Exploring Android Testing Best Practices and Developing Instructional Materials

Degree programme: BSc in Computer Science
Thesis advisor: Prof. Dr. Ulrich Fiedler
Expert: Urs Keller (Revault Sàrl)

Software testing is an often undervalued aspect in software projects. This thesis updated the Smart Device Programming course by evaluating essential testing aspects, blending general principles with Android-specific insights. The process included developing interactive code-along sessions and utilizing student feedback to refine the content. The result: Three dynamic slide sets designed to enhance learning and mastery in Android app testing.

Introduction

The Smart Device Programming course, part of the Distributed Systems and IoT specialization, provides a foundation in Android app programming. Currently, the course lacks materials that provide insights on how to test Android apps.
To address this, students should be provided with materials explaining the value of testing, what to test, and how to test it, with a focus on Android programming.

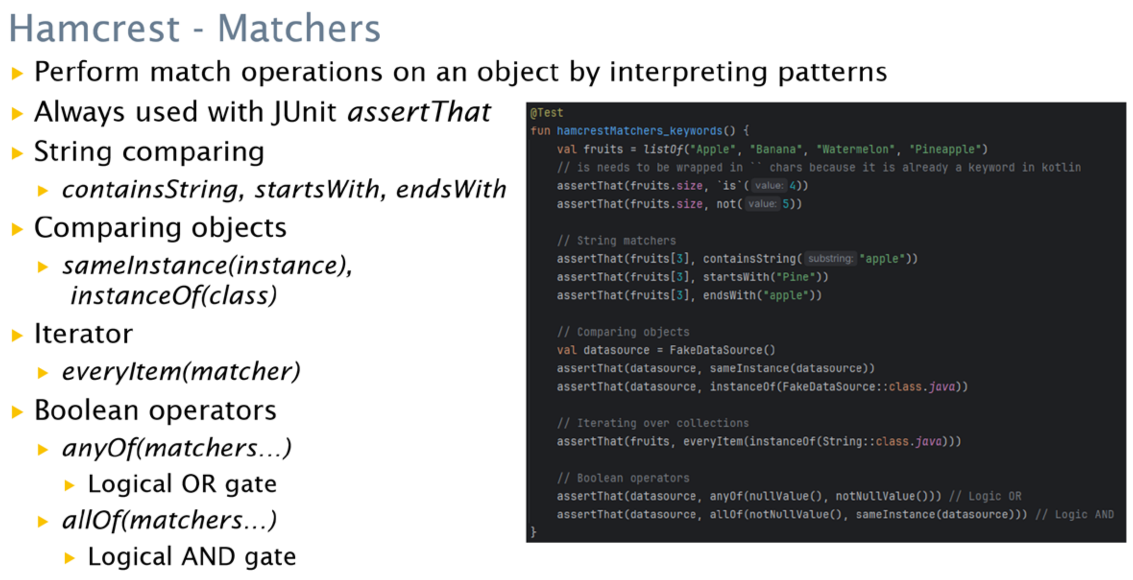
Approach

First, we evaluated which aspects of testing to teach, deciding whether to focus more on platform-specific frameworks or on more general concepts. The result was to concentrate on general topics, while integrating some Android-specific aspects.
The next step was to create the teaching material based on the researched topics. These topics include testing fundamentals, test automation frameworks, and an interactive code-along.
Finally, exam questions were created to harmonize with the learning objectives of the developed teaching material. Bloom's taxonomy was utilized as a guideline to ensure the questions tested not only factual knowledge but also higher-level thinking.
To evaluate the teaching material, a test lesson in Smart Device Programming was conducted, during which the teaching material was presented. Students were then asked to provide specific feedback about the teaching material, which was used to further refine the content developed..

Nathanaël Hunziker
Distributed Systems and IoT

Results

In the end, three different slide sets were created: two discussing theory and one for the code-along. Based on the feedback gathered during the test lesson, these slides were refined and improved. For instance, recaps were added at the end of the slides.
The test lesson was overall well received, especially the code-along, which provided some variety.



Hamcrest - Matchers

- Perform match operations on an object by interpreting patterns
- Always used with JUnit *assertThat*
- String comparing
  - *containsString, startsWith, endsWith*
- Comparing objects
  - *sameInstance(instance), instanceOf(class)*
- Iterator
  - *everyItem(matcher)*
- Boolean operators
  - *anyOf(matchers...)*
    - Logical OR gate
  - *allOf(matchers...)*
    - Logical AND gate

```
@Test
fun hamcrestMatchers_keywords() {
    val fruits = listOf("Apple", "Banana", "Watermelon", "Pineapple")
    // is needs to be wrapped in `` chars because it is already a keyword in kotlin
    assertThat(fruits.size, `is`( value: 4))
    assertThat(fruits.size, not( value: 5))

    // String matchers
    assertThat(fruits[3], containsString( substring: "apple"))
    assertThat(fruits[3], startsWith("Pine"))
    assertThat(fruits[3], endsWith("apple"))

    // Comparing objects
    val datasource = FakeDataSource()
    assertThat(datasource, sameInstance(datasource))
    assertThat(datasource, instanceOf(FakeDataSource::class.java))

    // Iterating over collections
    assertThat(fruits, everyItem(instanceOf(String::class.java)))

    // Boolean operators
    assertThat(datasource, anyOf(nullValue(), notNullValue())) // Logic OR
    assertThat(datasource, allOf(notNullValue(), sameInstance(datasource))) // Logic AND
}
```

Final gradle test summary of the Unit tests conducted in the code-along