# 什么是单元测试

写了个类，要给别人用，会不会有bug？怎么办？测试一下。

用main方法测试好不好？不好！

1. 不能一起运行！
2. 大多数情况下需要人为的观察输出确定是否正确

# 为什么要进行单元测试

重用测试，应付将来的实现的变化。

提高士气，明确知道我的东西是没问题的。

# JUnit4 HelloWorld

1. new project
2. 建立类
3. 建立testcase

# 放弃旧的断言,使用hamcrest断言

1. assertThat
2. 使用hamcrest的匹配方法
   1. 更自然
3. 示例
   1. assertThat( n, allOf( greaterThan(1), lessThan(15) ) );  
      assertThat( n, anyOf( greaterThan(16), lessThan(8) ) );  
      assertThat( n, anything() );  
      assertThat( str, is( "bjsxt" ) );  
      assertThat( str, not( "bjxxt" ) );
   2. assertThat( str, containsString( "bjsxt" ) );  
      assertThat( str, endsWith("bjsxt" ) );   
      assertThat( str, startsWith( "bjsxt" ) );   
      assertThat( n, equalTo( nExpected ) );   
      assertThat( str, equalToIgnoringCase( "bjsxt" ) );   
      assertThat( str, equalToIgnoringWhiteSpace( "bjsxt" ) );
   3. assertThat( d, closeTo( 3.0, 0.3 ) );  
      assertThat( d, greaterThan(3.0) );  
      assertThat( d, lessThan (10.0) );  
      assertThat( d, greaterThanOrEqualTo (5.0) );  
      assertThat( d, lessThanOrEqualTo (16.0) );
   4. assertThat( map, hasEntry( "bjsxt", "bjsxt" ) );  
      assertThat( iterable, hasItem ( "bjsxt" ) );  
      assertThat( map, hasKey ( "bjsxt" ) );  
      assertThat( map, hasValue ( "bjsxt" ) );

# Failure和Error

1. Failure是指测试失败
2. Error是指测试程序本身出错

# JUnit4 Annotation

1. @Test: 测试方法
   1. (expected=XXException.class)
   2. (timeout=xxx)
2. @Ignore: 被忽略的测试方法
3. @Before: 每一个测试方法之前运行
4. @After: 每一个测试方法之后运行
5. @BeforeClass: 所有测试开始之前运行
6. @AfterClass: 所有测试结束之后运行

# 运行多个测试

# 注意

1. 遵守约定，比如：
   1. 类放在test包中
   2. 类名用XXXTest结尾
   3. 方法用testMethod命名

# 其他框架

TestNG