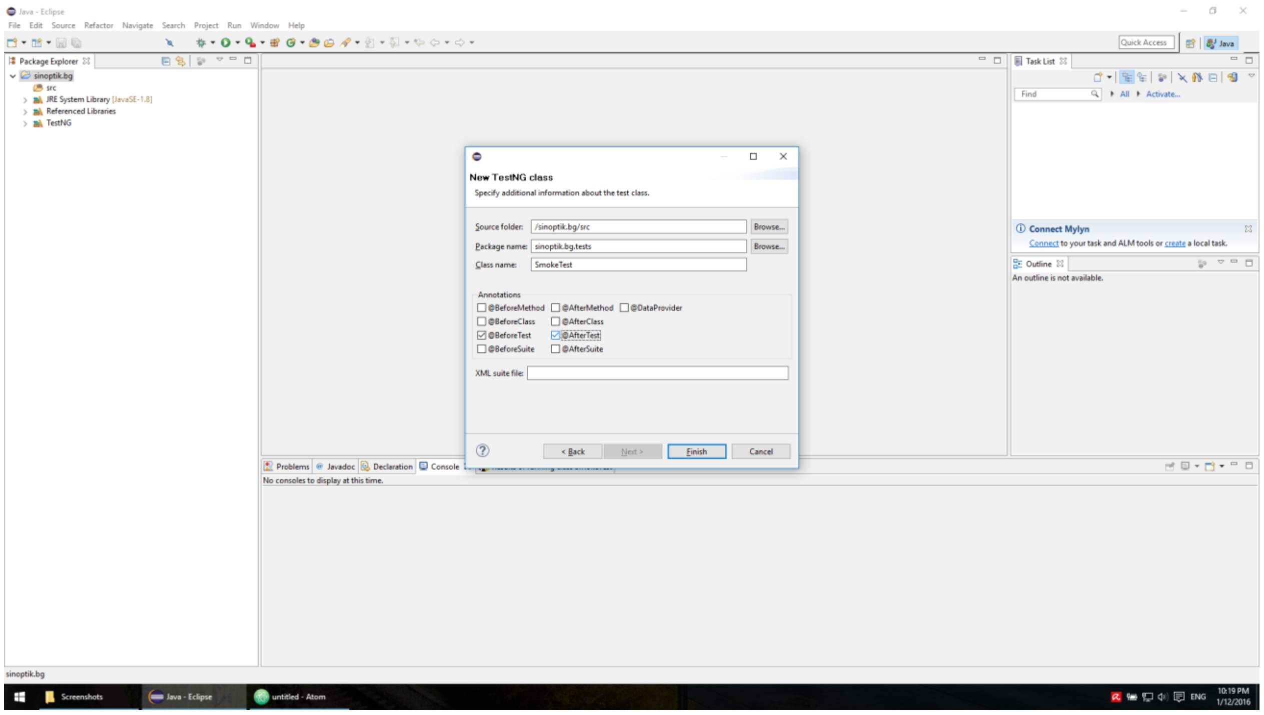
След като направим това избираме и опцията Add Library и добаваме TestNG от списъка.



JARs and class folders on the build path:
- apache-mime4j-0.6.jar - C:\Users\teddy\workspace\sele
- bsh-2.0b4.jar - C:\Users\teddy\workspace\selenium-2.48
- cglib-nodep-2.1_3.jar - C:\Users\teddy\workspace\seleni
- commons-codec-1.10.jar - C:\Users\teddy\workspace\se
- commons-collections-3.2.1.jar - C:\Users\teddy\workspa
- commons-el-1.0.jar - C:\Users\teddy\workspace\seleniu

- JRE System Library
- JUnit
- Maven Managed Dependencies
- TestNG
- User Library

За да продължим трябва да си създадем и TestNG клас. Това става от File->New->Other. Избираме TestNG клас и ни се отваря нов прозорец. На следващата снимка се вижда как да го попълним.


New TestNG class
Specify additional information about the test class.

