AssertJ - Improving Test Code Readability

Core Features of AssertJ

AssertJ Assertions

Rich and Fluent Assertions

```
assertThat("Mastering Unit Testing").startsWith("Mastering").endsWith("Testing");
```

Descriptive and Detailed Error Messages

```
assertThat(42).as("check the ultimate question's answer").isEqualTo(43);
```

Dealing with Collections and Arrays

```
assertThat(Arrays.asList("Java", "JUnit","AssertJ"))
                                        .contains("JUnit")
                                        .doesNotContain("Python");
```

Custom Assertions in AssertJ

```java
public class UserAssert extends AbstractAssert<UserAssert, User> {

    public UserAssert(User actual) {
        super(actual, UserAssert.class);
    }

    public static UserAssert assertThat(User actual) {
        return new UserAssert(actual);
    }

    public UserAssert hasName(String name) {
        isNotNull();

        if (!actual.getName().equals(name)) {
            failWithMessage("Expected user's name to be <%s> but was <%s>", name, actual.getName());
        }

        return this;
    }

    public UserAssert isActive() {
        isNotNull();

        if (!actual.isActive()) {
            failWithMessage("Expected user to be active");
        }

        return this;
    }
}
```

Asserting Exceptions

```java
@Test
public void shouldThrowInvalidUserExceptionForInvalidUser() {
    User invalidUser = new User("Invalid User", false);

    assertThatThrownBy(() -> userService.createUser(invalidUser))
        .isInstanceOf(InvalidUserException.class)
        .hasMessageContaining("Invalid user data");
}
```

Advanced Assertion Techniques

```java
@Test
public void advancedAssertionsExample() {
    User expectedUser = new User("Alice", true);
    User actualUser = userService.getUser("Alice");

    // Recursive Comparison
    assertThat(actualUser).usingRecursiveComparison().isEqualTo(expectedUser);

    // Conditional Assertions
    assertThat(actualUser.getRoles())
        .hasSizeGreaterThanOrEqualTo(1)
        .allMatch(role -> role.startsWith("ROLE_"));

    // Assertions for Java 8 Types
    Optional<User> optionalUser = userService.findUserById("123");
    assertThat(optionalUser).isPresent()
        .hasValueSatisfying(user -> assertThat(user.getName()).isEqualTo("Alice"));
}
```

Real-world Use Cases

Sorting List Test

```java
@Test
public void whenSortedList_thenCorrectlySorted() {
    List<String> names = Arrays.asList("Anna", "John", "Bob");
    Collections.sort(names);

    assertThat(names).containsExactly("Anna", "Bob", "John");
}
```

Test Code Readability and Maintainability

```
@Test
public void whenValidatingProperties_thenCorrect() {
    Person person = new Person("John", 30, "Engineer");

    assertThat(person)
        .hasFieldOrPropertyWithValue("name", "John")
        .hasFieldOrPropertyWithValue("age", 30)
        .hasFieldOrProperty("occupation");
}
```

Testing API Responses

```
@Test
public void whenApiResponse_thenCorrect() {
    ApiResponse response = apiClient.getUserDetails();

    assertThat(response)
        .extracting("status", "userId", "title")
        .containsExactly(200, 5, "Test User");
}
```

AssertJ Best Practices and Common Pitfalls

Best Practice 1: Leverage Fluent API for Clarity

```java
assertThat(user.getName()).as("check user's name")
                          .isEqualTo("Alice")
                          .startsWith("A")
                          .endsWith("e")
                          .isNotBlank();
```

Best Practice 2: Use Descriptive Custom Assertions

```
assertThat(user).as("verify active user")
                .hasName("Alice")
                .isActive();
```

Best Practice 3: Opt for Specific Assertions Over General Ones

```java
// Specific Assertion
assertThat(numbers).containsExactlyInAnyOrder(1, 2, 3);

// General Assertion
assertThat(numbers.size()).isEqualTo(3);
```

Pitfall 2: Ignoring AssertJ's Advanced Features

```
SoftAssertions softAssertions = new SoftAssertions();
softAssertions.assertThat(user.getName()).isEqualTo("Alice");
softAssertions.assertThat(user.getAge()).isBetween(20, 30);
softAssertions.assertAll();
```