ssertions

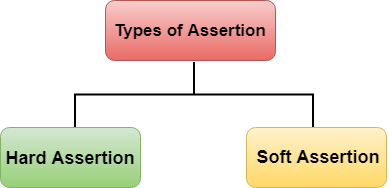
Assertion determines the state of the application whether it is the same what we are expecting or not. If the assertion fails, then the test case is failed and stops the execution.

To use the Assertion in Web Driver, you need to download the Testng jar file and add it to the eclipse. Download the Testng jar file from the link given below:

<https://mvnrepository.com/artifact/org.testng/testng/6.7>

**There are two types of Assertion:**

* Hard Assertion
* Soft Assertion



Hard Assertion

Hard Assertion is an Assertion that throws the **AssertException** when the test case is failed. In the case of Hard Assertion, you can handle the error by using a catch block like a java exception. Suppose we have two test cases in a suite. The first test case in a suite has an assertion that fails, and if we want to run the second case in a suit, then we need to handle the assertion error. A Hard Assertion contains the following methods:

* AssertEquals
* AssertNotEquals
* AssertTrue
* AssertFalse
* AssertNull
* AssertNotNull

AssertFalse()

Assertion verifies the boolean value returned by a condition. If the boolean value is false, then assertion passes the test case, and if the boolean value is true, then assertion aborts the test case by an exception. Syntax of AssertFalse() method is given below:

1. Assert.AssertFalse(condition);

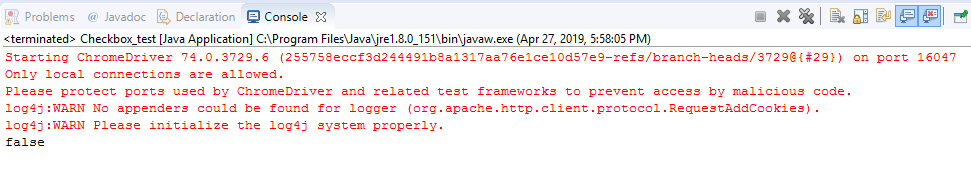
**Let's understand through an example:**

* **When the condition is false**

1. **package** mypack;
2. **import** org.junit.Assert;
3. **import** org.openqa.selenium.By;
4. **import** org.openqa.selenium.WebDriver;
5. **import** org.openqa.selenium.chrome.ChromeDriver;
6. **public** **class** Checkbox\_test {
8. **public** **static** **void** main(String[] args) {
9. // TODO Auto-generated method stub
10. System.setProperty("webdriver.chrome.driver","C:\\\\work\\\\chromedriver.exe");
11. WebDriver driver = **new** ChromeDriver();
12. driver.navigate().to("https://www.spicejet.com/");
13. Assert.assertFalse(driver.findElement(By.cssSelector("input[id\*='SeniorCitizenDiscount']")).isSelected());
14. System.out.println(driver.findElement(By.cssSelector("input[id\*='SeniorCitizenDiscount']")).isSelected());
16. }
18. }

In the above code, Assert.assertFalse() contains the condition which is returning false value. Therefore, it passes the test case.

**Output on the console**

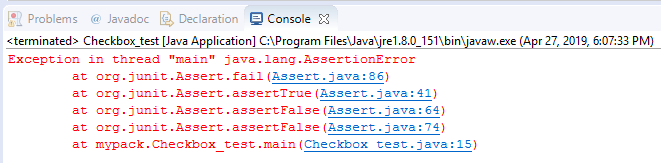


* **When the condition is true**

1. **package** mypack;
2. **import** org.junit.Assert;
3. **import** org.openqa.selenium.By;
4. **import** org.openqa.selenium.WebDriver;
5. **import** org.openqa.selenium.chrome.ChromeDriver;
6. **public** **class** Checkbox\_test {
8. **public** **static** **void** main(String[] args)
9. {
10. // TODO Auto-generated method stub
11. System.setProperty("webdriver.chrome.driver","C:\\\\work\\\\chromedriver.exe");
12. WebDriver driver = **new** ChromeDriver();
13. driver.navigate().to("https://www.spicejet.com/");
14. Assert.assertFalse(**true**);
15. System.out.println(driver.findElement(By.cssSelector("input[id\*='SeniorCitizenDiscount']")).isSelected());
17. }}

In the above code, Assert.assertFalse() method contains the true condition. Therefore, the assertion fails which means that the test case is also failed. Assertion failure will stop the execution by exception.

