AssertJ

Write Readable Tests!

Görge Albrecht

Software Developer since 1989

Code Mentor - taking care of code

Need help writing simpler code? Contact me!

@g_o_rge · code-mentor.com

What are Assertions?

- verify conditions at runtime
- fail if condition not met
- continue silently if condition is met

Java built in assertions

```
public static void main(String[] args) {
  assert args == null : "expected to fail";
}
```

- throws `java.lang.AssertionError`

```
Exception in thread "main" java.lang.AssertionError: expected to fail
        at com.codementor.assertj.JavaAssert.main(JavaAssert.java:6)
```

- must be enabled with `-ea`
- write your tests only based on `assert`?

Groovy PowerAssert

```
assert 1 + 2 > 3 + 4
```

```
assert 1 + 2 > 3 + 4
         |   |   |
         3   |   7
             false
```

Java Assertion Frameworks

JUnits Assert

- comparing expected with actual value

```
@Test
public void simpleAssertEquals() {
  assertEquals("text", "text");
}
```

- `org.junit.Assert`
- `assertEquals` for all Java primitives and `Object`
- `assertNull`, `assertSame`
- `assertNot*` of all the above
- `assertTrue`, `assertFalse`

Hamcrest

- DSL style: "assert that a value matches something"

```
@Test
public void simpleAssertEquals() {
  assertThat("text", is("text"));
}
```

- Matchers for `String`, `Iterable`, `File`, `XPath`
- extensible by writing own Matcher
- `org.hamcrest.MatcherAssert#assertThat`
- `org.hamcrest.Matchers`

AssertJ

- Fluent interface for writing assertions

```
@Test
public void simpleAssertEquals() {
  assertThat("text").isEqualTo("text");
}
```

- Fluent API enables IDE code completion
- Assertions for almost everything:
  `CompletableFuture`, Guava, Joda-Time, DB, Swing
- multiple ways to extend:
  Conditions, Assertions, Generator
- `org.assertj.core.api.Assertions#assertThat`

Features

Assertion chaining

- JUnit

```
assertTrue("Software Quality".contains("Software"));
assertTrue("Software Quality".contains("Quality"));
```

- Hamcrest

```
assertThat("Software Quality",
  both(containsString("Software"))
    .and(containsString("Quality")));
```

- AssertJ

```
assertThat("Software Quality")
  .contains("Software")
  .contains("Quality");
```

Asserting Lists

List contains items

- Hamcrest

```
assertThat(asList("Software", "Quality", "Days"),
  hasItems("Software", "Quality"));
```

- AssertJ

```
assertThat(asList("Software", "Quality", "Days"))
  .contains("Software", "Quality");
```

Asserting every item

- Hamcrest

```
assertThat(asList("Software", "Quality", "Days"),
  everyItem(containsString("a")));
```

- AssertJ

```
assertThat(asList("Software", "Quality", "Days"))
  .allMatch(s -> s.contains("a"));
```

```
// more expressive message
assertThat(asList("Software", "Quality", "Days"))
  .allSatisfy(s -> assertThat(s).contains("a"));
```

Verify all at once

Hamcrest

allOf

```
final List<String> swqd = Arrays.asList("Software", "Quality", "Days");

@Test
public void hamcrest() {
  org.hamcrest.MatcherAssert.assertThat(swqd,
    Matchers.<List<String>>allOf(
      hasSize(3),
      hasItem("Hardware"),
      hasItem("Konferenz")
    ));
}
```

AssertJ

SoftAssertions

```
final List<String> swqd = Arrays.asList("Software", "Quality", "Days");

@Test
public void assertJ() {
  SoftAssertions.assertSoftly(s ->
    s.assertThat(swqd)
      .hasSize(3)
      .contains("Hardware")
      .contains("Konferenz")
  );
}
```

JUnit 5

assertAll

```
final List<String> swqd = Arrays.asList("Software", "Quality", "Days");

@Test
public void junit5() {
  org.junit.jupiter.api.Assertions.assertAll(
    () -> assertEquals(3, swqd.size()),
    () -> assertEquals("Hardware", swqd.get(0)),
    () -> assertEquals("Konferenz", swqd.get(2))
  );
}
```

Asserting POJOs

Hamcrest

hasProperty

```
private final Address address =
  new Address("New City", "Some Street", "No");

@Test
public void hamcrest() {
  org.hamcrest.MatcherAssert.assertThat(address, allOf(
    hasProperty("city", equalTo("New City")),
    hasProperty("street", equalTo("Irgendeinestraße")),
    hasProperty("number", equalTo("Nr"))
  ));
}

public static class Address {
  public String getCity() {
    return city;
  }

  public String getStreet() {
```

AssertJ

hasFieldOrPropertyWithValue

```
private final Address address =
  new Address("New City", "Some Street", "No");

@Test
public void assertJ() {
  org.assertj.core.api.Assertions.assertThat(address)
      .hasFieldOrPropertyWithValue("city", "New City")
      .hasFieldOrPropertyWithValue("street", "Irgendeinestraße")
      .hasFieldOrPropertyWithValue("number", "Nr");
}
```

Messages

JUnit

```
assertEquals("Hardware", "Software");
```

```
Expected :Hardware
Actual   :Software
```

Hamcrest

```
assertThat("Software", is("Hardware"));
```

```
Expected: is "Hardware"
     but: was "Software"
```

AssertJ

```
assertThat("Software").isEqualTo("Hardware");
```

```
Expected: is "Hardware"
     but: was "Software"
```

String contains

Hamcrest

```
assertThat("Software Quality", containsString("Days"));
```

```
Expected: a string containing "Days"
     but: was "Software Quality"
```

AssertJ

```
assertThat("Software Quality").contains("Days");
```

```
Expecting:
 <"Software Quality">
to contain:
 <"Days">
```

Asserting every item

Hamcrest

```
assertThat(asList("Software", "Quality", "Days"),
  everyItem(containsString("o")));
```

```
Expected: every item is a string containing "o"
     but: an item was "Quality"
```

AssertJ

```
assertThat(asList("Software", "Quality", "Days"))
  .allMatch(s -> s.contains("o"));
```

```
Expecting all elements of:
  <["Software", "Quality", "Days"]>
to match given predicate but this element did not:
  <"Quality">
```

... improved

```
assertThat(asList("Software", "Quality", "Days"))
  .allMatch(s -> s.contains("o"));
```

```
assertThat(asList("Software", "Quality", "Days"))
  .allSatisfy(s -> assertThat(s).contains("o"));
```

```
Expecting all elements of:
  <["Software", "Quality", "Days"]>
to satisfy given requirements, but this element did not:
  <"Quality">
Details: "
Expecting:
 <"Quality">
to contain:
 <"o"> "
```

Readability

```
assertEquals("foo", "bar");
```

```
assertThat("foo", is("bar"));
```

```
assertThat("foo").isEqualTo("bar");
```

```
assertThat "foo" isEqualTo "bar"
```

Readability

- JUnit

```
assertEquals("foo", "bar");
```

- Hamcrest

```
assertThat("foo", is("bar"));
```

- AssertJ

```
assertThat("foo").isEqualTo("bar");
```

- AssertJ on Groovy

```
assertThat "foo" isEqualTo "bar"
```

What do you think?

Hamcrest?

AssertJ?

Try Groovy Spock!

```
class MathSpec extends Specification {
  def "maximum of two numbers"(int a, int b, int c) {
    expect:
    Math.max(a, b) == c

    where:
    a | b | c
    1 | 3 | 3
    7 | 4 | 7
    0 | 0 | 0
  }
}
```

Thanks for listening

@g_o_rge · code-mentor.com