JUnit

JUnit

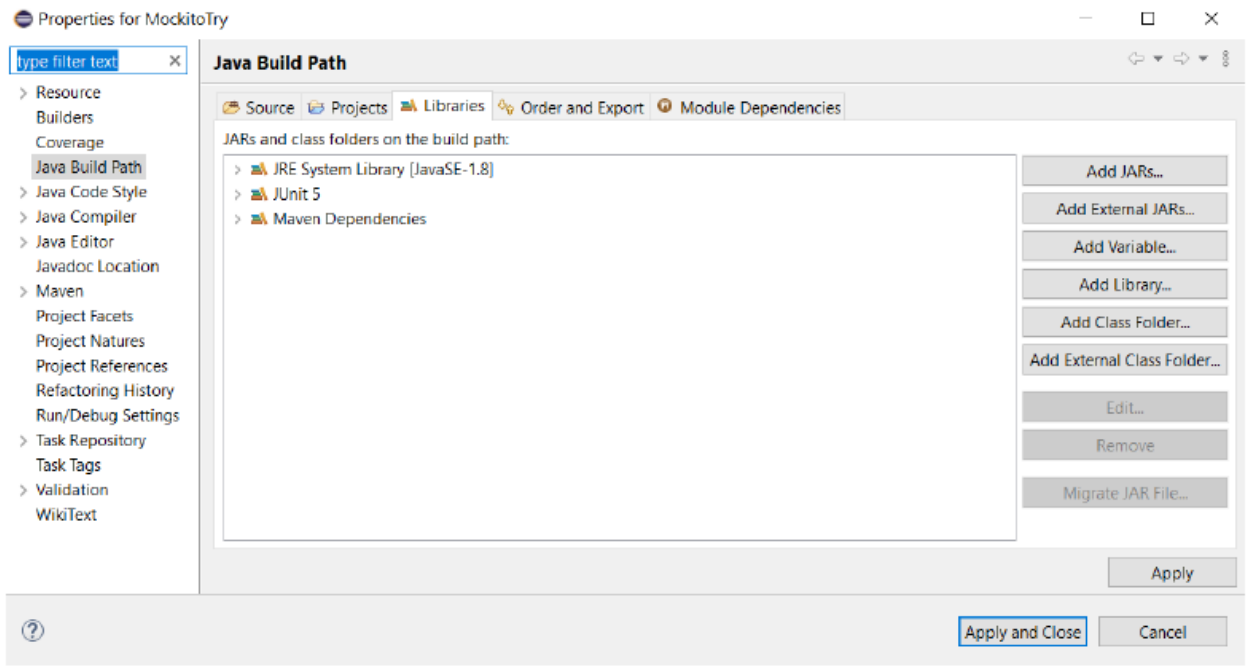
- JUnit is a testing framework for Java projects
- It is a framework with unit-testing functionalities
- It comes integrated with the Eclips development enviornment

Adding JUnit to a project

- Quick fix proposal on @Test directive

Adding JUnit to a project

- Otherwise: Properties – Java Build Path - Libraries



What JUnit does

- JUnit runs a suite of tests and reports the results (test log)

- For each test method in the test suite
  - JUnit executes the method
  - If the method terminates without problems
    - The test is considered passed