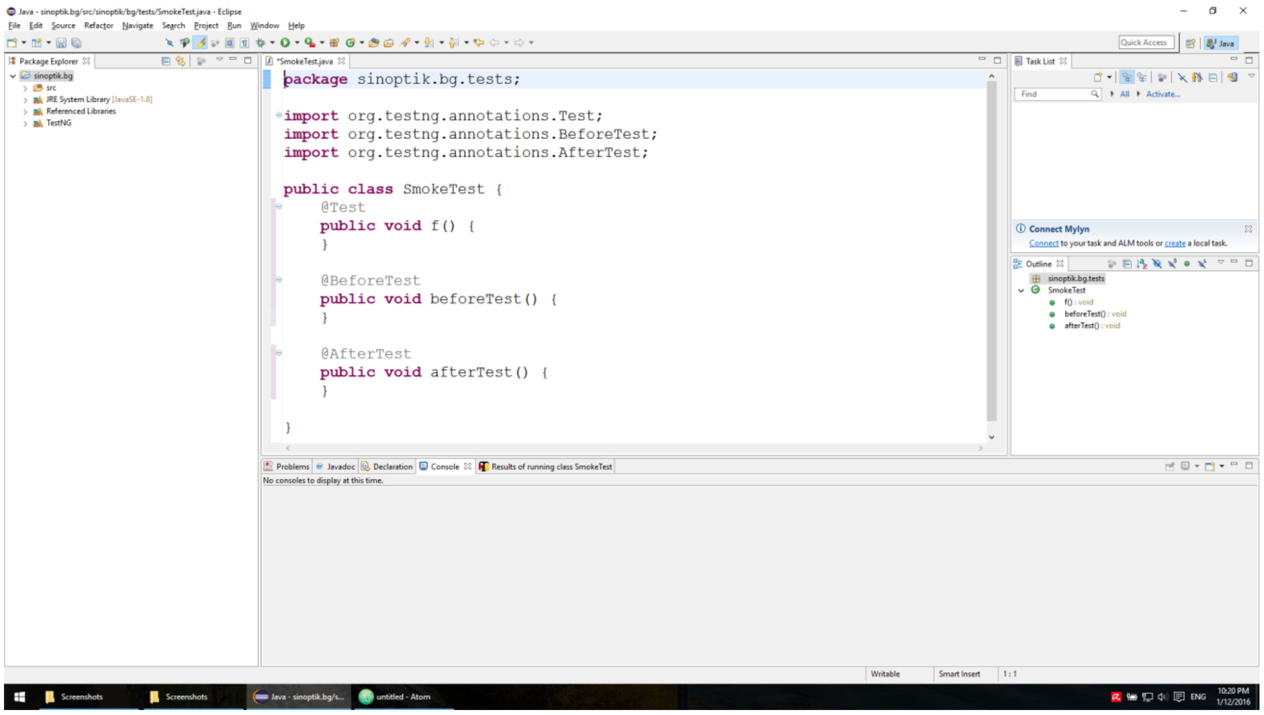
Source folder: /sinoptik.bg/src
Package name: sinoptik.bg.tests
Class name: SmokeTest

Annotations
- [ ] @BeforeMethod
- [ ] @AfterMethod
- [ ] @DataProvider
- [ ] @BeforeClass
- [ ] @AfterClass
- [x] @BeforeTest
- [x] @AfterTest
- [ ] @BeforeSuite
- [ ] @AfterSuite

XML suite file:

No consoles to display at this time.

След тази стъпка трабва да имаме подобен на следния екран


```java
package sinoptik.bg.tests;

import org.testng.annotations.Test;
import org.testng.annotations.BeforeTest;
import org.testng.annotations.AfterTest;

public class SmokeTest {
    @Test
    public void f() {
    }

    @BeforeTest
    public void beforeTest() {
    }

    @AfterTest
    public void afterTest() {
    }

}
```

