Junit Annotations
+++++++++++++++++
1. @Test
2. @DisplayName
3. @Order
4. @Disabled
5. @Tag(Can be used to run testcases through Junit and also through Maven life cycle)
6. @TestMethodOrder(value=..../.../...)
7. @BeforeEach
8. @BeforeAll[setUpCode() :: public static]
9. @AfterEach
10. @AfterAll[cleanUpCode():: public static]
11. @RepeatedTest(value=int,name="")
12. @ParameterizedTest(...)
13. @EmptySource
14. @NullSource
15. @NullAndEmptySource

AssertClass static methods
++++++++++++++++++++++++++
1. assertEquals(expectedOutput, actualOutput)
2. assertThrows(Exception.class,Executable(I))
3. assertTimeOut(Duration,Executable(I))
4. assertTrue(boolean)
5. asssertSame(Object expected,Object acutal)
6. assertNotNull(Object expected)

@RepeatedTest: Allows to execute test method repeatedly for multiple times having control on count and name.
               It is very useful batch processing/ updating related tests.

CensusService.java
++++++++++++++++++
```java
package in.ineuron.service;
public class CensusService {
      public String exportData() {
            //logics.....
            return "data exported";
      }
}
```

TestCensusService.java
++++++++++++++++++++++
```java
package in.ineuron.test;
import static org.junit.jupiter.api.Assertions.assertEquals;
import org.junit.jupiter.api.DisplayName;
import org.junit.jupiter.api.RepeatedTest;
import com.nt.service.CensusService;

public class TestCensusService {
      @RepeatedTest(value = 10, name="execution of {displayName}-
{currentRepetition}/{totalRepetitions}")
      @DisplayName("Testing data export")
      public void testexportData() {
            System.out.println("TestCensusService.testexportData()");
            CensusService service = new CensusService();
            assertEquals("data exported", service.exportData());
      }
}
Output
 run: 10/10
 TestCensusService
      Testing data export
            execution of Testing data export-1/10
                  ;;;
                  ;;;

CensusService.java
++++++++++++++++++
public boolean isOdd(int no) {
      if (no%2==0)
            return false;
      else
            return true;
}
public String sayHello(String user) {
      return "Hello: "+user;
}
public boolean isEmpty(String name){
      return name.isBlank();
}

TestCensusService.java
++++++++++++++++++++++
private static CensusService service;

@BeforeAll
public static void setUpOnce() {
      service = new CensusService();
}

@ParameterizedTest
@ValueSource(ints = {10, 21, 34, 56, 11, 78})
public void testIsOdd(int n) {
      System.out.println("TestCensusService.testIsOdd()");
      assertTrue(service.isOdd(n));
}

@ValueSource(strings = {"raja", "ram"})
public void testSayHello(String user) {
      System.out.println("TestCensusService.testSayHello()");
      assertEquals("Hello: "+user, service.sayHello(user));
}


@ParameterizedTest
@NullAndEmptySource
public void testIsEmpty(String data) {
      System.out.println("CensusServiceTest.testIsEmpty()");
      Assertions.assertTrue(service.isEmpty(data));
}

@AfterAll
public static void cleanUpOnce() {
      service = null;
}
```

Q. What is the difference b/w assertEquals() and assertSame()?
Ans. assertEquals() checks content of given two values (like equals() method).
assertSame() checks whether given two references are pointing to
     same object or not (like ==).

Note: If want to write failure message by writing manual checking then use fail(-) method,once the fail() gets exectued the remaining
      statements won't be executed.

++++++++++++
Printer.java
++++++++++++
```java
public class Printer {
      private static Printer INSTANCE = new Printer();
      private Printer() {
      }
      public static Printer getInstance() {
            return INSTANCE;
      }
}
```

++++++++++++++++
PrinterTest.java
++++++++++++++++
```java
public class PrinterTest {

      @Test
      public void singletonTest() {
            Printer p1 = Printer.getInstance();
            Printer p2 = Printer.getInstance();
            Assertions.assertNotNull(p1);
            Assertions.assertNotNull(p2);

            if (p1 == null || p2 == null)
                  Assertions.fail("p1,p2 should not be null");
            Assertions.assertSame(p1, p2);
      }
}
```

++++++++
HttpUnit
++++++++
=> Unit testing for "WebApplications".
=> Generally, after developing web application, we test it by using browser to send request and to get response (in Manual testing environment).
=> To automate the unit testing of web application, we need stimulator for the browser software that can created using a Programming API
   i.e. HttpUnit.
=> HttpUnit is developed on the top of JUnit.