**Output on the console**



AssertTrue()

Assertion verifies the boolean value returned by a condition. If the boolean value is true, then assertion passes the test case, and if the boolean value is false, then assertion aborts the test case by an exception. Syntax of AssertTrue() method is given below:

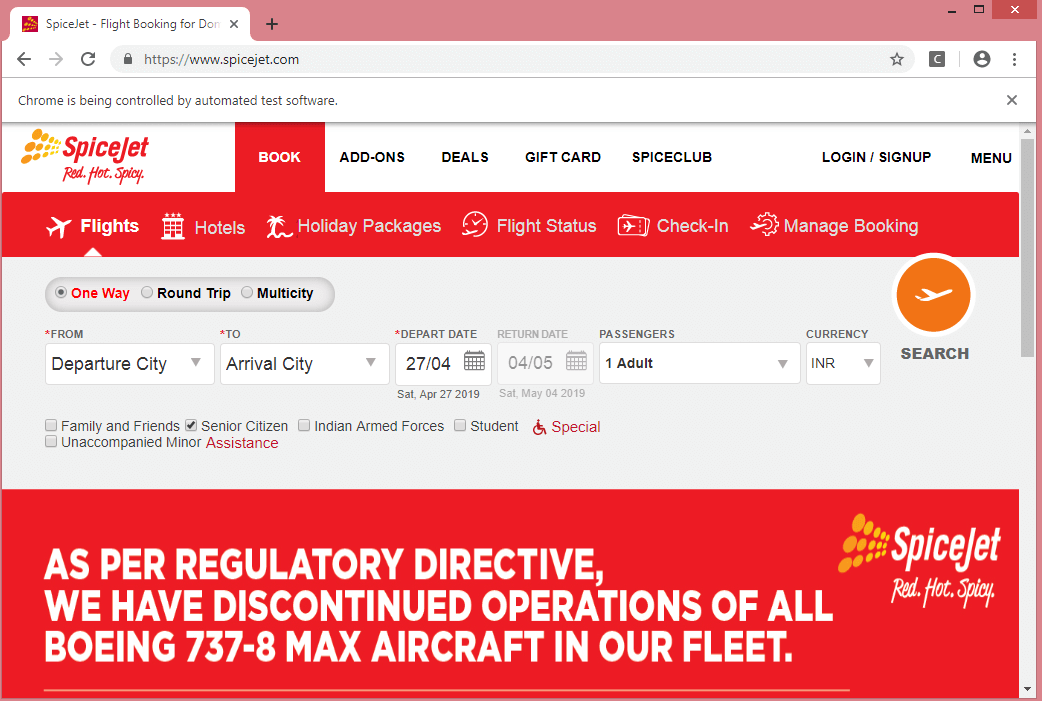
1. Assert.AssertTrue(condition);

