Software Testing

Software Engineering
Prof. Maged Elaasar

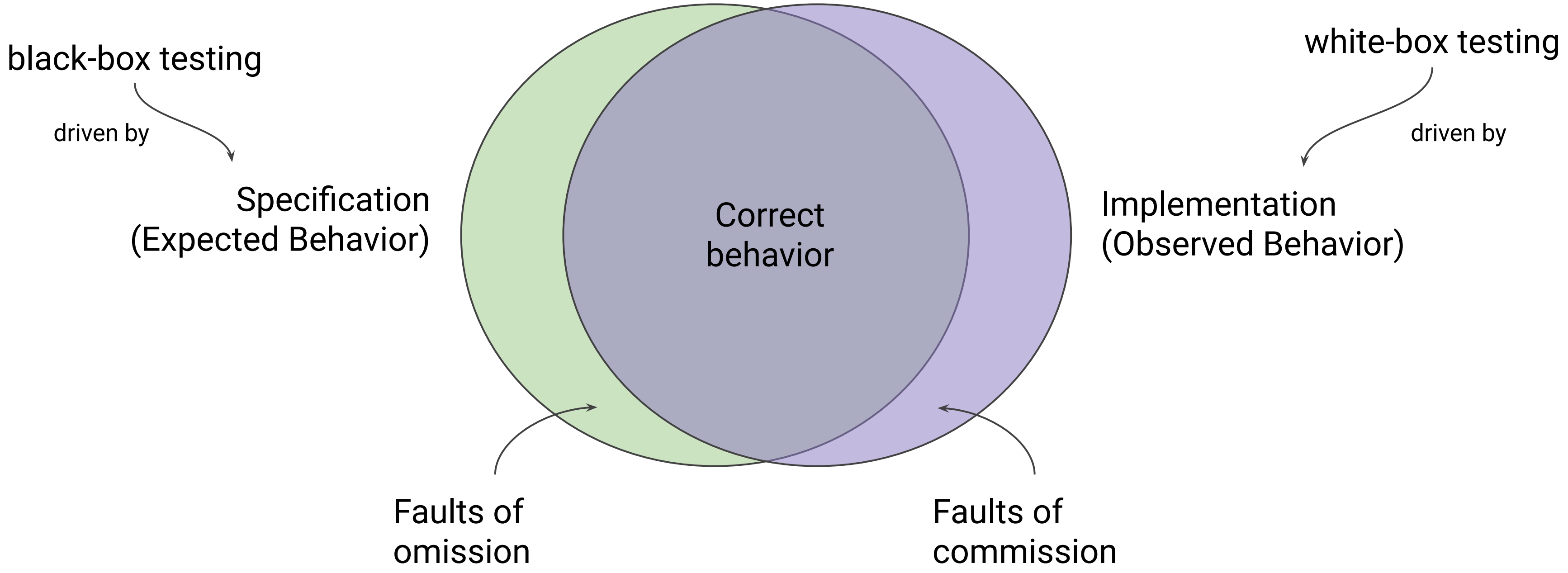
© 2020 Elaasar

Learning objectives

- Introduction to software testing
- Automated testing with JUnit

Software Testing

A process of analyzing a software item to detect the differences between **observed and expected behavior**