Maven dependency
```xml
<!-- https://mvnrepository.com/artifact/httpunit/httpunit -->
<dependency>
      <groupId>httpunit</groupId>
      <artifactId>httpunit</artifactId>
      <version>1.7</version>
      <scope>test</scope>
</dependency>
```

```
Application Development:
Step 1: Create maven project by taking maven-archetype-webapp as the archetype.
      [open pom.xml and change java version to 1.8, Right click on the Project
maven update the project]

      File --> maven project --> next --> select maven-archetype-webapp -->next -->
      group Id: iNeuron
      artifact Id: HttpUnit-LoginApp--> finish.

Step 2: Add following jars in pom.xml as dependent by collecting from
mvnrepository.com in pom.xml under <dependencies> tag.
      o httpunit.1.7.3.jar
      o junit-jupiter-api.5.7.0.jar
      o javax.servlet.api.4.0.1.jar
      o junit-jupiter-engine.5.7.0.jar

Step 3: Add index.html, verify.jsp as the web components in webapp folder having
login application logics.

Step 4: Configure Tomcat server with Eclipse IDE.

Step 5: Add Junit Test class in src/test/java folder using HttpUnit API.

Step 6: Run the LoginTest.java class as JUnit

++++++
pom.xml
+++++++
<dependencies>
      <dependency>
            <groupId>org.httpunit</groupId>
            <artifactId>httpunit</artifactId>
            <version>1.7.3</version>
            <scope>test</scope>
      </dependency>

      <dependency>
            <groupId>org.junit.jupiter</groupId>
            <artifactId>junit-jupiter-api</artifactId>
            <version>5.7.0</version>
            <scope>test</scope>
      </dependency>

      <dependency>
            <groupId>javax.servlet</groupId>
            <artifactId>javax.servlet-api</artifactId>
            <version>4.0.1</version>
            <scope>provided</scope>
      </dependency>

      <dependency>
            <groupId>org.junit.jupiter</groupId>
            <artifactId>junit-jupiter-engine</artifactId>
            <version>5.7.0</version>
            <scope>test</scope>
      </dependency>
</dependencies>

2. Create a index.html in src/main/webapp folder[public area]
```

```
index.html
++++++++++
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN"
"http://www.w3.org/TR/html4/frameset.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Login Page</title>
</head>
<body>
      <form action="verify.jsp" method="POST">
            <table border="0" bgcolor="cyan" align="center">
                  <tr>
                        <td>Enter username:</td>
                        <td><input type="text" name="uname"></td>
                  </tr>
                  <tr>
                        <td>Enter password:</td>
                        <td><input type="password" name="password"></td>
                  </tr>
                  <tr>
                        <td colspan="2"><input type="submit" value="Login"></td>
                  </tr>
            </table>
      </form>
</body>
</html>
```

3. Create verify.jsp in src/main/webapp folder[public area]

```
verify.jsp
++++++++++
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
      pageEncoding="ISO-8859-1"%>
<%
      //read form data
      String user = request.getParameter("uname").trim();
      String pwd = request.getParameter("password").trim();

      if(user.length()==0||user.equals("")||pwd.length()==0||pwd.equals("")) {
            out.print("provide credentials");
            return;
      }

      //write login/ authentication logic
      if(user.equalsIgnoreCase("sachin")&&pwd.equalsIgnoreCase("tendulkar"))
            out.print("valid credential");
      else
            out.print("invalid credential");
%>
```

4. Prepare a LoginTest.java in src/test/java folder

```
++++++++++++++
LoginTest.java
++++++++++++++
public class LoginTest {
```

```
private static WebConversation conversation;

@BeforeAll
public static void setUpOnce() {
      conversation = new WebConversation();
}

@Test
public void testWithValidCredentials() throws Exception {
      String url = "http://localhost:9999/HttpUnit-LoginApp/index.html";

      //get response by geneating request to index.html
      WebResponse response = conversation.getResponse(url);

      //get access to the form from the response
      WebForm form = response.getForms()[0];

      //set request param values to the form object
      form.setParameter("uname", "sachin");
      form.setParameter("password", "tendulkar");

      //submit the form and get the reponse
      WebResponse actualResponse = form.submit();

      // get actual output from actualResponse obj
      String actualOutput = actualResponse.getText().trim();

      // perform assertion (compare atual results with expected results)
      assertEquals("valid credential", actualOutput);
}

@Test
public void testWithInvalidCredentials() throws Exception {
      String url = "http://localhost:9999/HttpUnit-LoginApp/index.html";
      WebResponse response = conversation.getResponse(url);
      WebForm form = response.getForms()[0];
      form.setParameter("uname", "root");
      form.setParameter("password", "root123");
      WebResponse actualResponse = form.submit();

      String actualOutput = actualResponse.getText().trim();
      assertEquals("invalid credential", actualOutput);

}

@Test
public void testWithNoCredentials() throws Exception {
      String url = "http://localhost:9999/HttpUnit-LoginApp/index.html";
      WebResponse response = conversation.getResponse(url);
      WebForm form = response.getForms()[0];
      form.setParameter("uname", "");
      form.setParameter("password", "");
      WebResponse actualResponse = form.submit();

      String actualOutput = actualResponse.getText().trim();
      assertEquals("provide credentials", actualOutput);
}

@AfterAll
```

```
	public static void cleanOnce() {
		conversation = null;
	}

}
```

5. Run the server and Check it manually(url : http://localhost:9999/HttpUnit-LoginApp/index.html)
6. Run the test cases now and observe the Junit tab for the result of the test cases.