```
@BeforeAll
static void setup()
{
        //Executes only once before starting the tests
}

@BeforeEach
void init()
{
        //Executes before each test
}

@AfterEach
void tearDown()
{
        //Executes after each test is finished
}

@AfterAll
static void tearDownAll()
{
        //Executes only once after all the tests are executed
}
```

@Test //Used to mark a method as test

@Disabled //To disable a test. Can be used on method and class level

@DisplayName("Check string joined properly2") //Displays Test Name. Can be used on method and class level.

```
void test2() {

        String expected = new String(str[1] + str[2]);
        String actual = Main.combiner(str[1], str[2]);
        assertNotNull(actual);   //Asserts actual is not null
        assertEquals("Check your code..... ",expected, actual);    /*Asserts actual
is equal to expected if not shows the message*/
assertThrows(NumberFormatException.class,() -> Integer.parseInt(str[4]));
        //Asserts if this exception is thrown
        }

@RunWith(SpringRunner.class)
@SpringBootTest
class ProductServiceTest {

        @Autowired
        private ProductService service;

        @MockBean
        private ProductRepo repository;

        @Test
        public void getProductByIdTest() {

                when(repository.findByProductId("G1")).thenReturn(Optiona
        l.of(product));
                /*Returns Optional.of(product) for repository.findByProductId()
        call only is the method executes without error.*/
```

```
                Response productResponse = service.getProductById("G1");
                assertNotNull(productResponse);
                assertEquals("NOT EXPIRED",
productResponse.getResponseMessage());
        }
```

@RunWith (SpringRunner.class): You need this annotation to just enable spring boot features like @Autowire, @MockBean etc.. during junit testing is used to provide a bridge between Spring Boot test features and JUnit. Whenever we are using any Spring Boot testing features in our JUnit tests, this annotation will be required.

@SpringBootTest: This annotation is useful when we need to bootstrap the entire container. The annotation works by creating the ApplicationContext that will be utilized in our tests.

@MockBean: If no bean of the same type is defined, a new one will be added. This annotation is useful in integration tests where a particular bean, like an external service, needs to be mocked.

Mockito.doReturn(null).when(authService)
.getAuthenticatedTMForController(Matchers.anyString(), Matchers.any(PasswordAuthentication.class));

//returns null even if exceptions occurred in the called method

Mockito.verify(service, Mockito.times(1)).saveUser("1000");

//checks whether saveUser mthod was executed once. Mainly used to check if methods with void return has been executed.

@Spy

UserService service;

When using @Spy, mockito creates a real instance of the class and track every interactions with it. It maintains the state changes to it.