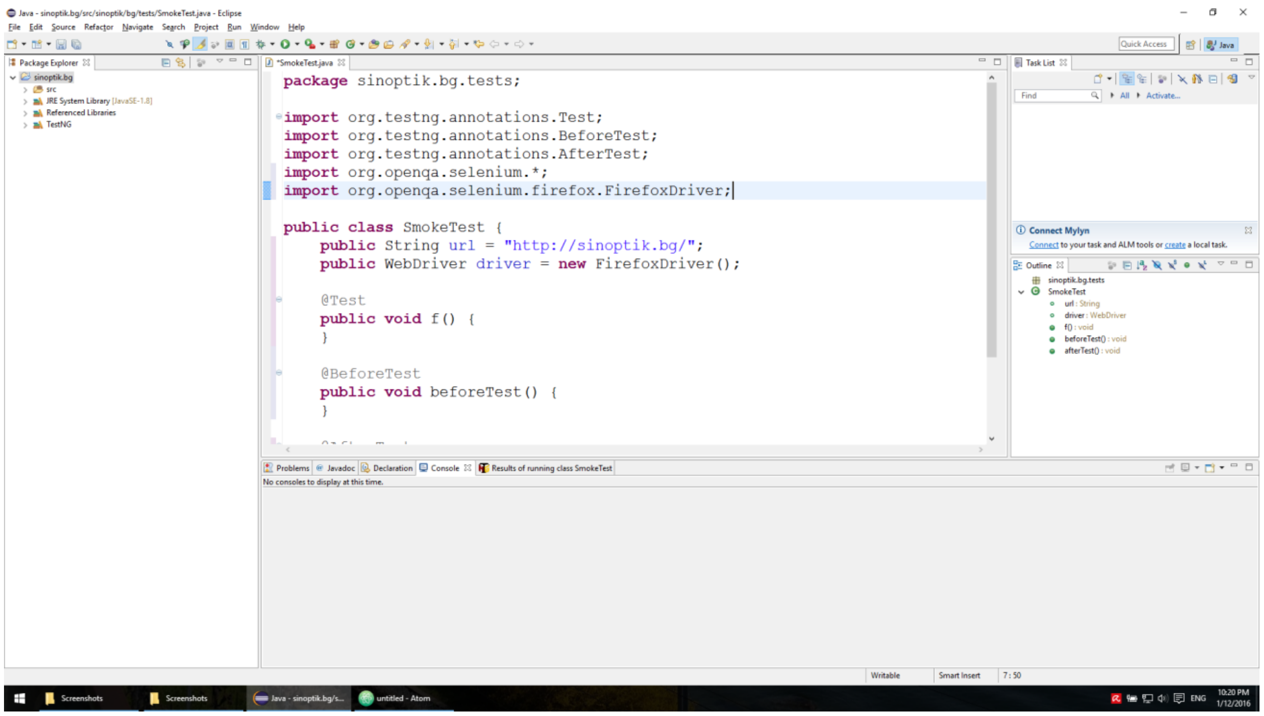
No consoles to display at this time.

Време е за истинската работа и да направим нашите тестове. Ще направим общо 2. Единия да отворим картата на България и оттам да изберем град и втори, в които търсим град в полето за търсене и също отваряме неговата страница с прогнози.

За да можем да стартираме Firefox чрез нашите тестове трябва да добавим необходимите библиотеки в нашия клас. Добавяме следните 2 реда на мястото в нашия клас, където се описват външните библиотеки.

```
import org.openqa.selenium.*;

import org.openqa.selenium.firefox.FirefoxDriver;
```

След това в началото на нашия клас декларираме 2 променливи. Едната да съдържа url на тествания сайт и друга, която да съдържа нашия WebDriver, изпълняващ тестовете.



```
package sinoptik.bg.tests;

import org.testng.annotations.Test;
import org.testng.annotations.BeforeTest;
import org.testng.annotations.AfterTest;
import org.openqa.selenium.*;
import org.openqa.selenium.firefox.FirefoxDriver;

public class SmokeTest {
    public String url = "http://sinoptik.bg/";
    public WebDriver driver = new FirefoxDriver();

    @Test
    public void f() {
    }

    @BeforeTest
    public void beforeTest() {
    }
```

```
No consoles to display at this time.
```

Пред началото на всеки тест искаме сайтът да ни се отваря. Затова в метода beforeTest добавяме следния код