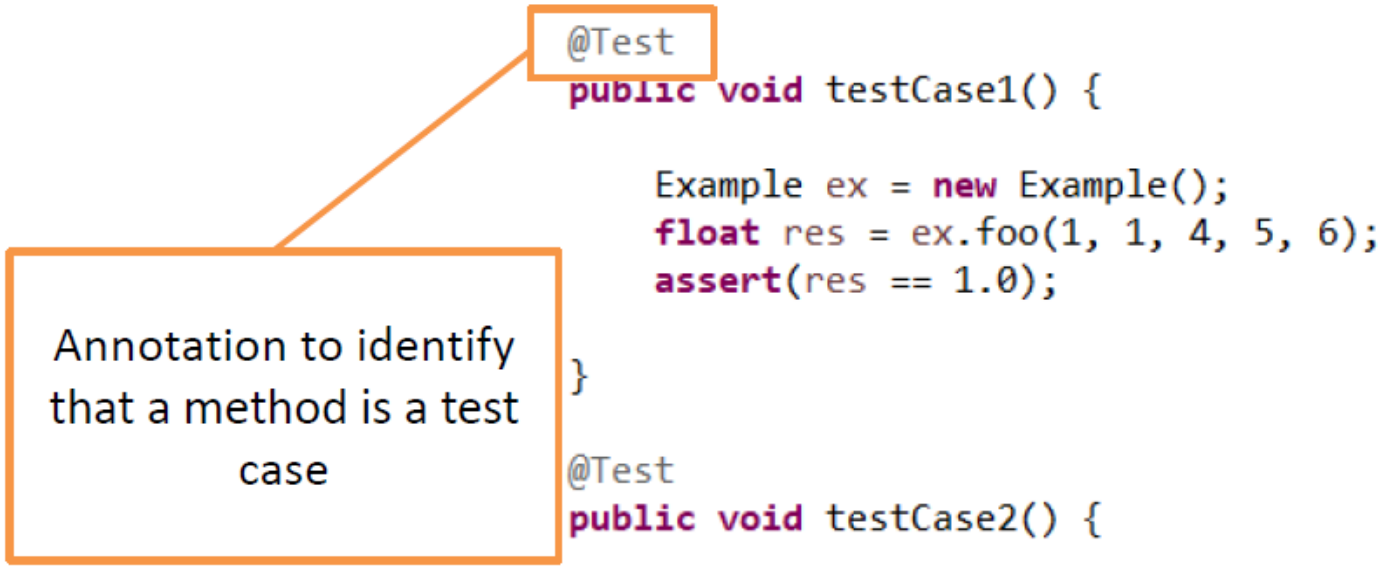
What JUnit does

- JUnit executes each test…() method in the Test class (**non-capital**)
- If the test runs correctly, nothing is done
- If the test fails, the engine throws an AssertionFailedError
  - No need to catch the assertion: the JUnit framework catches it and deals with it for reporting

Sample test cases

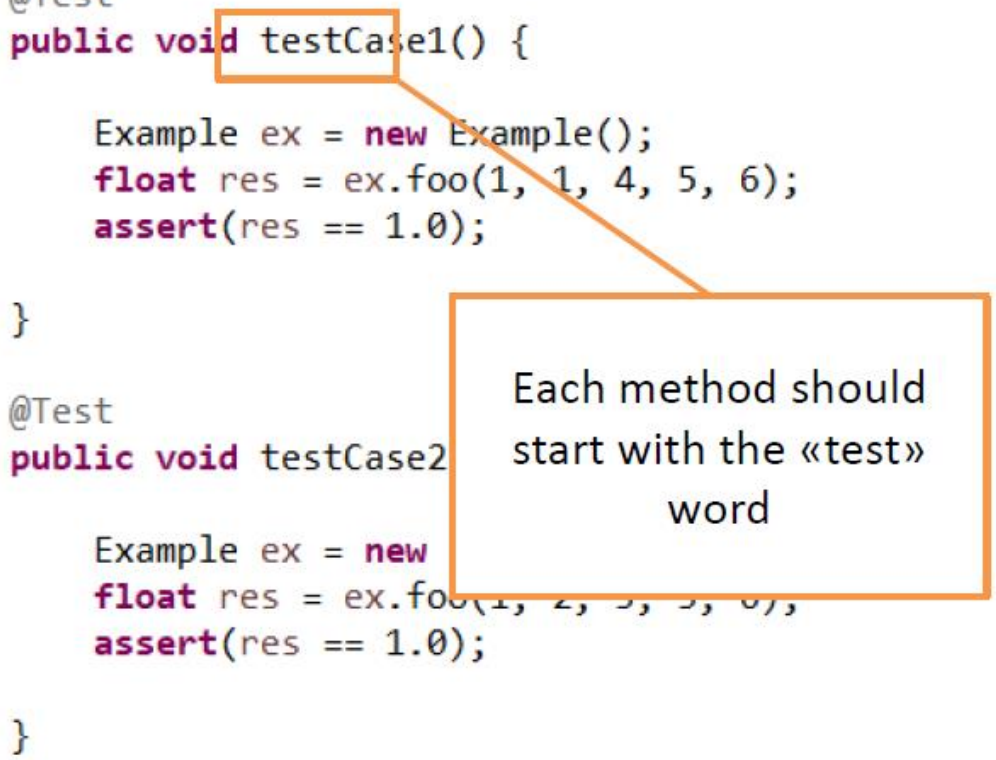
```java
public class Example {

    public boolean bug(int n) {

        if (n == 1) return true;
        else return false;

    }

    public float foo(int a, int b, int c, int d, float e) {

        float res;
        if (a == 0) {
            return 0;
            }
        int x = 0;
        if ((a==b) || ((c == d)) || n) {
            x=1;
        }
        res = 1/x;
        return res;

    }


}
```

