Java JUnit

Table of contents

Introduced in 2026-03-26-testing[1] (Lecture, Week 5). See java-testing[2] for the broader testing concepts JUnit is used to implement.

What is JUnit?

An open-source test automation framework for Java — one of many *xUnit* frameworks (JUnit, NUnit, CPPUnit, PyUnit…). This course uses JUnit 4. Depending on how software is structured, unit-testing frameworks like JUnit can also be used for other kinds of automated testing (the term "unit testing" is sometimes loosely used to mean *any* automated test).

Writing tests

Tests are defined in classes — conventionally one test class per real class. `@Test` marks a method as a test JUnit should run; test methods take no parameters and return `void`, and should be named for what they check.

[1] courses/csse2002/2026-03-26-testing.pdf
[2] courses/csse2002/java-testing.pdf

```java
public class Person {
    private String name;
    private int age;

    public Person(String name, int age) {
        this.name = name;
        this.age = age;
    }

    public String getName() { return name; }
    public int getAge() { return age; }
}
```

```java
import org.junit.Test;
import static org.junit.Assert.assertEquals;

public class PersonTest {
    @Test
    public void testGetAge() {
        Person p1 = new Person("Jack", 10);
        assertEquals(10, p1.getAge());
    }

    @Test
    public void testGetName() {
        Person p1 = new Person("Jack", 10);
        assertEquals("Jack", p1.getName());
    }
}
```

`@Before` and `@After`

Each test method should be independent, but common setup (e.g. object creation) can be factored out:

```java
public class PersonTest {
    private Person person;

    @Before
    public void setUp() {
        person = new Person("Jack", 10);
    }

    @After
    public void tearDown() {
```

```
        person = null;
    }
    // ...
}
```

`@Before` runs before **each** test; `@After` runs after **each** test.

`Assert`

Some static methods on `Assert`:

- `assertEquals` — asserts two objects are equal.
- `assertArrayEquals` — asserts two object arrays are equal.
- `assertFalse` / `assertTrue` — asserts a condition is false/true.
- `assertSame` / `assertNotSame` — asserts two objects do/don't refer to the same object.

Checking for exceptions

Don't catch the exception yourself — tell the test runner which exception type to expect via `@Test(expected = ...)`, and let it catch it:

```
@Test(expected = EOFException.class)
public void testExceptions() throws EOFException {
    callThatShouldThrowEOFException();
}
```

Note the `.class` after the exception type is required.

Black box vs white box JUnit tests

- **Black box**: write tests purely from the specification/Javadoc, without looking at the implementation, then apply smoke tests / boundary tests / equivalence classes (see java-testing[3]) to decide what to check — tests must not violate the method's preconditions.
- **White box**: design tests *using* knowledge of the internal logic, aiming to cover branches, loops, and edge cases (e.g. targeting 100% branch coverage) — e.g. for a `calculateDiscount(amount, isMember)` method with nested `if/else` branches on both parameters, white-box tests are written to exercise every branch combination (amount 0; member above/at-or-below 100; non-member above/at-or-below 100).

[3] courses/csse2002/java-testing.pdf