**TestSuite** – коли ми можемо виконати декілька класів з юніт тестами за раз. Використовуються з аннотаціями @RunWith і @Suite.SuitClasses

//JUnit Suite Test

@RunWith(Suite.class)

@Suite.SuiteClasses({TestJunit1.class, TestJunit2.class})

public class JunitTestSuite {

}

**TestRunner** запускають клас з тестами і повертають обєкт в якому можна перевірити чи були фейли.

Result result = JUnitCore.runClasses(TestJunit.class);

**Fixtures** – це фіксовані стани якихось обєктів які ми можемо зібрати перед виконанням тестів чи шось зробити після тестів. Методи setup(), teardown();

Головні класи **jUnit**:

* **Assert** – набір асертів. (assertEquals, assertNotNull, assertFalse)
* **TestCcase** – набір фікстур. (setup, teardown, getName)
* **TestResult** – можуть отримувати результат тестів. (errorCount, failureCounts)

@AfterClass – виконується лише раз після метода.

@Test(expected = ArithmeticException.class) – якшо ми очікуєм якийсь ексепшн.

Можна налаштувати тімсіті шоб він генерив репорти на тести.

**Моки** – обєкти які копіюють поведінку якихось реальних обєктів і коли ми їх використовуємо, то ми **очікуєм від них якусь поведінку**. Наприклад, якщо ми тестуємо метод сервіса, то ми миожемо замокати дані так, ніби сервіс їх віддає в нормальних умовах. Нприклад є сервіс в якого є метод getBooks() і він нам потрібен для тесту, тоді ми робимо мок імплементацію сервіса і ісплементуєм метод в якому самі популейтимо список книжок і віддаємо їх. Таким чином ми мокаєм сервіс і моделюємо роботу сервіса.

//I am a mock object!

public class MockBookServiceImpl implements BookService {

@Override

public List<Book> findBookByAuthor(String author) {

List<Book> books = new ArrayList<>();

if ("mkyong".equals(author)) {

books.add(new Book("mkyong in action"));

books.add(new Book("abc in action"));

books.add(new Book("bot"));

}

return books;

}

//implements other methods...

}

Також ми можемо у себе в тесті збирати ліст книг і підсовувати їх коли буде відбуватися кол до метода реального сервіса використовуючи when - thenReturn.

public class AuthorServiceTest {

@Test

public void test\_total\_book\_by\_mockito() {

//1. Setup

List<Book> books = Arrays.asList(

new Book("mkyong in action"),

new Book("abc in action"),

new Book("bot"));

BookServiceImpl mockito = mock(BookServiceImpl.class);

//if the author is "mkyong", then return a 'books' object.

when(mockito.findBookByAuthor("mkyong")).thenReturn(books);

AuthorServiceImpl obj = new AuthorServiceImpl();

obj.setBookService(mockito);

obj.setBookValidatorService(new FakeBookValidatorService());

//2. Test method

int qty = obj.getTotalBooks("mkyong");

//3. Verify result

assertThat(qty, is(2));

}

}

**Stub** – це клас з предефейненими значеннями які ми хочемо отримувати в своєму тесті. Тобто ми користуємося ним як якоюсь депенденсьою що репрезентує **стан** з певним набором респонсів.