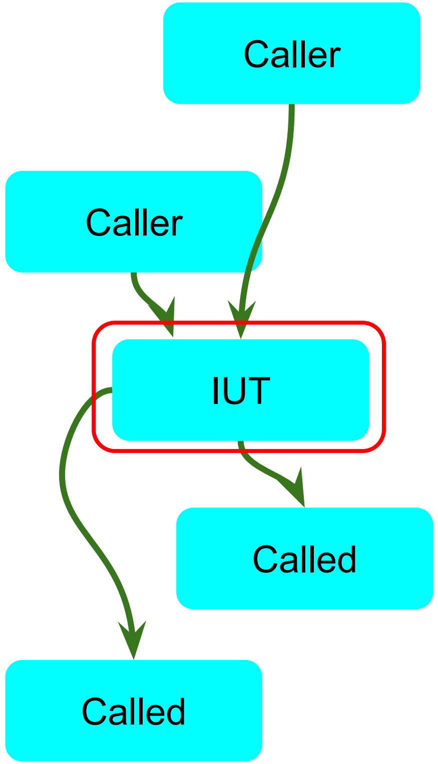


Levels of Software Testing

- **Unit testing** (done by developers) tests if individual modules of the source code are working properly.
- **Regression testing**: (done by developers or testers) tests if previously tested software still works properly after a change
- **Integration testing** (done by testers) tests the interface between two or more unit tested modules.
- **System testing** (done by testers) tests the fully integrated system using end to end scenarios.
- **Acceptance testing** (done by users) tests user requirements on a release candidate of the system.

Isolating the Item-Under-Test (IUT)

- Isolate IUT from its caller and called items
  - To reduce uncertainties
- **Test stub**: used to simulate the behavior of called items not yet integrated with the IUT
- **Test driver**: used to test the IUT when the callers are not available yet



Test Driven Development

- Writing testable code is hard
    - Testable code is componentized and loosely coupled
    - Requires you to create seams where tests can be inserted

- Test driven development advocates writing tests first before code
    - Forces you to write testable code
    - can be applied at any level, but is most common for unit testing
    - Outcome is compared to the expected result (**gold standard** or **test oracle**)

Limitations of Software Testing

```
int foo(int j) {
        jnt j = j - 1; // should have been j = j + 1
        return j / 30000;
}
```

For a 16-bit integers, out of 65,530 testable values, only 4 will detect the bug:
30000,-30000, 29999, -29999

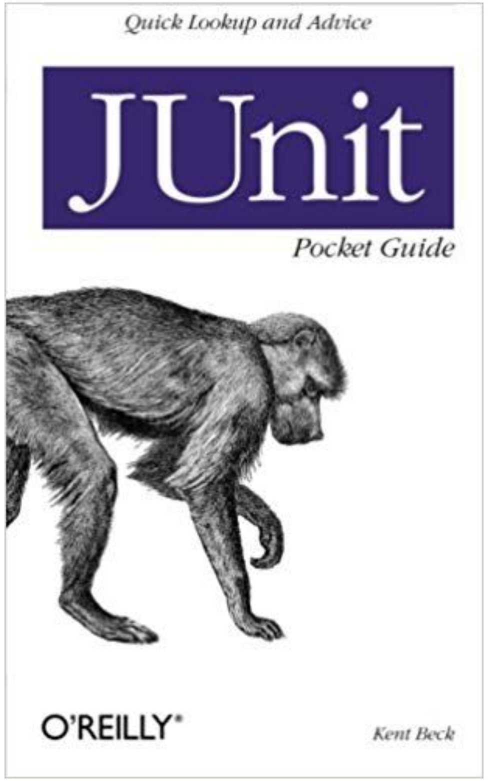
Input (j)	Expected Result	Actual Result
1	0	0
42	0	0
40000	1	1
-64000	-2	-2

Software Test Automation

- Test automation is the automatic execution of software tests
- Why automate software testing?
  - Human testers are expensive, inconsistent, slow, have needs and have better things to do
  - It provides rapid feedback
  - It builds confidence in change
- Requires specific tools, frameworks or environments

JUnit

- Simple framework for writing automated unit tests in Java
- Support for test assertions
- Support for test suite development
- Support for immediate test reporting

