```
===================================
Unit Testing with Junit & Mocking
===================================

=> Testing individual components of the application is called as Unit Testing

=> Unit Testing is used to identify bugs in the code

=> Unit Testing will help us in developing Quality Code (bug free code)

=> To perform Unit Testing we will use Junit with Mocking

=================
What is Junit
=================

=> Junit is a java based framework

=> Junit is used to implement unit testing for java applications

====================
What is Mocking ?
====================

=> Unit Testing will be performed for individual components (isolated unit testing)

=> To perform Isolated Unit Testing we will use Mocking.

=> The process of creating dummy object is called as mocking.

=> Mock Objects are used only for unit testing.

Note: We can define behaviour for the mock object.

==========================
What is Code Coverage ?
==========================

=> The process of identifying which lines of code is executed in unit testing and which lines of code
is not executed in unit testing is called as Code Coverage.

=> Industry standard is 80% of code coverage for the project.

=> To identify code coverage of the project we have several tools

                        1) SonarQube
                        2) Jacocco

=> Using code coverage report, we can identify which lines of code is missed in unit testing so that
we can write effective unit test cases.

===========
Summary
===========

1) What is Unit Testing

2) Why Unit Testing

3) What is Isolated Unit Testing

4) What is Junit
```

```
5) What is Mocking

6) What is Code Coverage

7) Code Coverage Tools


==============================================================
public class PalindomeCheck {

        public boolean isPalindrome(String str) {
                StringBuffer sb = new StringBuffer(str);
                String revStr = sb.reverse().toString();
                if (str.equals(revStr)) {
                        return true;
                }
                return false;
        }
}
================================================================
public class PalindromeCheckTest {

        @ParameterizedTest
        @ValueSource(strings = { "racecar", "madam", "liril", "ashok" })
        public void testIsPalindrome(String str) {
                PalindomeCheck p = new PalindomeCheck();
                boolean actual = p.isPalindrome(str);

                if (str.equals("ashok")) {
                        assertFalse(actual);
                } else {
                        assertTrue(actual);
                }

        }
}
=================================================================


@Test

@ParameterizedTest

@ValueSource

Junit Assertions


=========================
Rest API Unit Testing
=========================

@WebMvcTest : To represent our target class for unit testing

@MockBean : To create mock obj for given class or interface

MockMvcRequestBuilder: It is used to prepare HTTP Request

MockMvc: It provided methods to send the request

MvcResult : It is used to hold response given by REST API

==========================
@WebMvcTest(value = WelcomeRestController.class)
public class WelcomeRestControllerTest {
        @MockBean
        private WelcomeService service;

        @Autowired
        private MockMvc mvc;

        @Test
        public void testGetWelcomeMsg() throws Exception {

                // define mock obj behaviour
                when(service.getMsg()).thenReturn("Welcome to Ashok IT..!!");

                // prepare http get request
                MockHttpServletRequestBuilder reqBuilder =
                                MockMvcRequestBuilders.get("/welcome");

                // send request & hold response
                MvcResult mvcResult = mvc.perform(reqBuilder).andReturn();

                // validate response
                MockHttpServletResponse response = mvcResult.getResponse();

                //String contentAsString = response.getContentAsString();

                int status = response.getStatus();
                assertEquals(200, status);

        }
}
--------------------------------------------------
===============
Code Coverage
===============

=>To check which lines of code executed in unit testing and which lines of code not executed in unit
testing.

=> Using Jacoco plugin we can generate code coverage report

=> Add below plugin in pom.xml and execute maven goals 'clean package'

<plugin>
                <groupId>org.jacoco</groupId>
                <artifactId>jacoco-maven-plugin</artifactId>
                <version>0.8.11</version>
                <executions>
                        <execution>
                                <goals>
                                        <goal>prepare-agent</goal>
                                </goals>
                        </execution>
                        <execution>
                                <id>report</id>
                                <phase>test</phase>
                                <goals>
                                        <goal>report</goal>
                                </goals>
                        </execution>
                </executions>
</plugin>


Note: Code Coverage Report will be available in "/target/site/jacoco/index.html"
```

```
====================
Exclusion in Jacocco
====================

=> In project for few classes, unit testing is not required

        Ex: Entities, bindings, constants & start classs


=> We can exclude those classes from jacocco report

                <configuration>
                        <excludes>
                                <exclude>**/in/ashokit/bindings/</exclude>
                                <exclude>**/in/ashokit/Application.class</exclude>
                        </excludes>
                </configuration>


==========================================
Spring Boot + REST API + Junit + Jacocco
==========================================

1) Rest Controller  : To handle request & response

2) Service : Business logic

3) DAO : data access logic

4) Junit : For Unit Testing

5) Mocking : To create Dummy Objects

6) Jacocco : For Code Coverage Report


Git Hub Repo : https://github.com/ashokitschool/SB_REST_API_JUnit_Jacocco_App.git
```