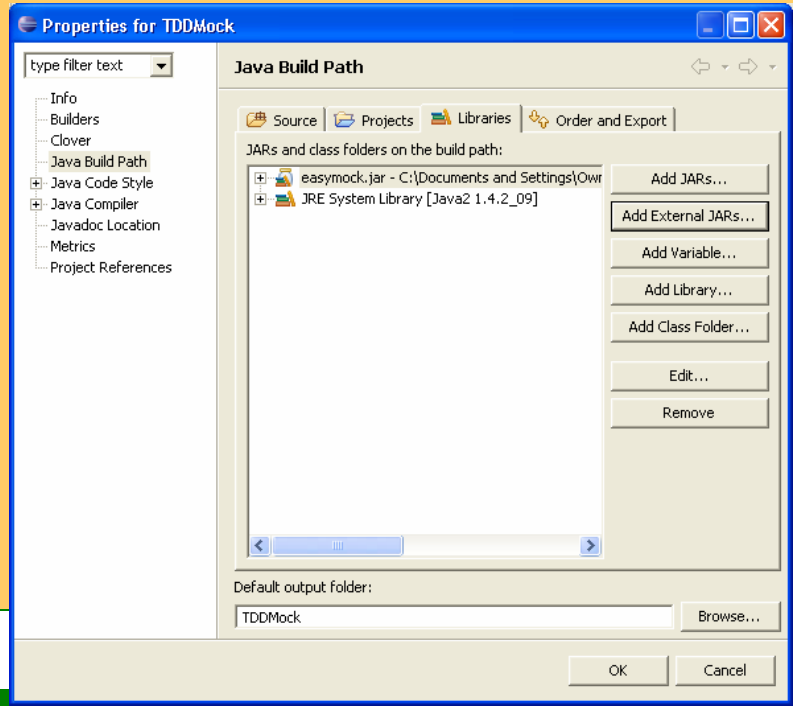
Testing with Mock Objects

- A *mock object* is an object created to stand in for an object that your code will be collaborating with. Your code can call methods on the mock object, which will deliver results as set up by your tests.
- In other words, in a fine-grained unit test, you want to test only one thing. To do this, you can create mock objects for all of the other things that your method/object under test needs to complete its job.

EasyMock

- EasyMock is a third-party library for simplifying creating mock objects
- Download EasyMock1.2 for Java1.3 from www.easymock.org
  - EasyMock 2.x depends on Java 1.5
- Extract download
- Add easymock.jar to project



Testing Bank with EasyMock

```
package bank;
import java.util.Collection;
import org.easymock.MockControl;
import junit.framework.TestCase;
public class TestBank extends TestCase {
   private Bank b;
   private MockControl control;
   private Collection mock;
   protected void setUp() {
      control = MockControl.createControl(Collection.class);
      mock = (Collection) control.getMock();
      b = new Bank(mock);
   }
     public void testNumAccounts() {
            mock.size();
            control.setReturnValue(7);
            control.replay();
            assertEquals(b.getNumAccounts(),7);
            control.verify();
     }
}
```