**Let's understand through an example.**

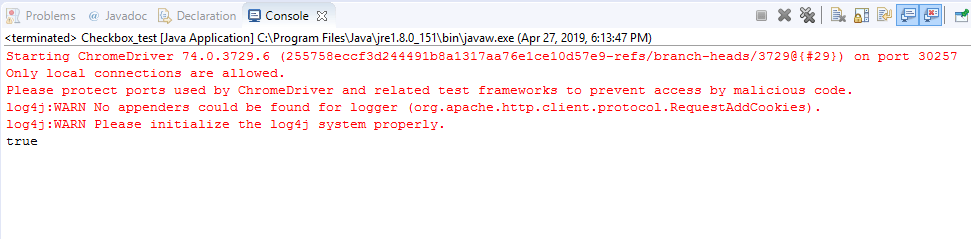
1. **package** mypack;
2. **import** org.junit.Assert;
3. **import** org.openqa.selenium.By;
4. **import** org.openqa.selenium.WebDriver;
5. **import** org.openqa.selenium.chrome.ChromeDriver;
6. **public** **class** Checkbox\_test
7. {
9. **public** **static** **void** main(String[] args)
10. {
11. // TODO Auto-generated method stub
12. System.setProperty("webdriver.chrome.driver","C:\\\\work\\\\chromedriver.exe");
13. WebDriver driver = **new** ChromeDriver();
14. driver.navigate().to("https://www.spicejet.com/");
15. driver.findElement(By.cssSelector("input[id\*='SeniorCitizenDiscount']")).click();
16. Assert.assertTrue(driver.findElement(By.cssSelector("input[id\*='SeniorCitizenDiscount']")).isSelected());
17. System.out.println(driver.findElement(By.cssSelector("input[id\*='SeniorCitizenDiscount']")).isSelected());
19. }
21. }

In the above code, **driver.findElement(By.cssSelector("input[id\*='SeniorCitizenDiscount']")).click();** This statement is used to select the 'Senior Citizen' box. In the next statement, we are applying assertion to check whether the test case fails or pass. The parameter inside the Assert.assertTrue() method is returning true value, therefore the test case pass.

**Output**



**Output on the Console**



AssertEquals()

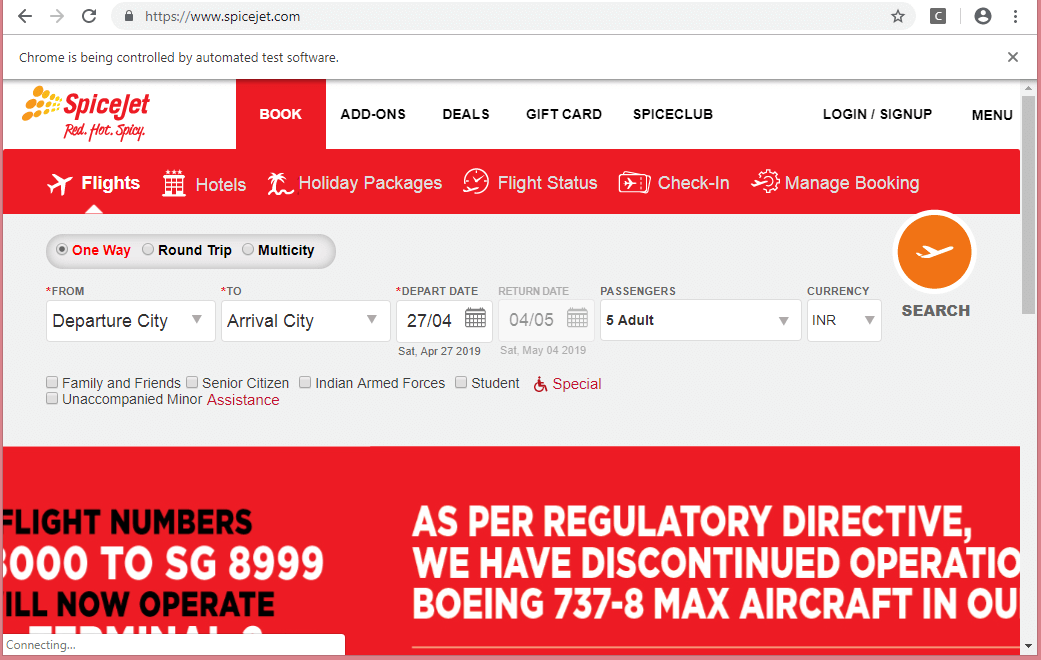
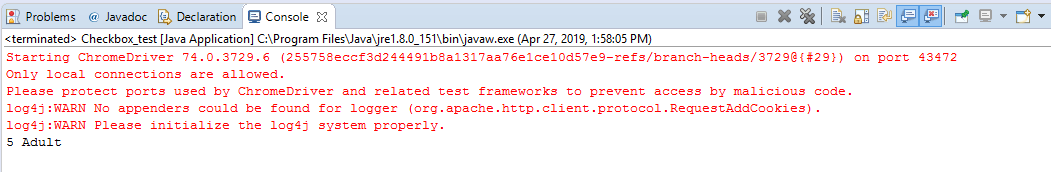
AssertEquals() is a method used to compare the actual and expected results. If both the actual and expected results are same, then the assertion pass with no exception and the test case is marked as "passed". If both the actual and expected results are not the same, then the assertion fails with an exception and the test case is marked as "failed". Syntax of an AssertEquals() method is given below:

1. Assert.assertEquals(actual,expected);

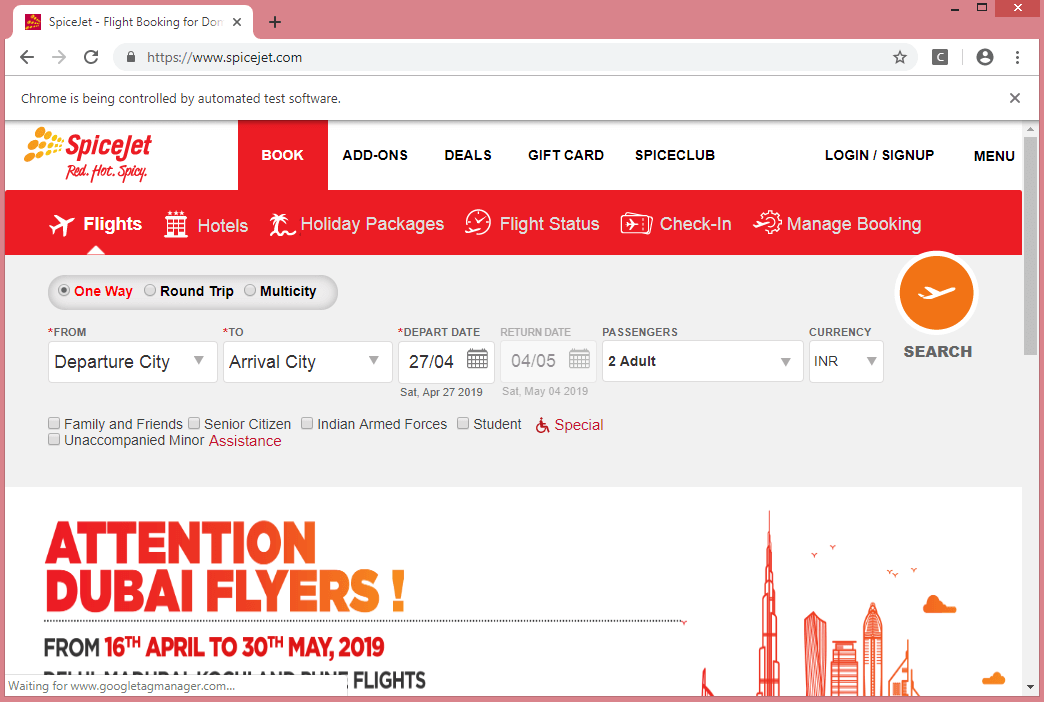
**Let's understand through an example.**

