Manual testing

How-to

- Write code

- Manually run it

- Manually check it

- What can go wrong?

- What can be missing?

What can go wrong

- Incorrect execution

- Misread results

- Forgotten tests

- …

What can be missing

- Multiple users

- Time dependencies

- Location dependencies

- …

Automated testing

How-to

- Write code

- Write test code

- Run tests

Benefits

- No forgetting tests

- No mistakes running tests

- No effort to run

- Can "fake" parts of the system

	Manual	Automated
Fixed cost	Low	High
Variable cost	High	Low
Input	Requirement	Specification
Human error	Yes	No
Edge cases	No	Yes

JUnit

```java
@Test
void onePlusOneIsTwo() {
    assertEquals(1 + 1, 2);
}
```

assertEquals

```java
public static void assertEquals(int expected,
                                int actual)
```

JUnit + Hamcrest

```
@Test
void onePlusOneIsTwo() {
  assertThat(1 + 1, is(2));
}
```

JUnit + Hamcrest

```
@Test
```
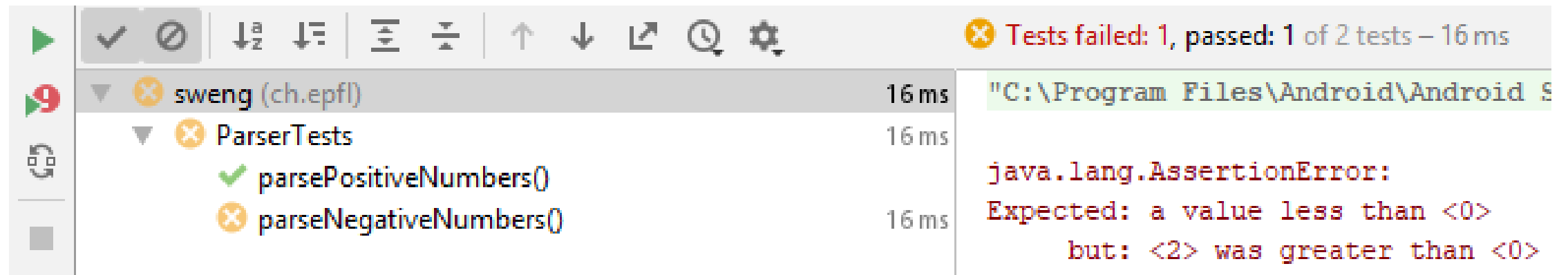
```
    either(empty())
   .or(contains(1, 2, 3)));
}
```

IntelliJ (and others)

Writing good tests

Test shape

- Arrange

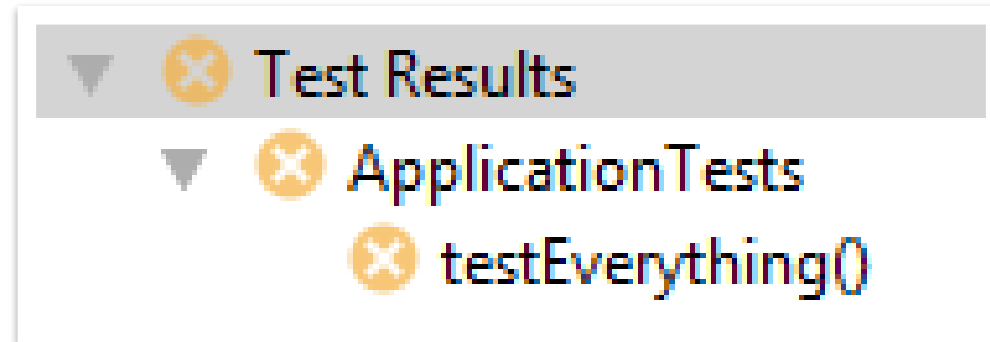 "Given this situation…"
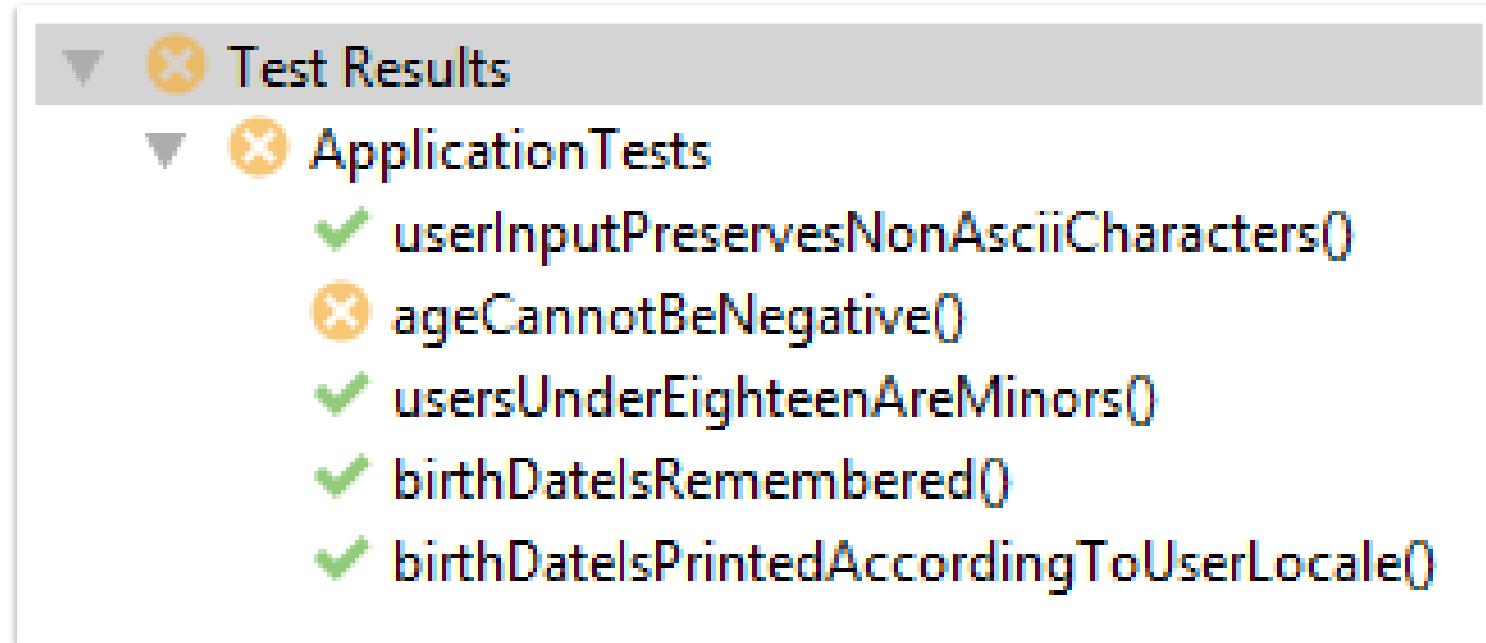
- Act

  "When this happens…"

- Assert

  "Then the outcome should be…"

Test size

Test size

Test code is code

- Also in repository

- Same standards

- Same quality level

Test names

- birthDateIsPrintedCorrectly

- nameCanIncludeThaiCharacters

- shareFeatureWorks

- canParseNumbersAtLeastOneThousand

Test names

- birthDateIsPrintedCorrectly

- nameCanIncludeThaiCharacters

- shareFeatureWorks

- canParseNumbersAtLeastOneThousand

Sharing test code

```
@BeforeAll static void globalSetup() { ... }

@AfterAll static void globalTeardown() { ... }

@BeforeEach void localSetup() { ... }

@AfterEach void localTeardown() { ... }

@Test ...
```