Import MockControl

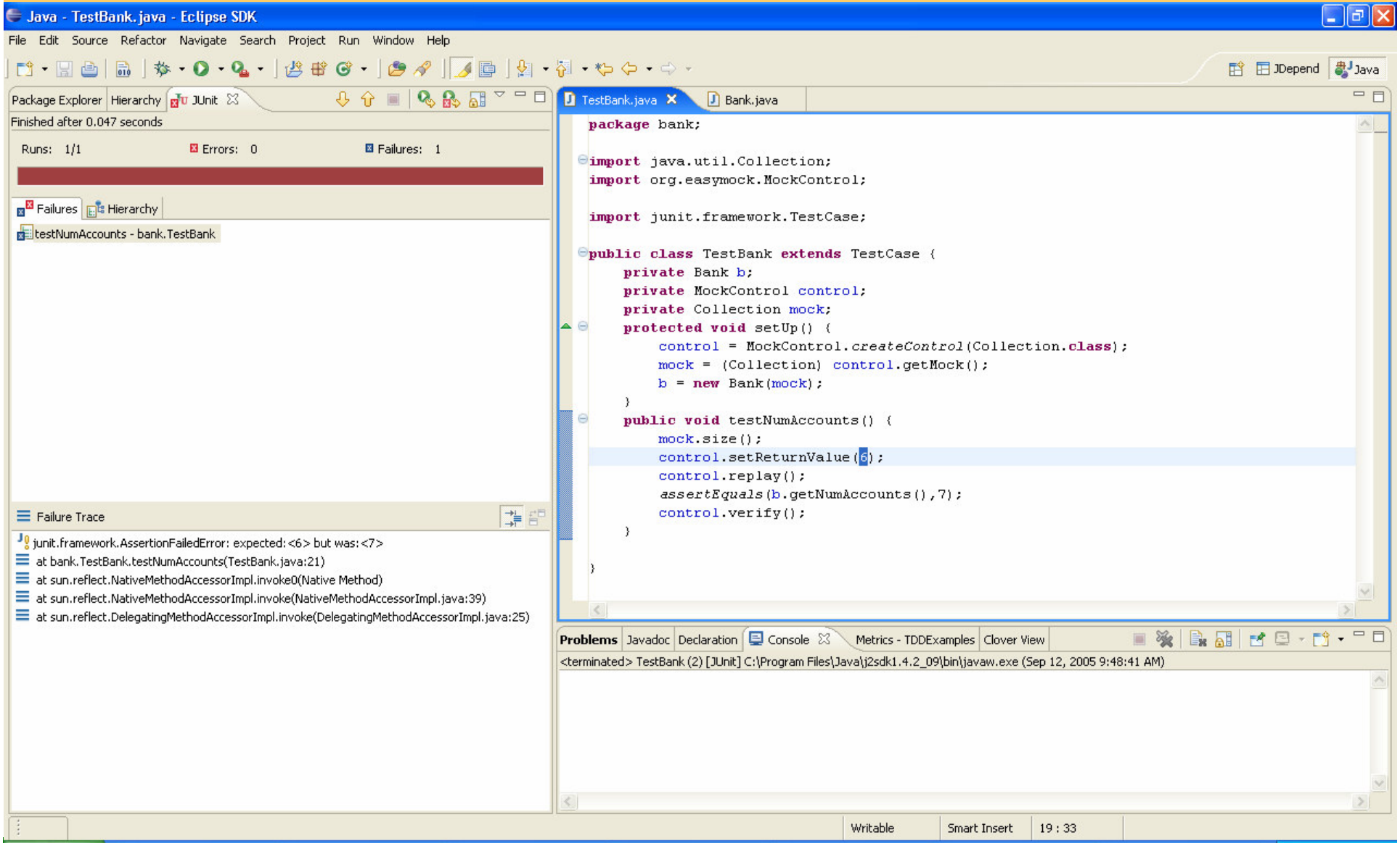
Collection is mock. We want to test Bank, not Collection

Recording what we we expect: size() should be called, returning 7

Turn on mock with replay()

Check expectations with verify()

Failing Expectations



Testing getLargest() with EasyMock

```
package bank;
import java.util.SortedSet;
import org.easymock.MockControl;
import junit.framework.TestCase;
public class TestBank extends TestCase {
    …
    public void testGetLargest() {
        control = MockControl.createControl(SortedSet.class);
        SortedSet mock = (SortedSet) control.getMock();
        b = new Bank(mock);
        mock.last();
        try{
            control.setReturnValue(new Account("Richie Rich",77777,99999.99));
            control.replay();
            assertEquals(b.getLargest().getBalance(),99999.99,.01);
        } catch (Exception e) { fail("testGetLargest should not throw exception"); }
        control.verify();
    }
}
```

last() should be called on mock

When to use Mock Objects

- When the real object has non-deterministic behavior (e.g. a db that is always changing)
- When the real object is difficult to set up
- When the real object has behavior that is hard to cause (such as a network error)
- When the real object is slow
- When the real object has (or is) a UI
- When the test needs to query the object, but the queries are not available in the real object
- When the real object does not yet exist

* from http://c2.com/cgi/wiki?MockObject