* **When a number of adults is 5.**

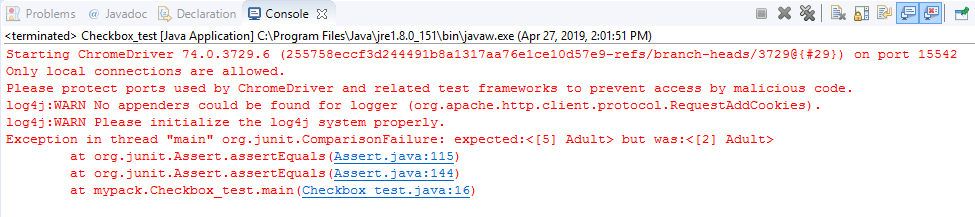
1. **package** mypack;
2. **import** org.junit.Assert;
3. **import** org.openqa.selenium.By;
4. **import** org.openqa.selenium.WebDriver;
5. **import** org.openqa.selenium.chrome.ChromeDriver;
6. **public** **class** Checkbox\_test {
7. **public** **static** **void** main(String[] args)
8. {
9. // TODO Auto-generated method stub
10. System.setProperty("webdriver.chrome.driver","C:\\\\work\\\\chromedriver.exe");
11. WebDriver driver = **new** ChromeDriver();
12. driver.navigate().to("https://www.spicejet.com/"); Assert.assertEquals("5Adult",driver.findElement(By.id("divpaxinfo")).getText());
13. System.out.println(driver.findElement(By.id("divpaxinfo")).getText());
14. }}

* **When the number of adults is not equal to 5**



**Output on the console**



AssertNotEquals()

It is opposite to the function of **AssertNotEquals()** method. AssertNotEquals() is a method used to compare the actual and expected results. If both the actual and expected results are not the same, then the assertion pass with no exception and the test case is marked as "passed". If both the actual and expected results are same, then the assertion fails with an exception and the test case is marked as "failed". Syntax of an AssertNotEquals() method is given below:

1. AssertNotEquals(actual,expected,message);

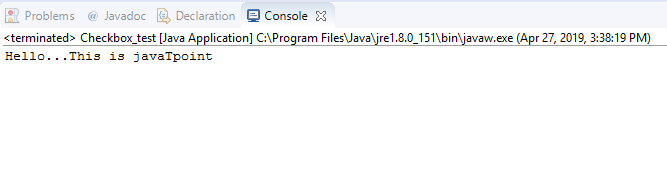
**Let's understand through an example.**

* **When actual string is not equal to the expected string.**

1. **package** mypack;
2. **import** org.junit.Assert;
3. **public** **class** Checkbox\_test {
4. **public** **static** **void** main(String[] args) {
5. // TODO Auto-generated method stub
6. Assert.assertNotEquals("Hello", "How are you");
7. System.out.println("Hello...This is javaTpoint");
9. }
11. }

In the above code, actual string, i.e., Hello is not equal to the expected string, i.e., How are you. Therefore, the assertion passes the test case. This will execute the next statement and the next statement is **System.out.println("Hello...This is javaTpoint");**.

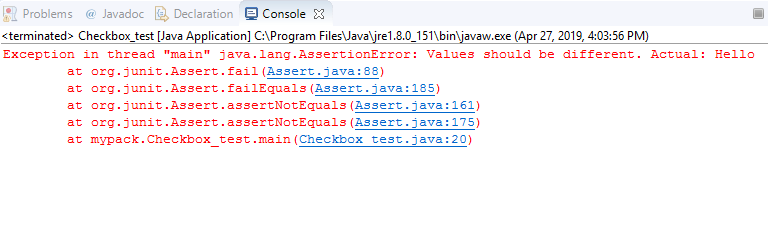
**Output**



* **When the actual string is equal to the expected string.**

1. **package** mypack;
2. **import** org.junit.Assert;
3. **public** **class** Checkbox\_test {
4. **public** **static** **void** main(String[] args)
5. {
6. // TODO Auto-generated method stub
7. Assert.assertNotEquals("Hello", "Hello");
8. System.out.println("Hello...This is javaTpoint");
9. }}

**Output**



AssertNull()

AssertNull() is a method that verifies whether the object is null or not. If an object is null, then assertion passes the test case, and the test case is marked as "passed", and if an object is not null, then assertion aborts the test case and the test case is marked as "failed". Syntax of AssertNull() method is given below:

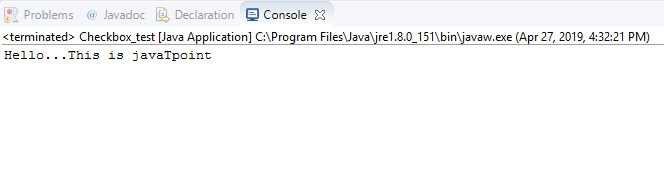
1. Assert.assertNull(object);

**Let's understand through an example.**

* **When an object is null.**

1. **package** mypack;
2. **import** org.junit.Assert;
3. **public** **class** Checkbox\_test {
5. **public** **static** **void** main(String[] args) {
7. Assert.assertNull(**null**);
8. System.out.println("Hello...This is javaTpoint");
9. }}

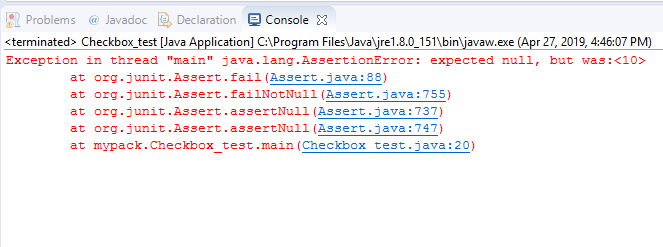
**Output**



* **When an object is not equal to null.**

1. **package** mypack;
2. **import** org.junit.Assert;
3. **public** **class** Checkbox\_test {
5. **public** **static** **void** main(String[] args) {
6. // TODO Auto-generated method stub
7. Assert.assertNull(10);
8. System.out.println("Hello World");
10. }
12. }

**Output**



AssertNotNull()

AssertNotNull() is a method that verifies whether the object is null or not. If an object is not null, then assertion passes the test case and test case is marked as "passed", and if an object is null, then assertion aborts the test case and test case is marked as "failed". Syntax of AssertNotNull() method is given below:

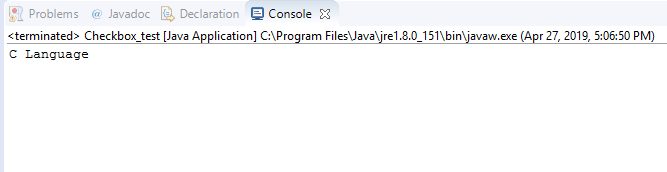
1. Assert.assertNotNull(object);

**Let's understand through an example.**

* **When an object is not null.**

1. **package** mypack;
2. **import** org.junit.Assert;
3. **public** **class** Checkbox\_test
4. {
5. **public** **static** **void** main(String[] args) {
6. // TODO Auto-generated method stub
7. Assert.assertNotNull(10);
8. System.out.println("C Language");
10. }}

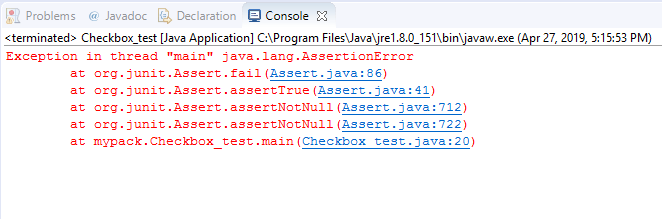
**Output**



* **When an object is null.**

1. **package** mypack;
2. **import** org.junit.Assert;
3. **public** **class** Checkbox\_test {
4. **public** **static** **void** main(String[] args) {
5. // TODO Auto-generated method stub
7. Assert.assertNotNull(**null**);
8. System.out.println("C Language");
10. }
12. }

**Output**



SoftAssertion

Till now, we have learnt about the Hard Assertion in Web Driver using Testng framework. In hard assertion, if an assertion fails then it aborts the test case otherwise it continues the execution. Sometimes we want to execute the whole script even if the assertion fails. This is not possible in Hard Assertion. To overcome this problem, we need to use a soft assertion in testng.