Sample test cases




```
@Test
public void testCase1() {

    Example ex = new Example();
    float res = ex.foo(1, 1, 4, 5, 6);
    assert(res == 1.0);

}

@Test
public void testCase2() {

    Example ex = new Example();
    float res = ex.foo(1, 2, 3, 3, 6);
    assert(res == 1.0);

}
```

Sample test cases



JUnit assertions

- Assertions are the primary means of verifying the software functioning with JUnit

- For a condition
  - assertTrue(“message when test fails”, condition)

- If the tested condition is
  - True -> execute the following instruction
  - False -> break the test method, and print out the optional message

JUnit assertions

- For objects, int, long, byte:
  - assertEquals(expected value, expression);
    - E.g. assertEquals(2, stack.size());
- For floating point values
  - assertEquals(expected value, expression, error);
    - E.g., assertEquals(1.0, Math.cos(3.14), 0.01);
- For exceptions:
  - assertThrows(ExceptionClass.class, ()-> { *your code* });

JUnit assertions

- Exception management example:

```
@Test
public void testCase4() {


    Assertions.assertThrows(java.lang.ArithmeticException.class,
            () -> { float res = 1 / 0; }    );



}
```

JUnit assertions

- assertTrue(boolean test)
- assertFalse(boolean test)
- assertEquals (expected, actual)
- assertSame (Object expected, Object actual)
- assertNotSame (Object expected, Object actual)
- assertNull (Object object)
- assertNotNull (Object object)
- Fail()

Before and After methods

- In JUnit 5, the method annotated with @BeforeEach is run before each test…() method
  - Useful to create the needed clean context for the test
- The metod annotated with @AfterEach is run after each test…() method
  - Useful to clean up, when needed

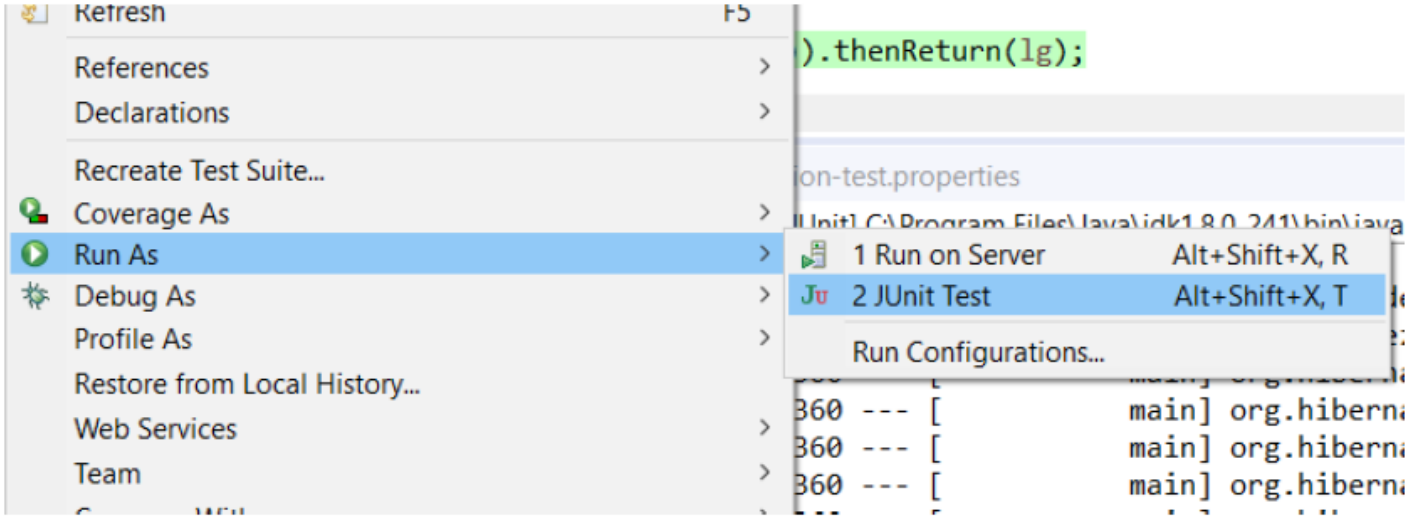
Test suite

- In JUnit 5, you can use the @SelectClasses annotation to combine many test classes when launching a single test class

```
@RunWith(JUnitPlatform.class)
@SelectClasses( { ClassATest.class, ClassBTest.class, ClassCTest.class } )
public class JUnit5TestSuiteExample
{
}
```

Running a test case/suite

- Run
- Run As
- JUnit test



Running a test case/suite