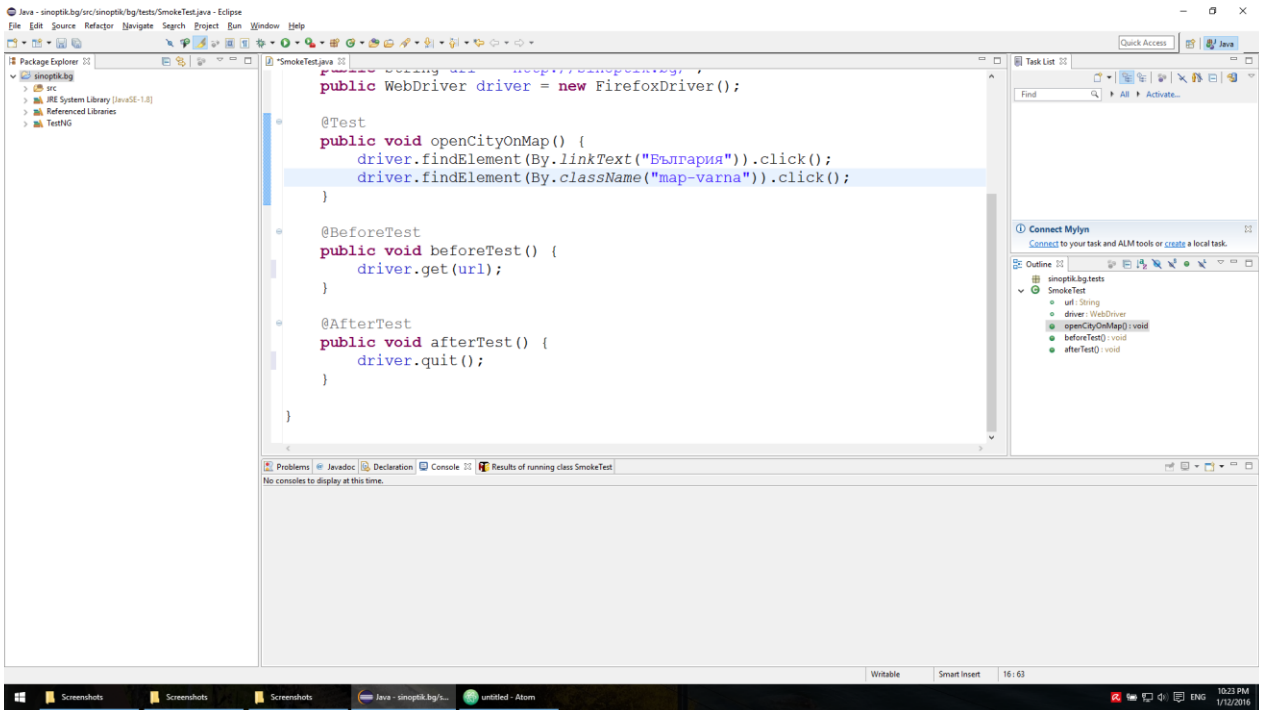
```
driver.get(url);
```

Респективно искаме и след всеки тест да приключваме изпълнението. Това става като добавим следния код в метода afterTest.

```
driver.quit();
```

Сега локализираме линка към картата на България и след това Варна. Кодът за тези действия е следният:

```
driver.findElement(By.linkText("България")).click();

driver.findElement(By.className("map-varna")).click();
```

