

# AssertJ

Write Readable Tests!

# Göрге Albrecht

Software Developer since 1989

Code Mentor - taking care of code

Need help writing simpler code? Contact me!

 @g\_o\_rge ·  [code-mentor.com](https://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, assertEquals
- 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("JavaLand".contains("Java"));  
assertTrue("JavaLand".contains("Land"));
```

- Hamcrest

```
assertThat("JavaLand Brühl Germany",  
    both(containsString("Java"))  
    .and(containsString("Germany")));
```

- AssertJ

```
assertThat("AssertJ @ JavaLand")  
    .contains("Java")  
    .contains("Assert");
```

# Asserting Lists

# List contains items

- Hamcrest

```
assertThat(asList("AssertJ", "Talk", "JavaLand"),  
           hasItems("AssertJ", "JavaLand"));
```

- AssertJ

```
assertThat(asList("AssertJ", "Talk", "JavaLand"))  
           .contains("AssertJ", "JavaLand");
```

# Asserting every item

- Hamcrest

```
assertThat(asList("JavaLand", "Talk", "Germany"),  
    everyItem(containsString("a")));
```

- AssertJ

```
assertThat(asList("JavaLand", "Talk", "Germany"))  
    .allMatch(s -> s.contains("a"));
```

```
// more expressive message  
assertThat(asList("JavaLand", "Talk", "Germany"))  
    .allSatisfy(s -> assertThat(s).contains("a"));
```

Verify all at once

# Hamcrest

## allOf

```
final List<String> list = Arrays.asList("JavaLand", "AssertJ", "Talk");

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



# AssertJ

## SoftAssertions

```
final List<String> list = Arrays.asList("JavaLand", "AssertJ", "Talk");

@Test
public void assertJ() {
    SoftAssertions.assertSoftly(s ->
        s.assertThat(list)
            .hasSize(3)
            .contains("JavaLand")
            .contains("Talk")
    );
}
```

# JUnit 5

## assertAll

```
final List<String> list = Arrays.asList("JavaLand", "AssertJ", "Talk");

@Test
public void junit5() {
    org.junit.jupiter.api.Assertions.assertAll(
        () -> assertEquals(3, list.size()),
        () -> assertEquals("JavaLand", list.get(0)),
        () -> assertEquals("Talk", list.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("JavaLand 2017", containsString("Brühl"));
```

```
Expected: a string containing "Brühl"  
but: was "JavaLand 2017"
```

# AssertJ

```
assertThat("JavaLand 2017").contains("Brühl");
```

```
Expecting:  
  <"JavaLand 2017">  
to contain:  
  <"Brühl">
```

Asserting every item

# Hamcrest

```
assertThat(asList("JavaLand", "Brühl", "Germany"),  
           everyItem(containsString("a")));
```

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

# AssertJ

```
assertThat(asList("JavaLand", "Brühl", "Germany"))  
    .allMatch(s -> s.contains("a"));
```

Expecting all elements of:

```
<["JavaLand", "Brühl", "Germany"]>  
to match given predicate but this element did not:  
<"Brühl">
```

# ... improved

```
assertThat(asList("JavaLand", "Brühl", "Germany"))  
    .allMatch(s -> s.contains("a"));
```

```
assertThat(asList("JavaLand", "Brühl", "Germany"))  
    .allSatisfy(s -> assertThat(s).contains("a"));
```

```
Expecting all elements of:  
  <["JavaLand", "Brühl", "Germandy"]>  
to satisfy given requirements, but this element did not:  
  <"Brühl">  
Details: "  
Expecting:  
  <"Brühl">  
to contain:  
  <"a"> "
```



# 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](https://code-mentor.com)