Cheat Sheet for Workshop 3: Code Comprehension

JUnit annotations

`@Test`
Identifies a method as a test method.

`@Test(expected = Exception.class)`
Fails if the method does not throw the named exception.

`@Test(timeout=100)`
Fails if the method takes longer than 100 milliseconds.

`@Before`
This method is executed before each test. It is used to prepare the test environment (e.g., read input data, initialize the class).

`@After`
This method is executed after each test. It is used to cleanup the test environment (e.g., delete temporary data, restore defaults). It can also save memory by cleaning up expensive memory structures.

`@BeforeClass`
This method is executed once, before the start of all tests. It is used to perform time intensive activities, for example, to connect to a database.

`@AfterClass`
This method is executed once, after all tests have been finished. It is used to perform clean-up activities, for example, to disconnect from a database.

JUnit statements

`fail(String message)`
Let the method fail. Might be used to check that a certain part of the code is not reached or to have a failing test before the test code is implemented.

`assertTrue(String message, boolean condition)`
Checks that the boolean condition is true.

`assertFalse(String message, boolean condition)`
Checks that the boolean condition is false.

`assertEquals(String message, expected, actual)`
Tests that two values are the same. Note: for arrays the reference is checked not the content of the arrays.

`assertEquals(String message, expected, actual, tolerance)`
Test that float or double values match. The tolerance is the number of decimals that must be the same.

`assertNull(String message, object)`
Checks that the object is null.

`assertNotNull(String message, object)`
Checks that the object is not null.

`assertSame(String message, expected, actual)`
Checks that both variables refer to the same object.

`assertNotSame(String message, expected, actual)`
Checks that both variables refer to different objects.