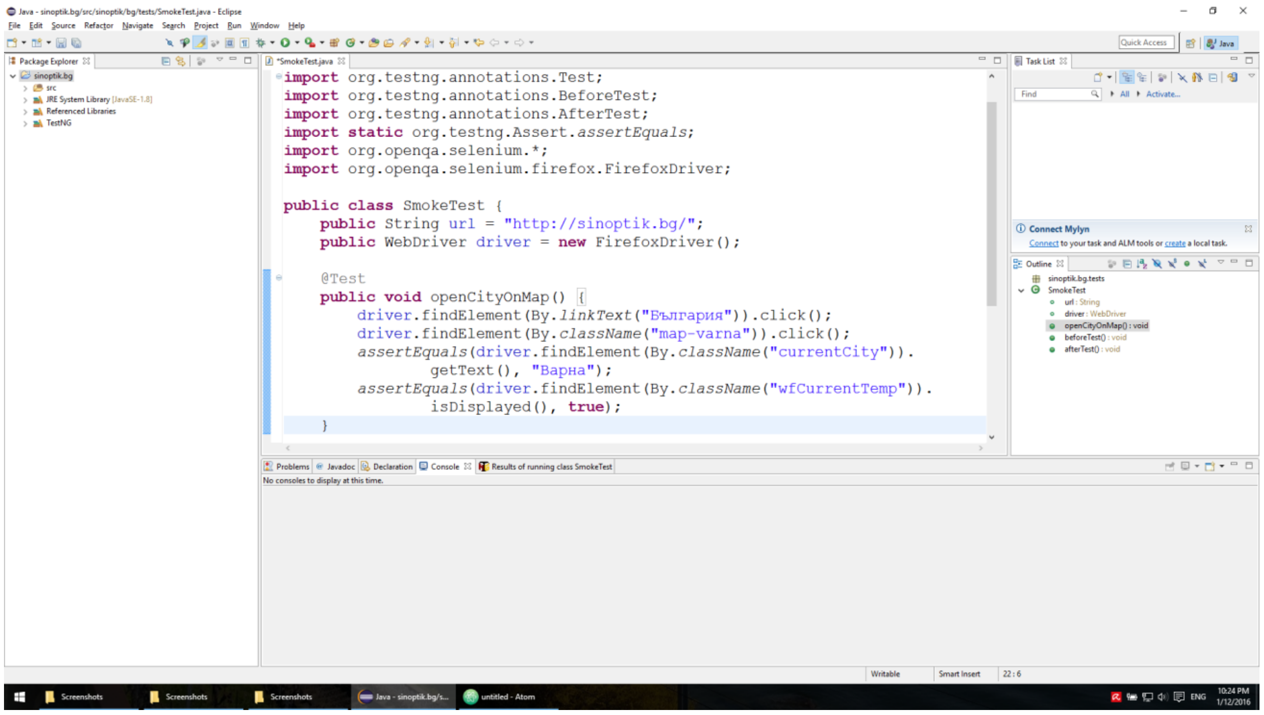


```java
    public WebDriver driver = new FirefoxDriver();

    @Test
    public void openCityOnMap() {
        driver.findElement(By.linkText("България")).click();
        driver.findElement(By.className("map-varna")).click();
    }

    @BeforeTest
    public void beforeTest() {
        driver.get(url);
    }

    @AfterTest
    public void afterTest() {
        driver.quit();
    }

}
```

След като сме отворили страницата с прогнозата за Варна, правим две проверки – дали пише Варна на мястото, където се изписва името на населеното място и дали е показана температурата.

Проверките правим със следния код:

```
assertEquals(driver.findElement(By.className("currentCity")).getText(), "Варна");

assertEquals(driver.findElement(By.className("wfCurrentTemp")).isDisplayed(), true);
```



