Testing phases

• Unit
• Regression
• Integration (interfaces)
• System
• Acceptance
‘Glass box’ unit testing

• Coverage testing: executing all statements, including error handling code

• Structured basis testing
  – Minimal coverage testing
  – Number of tests required
    
    Test = 1;
    
    For each branch top (if, while, case, and, or, for, …)
    
    Test++;
JUnit

• Reusable
• Asserts, e.g.
  Assert.assertTrue(!m12CHF.equals(null));
  Assert.assertEquals(m12CHF, m12CHF);

• See
  – Gamma, Beck, *JUnit Test Infected: Programmers Love Writing Tests*
JUnit Test framework

• TestCases, e.g.

```java
public class MoneyTest extends TestCase {
    //...
    public void testSimpleAdd() {
        Money m12CHF = new Money(12, "CHF"); // (1)
        Money m14CHF = new Money(14, "CHF");
        Money expected = new Money(26, "CHF");
        Money result = m12CHF.add(m14CHF); // (2)
        Assert.assertTrue(expected.equals(result)); // (3)
    }
}
```

• A TestCase subclass implements multiple public void test...() methods
JUnit Test framework, cont.

- The TestCase calls setUp() and tearDown() before each test...() method
- setUp() creates a ‘fixture’
- A TestSuite runs all the tests
- TestSuite extracts the test methods from the TestCase subclass, e.g.
  ```java
  public static Test suite() {
      return new TestSuite(MoneyTest.class);
  }
  ```
JUnit Test framework, cont.

• The TestSuite tracks the results of all the Asserts
• Reports
  – Number of successes
  – Individual failures
• GUI too
Limitations

• What doesn’t the testing we’ve discussed do?