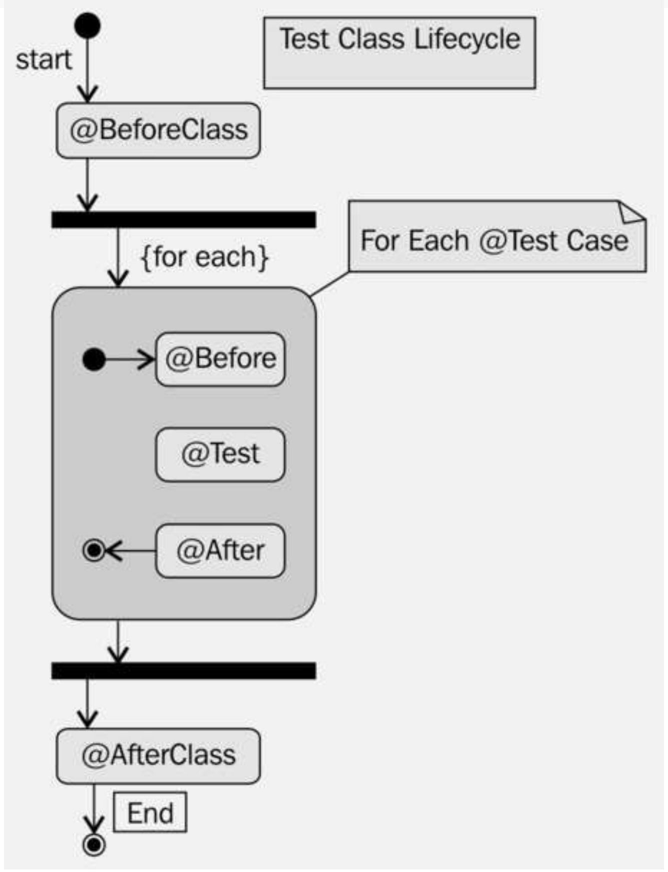


JUnit features

- Annotations to identify the test methods.
- Assert statement(s) for testing expected results.
- Test fixtures for sharing common test data.
- Aggregating tests using test cases and test suites.

JUnit annotations

- **@BeforeClass**
- **@Before**
- **@Test**
- **@After**
- **@AfterClass**



JUnit example 1

```
public class DummyTest {
        private static List<String> list;
        @BeforeClass public static void beforeClass() {
                list = new ArrayList<String>();
        }
        @Before public void before() {
                list.add("Alex");
        }
        @Test public void getElement() {
                String element = list.get(0);
                assertEquals(element, "Alex");
        }
        @After public void after() {
                list.remove("Alex");
        }
        @AfterClass public static void afterClass() {
                list = null;
        }
}
```

JUnit assertions

- void assertEquals(boolean expected, boolean actual)
- void assertTrue(boolean expected, boolean actual)
- void assertFalse(boolean condition)
- void assertNotNull(Object object)
- void assertNull(Object object)
- void assertSame(boolean condition)
- void assertNotSame(boolean condition)
- ...

JUnit example 2

```
public class StudentDetails {
        public String calculateStatus(Student student) {
                String status;
                if (student.getGrade() < 50){
                        status = "Fail";
                } else {
                        status = "Pass";
                }
                return status;
        }
}
public class Student {
        private String name;
        private int grade;
}
```

```
public class StudentDetailsTest {
        StudentDetails studentDetails = new StudentDetails();
        Student student = new Student();
        @Test public void CalculatePercentage() {
                student.setName("Derick");
                student.setGrade(79);
                String status=studentDetails.calculateStatus(student);
                assertEquals("Pass", status);
        }
}
```

Other JUnit features

- Test Case
  - A Test Case is a class that defines a set of tests.
- Test Suite
  - A Test Suite is a collection of test cases.
- JUnit tutorials
  - https://www.guru99.com/junit-tutorial.html
  - https://www.vogella.com/tutorials/JUnit/article.html
- Unit test frameworks for other languages:
  - CppUnit: https://en.wikipedia.org/wiki/CppUnit
  - PyUnit: https://en.wikipedia.org/wiki/PyUnit
  - etc.

```
@SuiteClasses({
    JUnitTestCase1.class,
    JUnitTestCase2.class,
    JUnitTestCase3.class
})
public class JUnitSuite1 {

}
```

Software Testing 1 Quiz