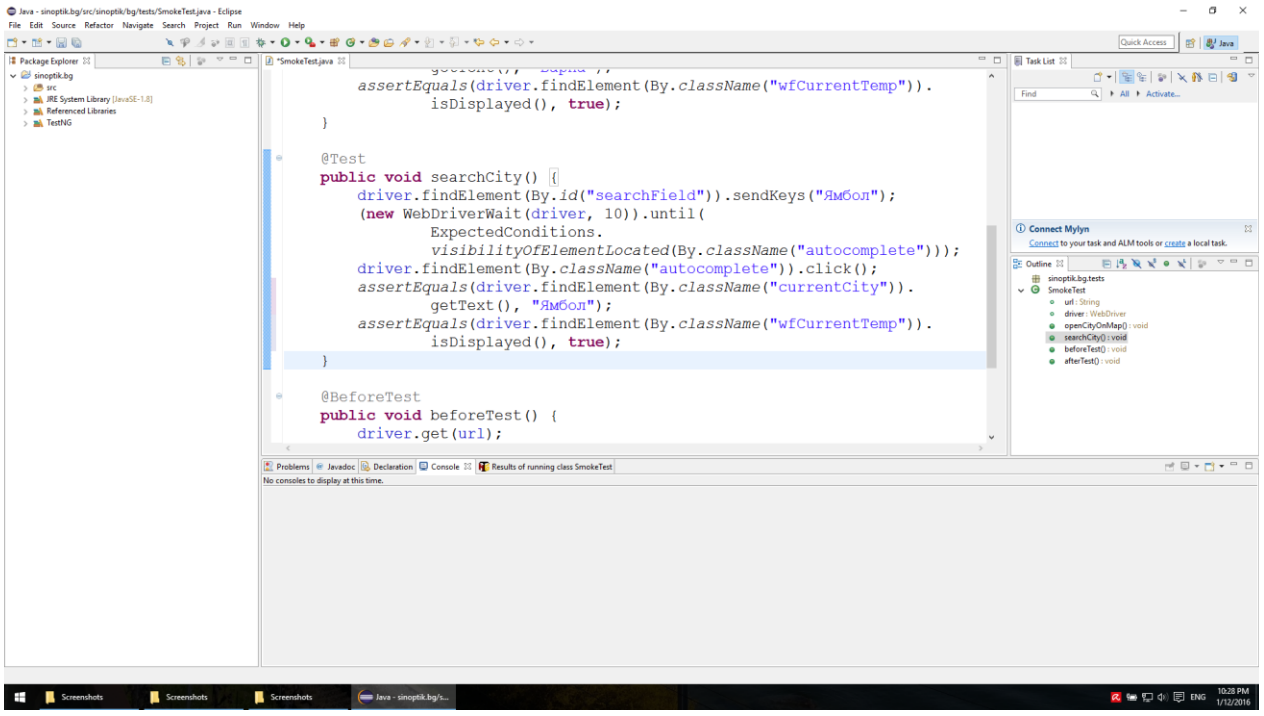
```java
import org.testng.annotations.Test;
import org.testng.annotations.BeforeTest;
import org.testng.annotations.AfterTest;
import static org.testng.Assert.assertEquals;
import org.openqa.selenium.*;
import org.openqa.selenium.firefox.FirefoxDriver;

public class SmokeTest {
    public String url = "http://sinoptik.bg/";
    public WebDriver driver = new FirefoxDriver();

    @Test
    public void openCityOnMap() {
        driver.findElement(By.linkText("България")).click();
        driver.findElement(By.className("map-varna")).click();
        assertEquals(driver.findElement(By.className("currentCity")).
                getText(), "Варна");
        assertEquals(driver.findElement(By.className("wfCurrentTemp")).
                isDisplayed(), true);
    }
```

За следващия тест, проверките които правим са същите. Разликата е, че отваряме страницата с времето на населено място през функционалността за търсене на сайта, като изчакваме на се появи мястото в autocomplete списъка и кликваме върху него. За тази цел локализираме елементът от страницата за търсене и попълваме името на града, който ще търсим – Ямбол.

Тъй като е възможно да имаме проблеми с интернета трябва след въвеждането на населено място в полето за търсене да изчакаме то да се появи в списъка:

```
(new WebDriverWait(driver, 10)).until(ExpectedConditions.
        visibilityOfElementLocated(By.className("autocomplete")));
```

Ако не добавим този код е възможно тестът ни понякога да не минава успешно, защото няма да може да намери елементът, които очакваме – елемент с клас autocomplete.



```
        assertEquals(driver.findElement(By.className("wfCurrentTemp")).
                isDisplayed(), true);
    }

    @Test
    public void searchCity() {
        driver.findElement(By.id("searchField")).sendKeys("Ямбол");
        (new WebDriverWait(driver, 10)).until(
                ExpectedConditions.
                visibilityOfElementLocated(By.className("autocomplete")));
        driver.findElement(By.className("autocomplete")).click();
        assertEquals(driver.findElement(By.className("currentCity")).
                getText(), "Ямбол");
        assertEquals(driver.findElement(By.className("wfCurrentTemp")).
                isDisplayed(), true);
    }

    @BeforeTest
    public void beforeTest() {
        driver.get(url);
```

No consoles to display at this time.