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

=> Free & open source java based framework

=> It is used for unit testing

Note: It is used only for java projects unit testing

========================
What is Unit Testing ?
========================

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

```
class UserService {

        m1() {
        }

        m2(){
        }

        m3(){
        }
}
```

=> Unit testing is used to identify weather code is working as expected or not.

=> Unit testing is used to identify bugs in the code.

=> With the help of unit testing we can identify bugs at early stage.

=> By performing unit testing we can provide quality code to higher environments

Note: Developer is responsible to perform unit testing in the project

================================
What is isolated unit testing ?
================================

=> Controller methods are depends on service methods and service methods depends on dao/repo methods.

=> When we are performing unit testing for controller method that time only controller method should be executed (service method shouldn't be called).

Ex:

class A {

m1() {

        //logic

        m2 ( );

        // logic

```
}

}

class B{

        m2(){
                //logic
        }

}
```

Note: When we are unit testing m1() method only m1() logic should be executed without calling real m2() method.

=> To perform isolated unit testing we will use Mocking.

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

=> The process of creating substitute object for real-object is called as Mocking.

=> Mocking is the practice of creating fake (mock) objects that imitate the behavior of real objects in a controlled way.

Note: Mock objects are required only for unit testing.

==========
Why Do We Need Mocking?
==========

1) Isolated unit testing

- You want to test only your component or method, not its dependencies.

Example: When testing a service method, you donâ€™t want the database or external API to be involved.

2) Avoiding External Dependencies

- Real services (like APIs, DBs) might be slow, costly, or unavailable.

3) Simulating Specific Scenarios

- You can mock unusual or error conditions (e.g., timeout, 404 error).

- Helps test how your code behaves under different conditions.

4) Improving Test Speed

- Mocks avoid real network/database calls, making tests faster.

==========
Mock Frameworks
==========

=> We have several mock frameworks in the market for creating mock objects

1) Mockito

2) Easy Mock

3) JMock

4) PowerMock

=> In Spring Boot, the default and most commonly used mocking framework is Mockito.

=> It is fully integrated with Spring Boot's testing support and used by default when you write unit or integration tests.

=> When we add 'spring-boot-starter-test' dependency then "junit + mockito" dependencies will be available by default in project build path.

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-test</artifactId>
  <scope>test</scope>
</dependency>
```

==========
Common Mockito Annotations in Spring Boot Tests
==========

1) @Mock

2) @MockBean

3) InjectMocks

4) @Spy