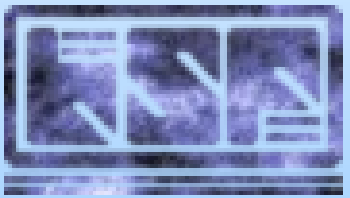
Nástroje pro automatizované testování aplikací

Ing. Roman Nováček, HSF s.r.o.
<novacek@hsf.cz>

4. dubna 2006



Cíl přednášky

Ukázka praktického použití
vybraných nástrojů
pro automatizované testování

Obsah přednášky

1. úvod
2. příklad – správce bankovních účtů
3. JUnit – standardní testovací nástroj
4. DbUnit – testování databázových aplikací
5. mock objekty
6. programování do rozhraní
7. rMock
8. JMock
9. závěr

Automatizované testování

Zásady:

- program musí být od začátek vyvíjen tak, aby bylo možné automatizovaně testovat
- psaní testů musí být snadné
- program musí být udržován testovatelný

Rozsah testů:

- unit testy
- integrační testy
- systémové (celkové) testy

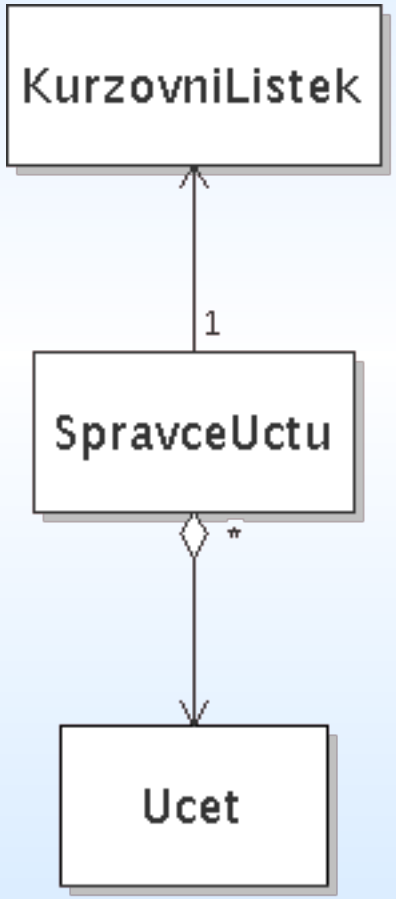
Demonstrační příklad:
Správce bankovních účtů

Datové třídy:

1) **`Ucet`**
   - reprezentován číslem, měnou a aktuálním zůstatkem
   - metoda pro změnu zůstatku

Části systému:

1) **`SpravceUctu`**
   - spravuje účty
   - provádí bankovní operace
2) **`KurzovniListek`**
   - vrací aktuální kurz pro zdrojovou a cílovou měnu



JUnit

- vznikl v roce 2000
- velmi oblíbený nástroj (široká podpora ve vývojových prostředí a nástrojích – např.: Eclipse, ANT, Maven)
- de facto standard (existuje mnoho portací na další platformy – např.: NUnit pro .NET, CppUnit pro C++)

JUnit 3 – použití (1)

```
import junit.framework.TestCase;

public class MyTest extends TestCase {

   protected void setUp() {}
   protected void tearDown() {}

   public void testA() {
      assertXxx(expected, actual);
   }

   public void testB() {}
}
```

JUnit 3 – použití (2)

```
import junit.framework.Test;
import junit.framework.TestSuite;

public class AllTests {

    public static Test suite() {

        TestSuite suite = new TestSuite("AllTests");

        //$JUnit-BEGIN$
        suite.addTestSuite(AllTestsFromClass.class);
        suite.addTest(new SomeTestClass("testMethod"));
        //$JUnit-END$

        return suite;
    }
}
```

JUnit 3 – ukázka (1)

```
public class SpravceUctuImplTest extends TestCase {

  protected void setUp() throws Exception {

    // vytvoreni hard-coded mock objektu
    KurzovniListek kurzovniListekMock = ...;

    // vytvoreni spravce uctu
    spravceUctu = new SpravceUctuImpl(kurzovniListekMock);

    // zalozeni testovacich uctu
    CZK_UCET = spravceUctu.zalozUcet(Mena.CZK, 100.0);
    EUR_UCET = spravceUctu.zalozUcet(Mena.EUR, 10.0);
    UCET_100_CZK = spravceUctu.vratUcet(CZK_UCET);
    UCET_10_EUR = spravceUctu.vratUcet(EUR_UCET);
  }
```

JUnit 3 – ukázka (2)

```
  public void testVyber() throws NeznamyUcetException,
        MaloPenezException {

    assertEquals(100.0, UCET_100_CZK.zustatek());
    spravceUctu.vyber(CZK_UCET, 20.0);
    assertEquals(80.0, UCET_100_CZK.zustatek());

    try {
      spravceUctu.vyber(CZK_UCET, 100.0);
      fail("MaloPenezException nebyla vyhozena");
    } catch (MaloPenezException e) {}

    assertEquals(80.0, UCET_100_CZK.zustatek());
  }
}
```

JUnit 4 (nové vlastnosti)

1. vyžaduje Javu 1.5 (anotace)
2. testovací třída nemusí rozšiřovat `TestCase`
3. testovací metody nemusí mít prefix „test“.
4. metody „assert“ jsou stejné (přímo z třídy `Assert`)
5. anotace `@Test` označuje testovací metody
6. anotace `@Before` / `@After` značí metody pro inicializaci / úklid (před každým testem)
7. anotace `@BeforeClass` / `@AfterClass` značí metody pro inicializaci / úklid (pro celou třídu)
8. anotace `@Test` může mít parameter „timeout“
9. anotace `@Test` může mít parameter „exception“
10. `JUnit4Adapter` pro staré spouštěče JUnit testů

JUnit 4 – použití

```
import org.junit.Test;
import static org.junit.Assert.*;

public class MyTest {
    @Before //nebo @BeforeClass
    protected void setUp() {}
    @After //nebo @AfterClass
    protected void tearDown() {}

    @Test(timeout=10,expected=SomeException.class)
    public void tested() {
        assertXxx(expected, actual);
    }
    @Ignore("Database is down")
    public void ignoredTest() {}
}
```

JUnit – zhodnocení

+ široká podpora ve vývojových nástrojích
+ řada rozšíření (DbUnit, RMock, …)
+ lze použít i pro *test-driven development*
– vytváření a především pak **udržování** skupin testů (`TestSuite`) je obtížné
– nelze definovat vzájemné závislosti testů

DbUnit

- vznikl v roce 2002
- rozšíření JUnit pro testování databázových aplikací
- umožňuje exportovat / importovat data z DBMS do XML souborů (*datasetů*)
- funguje na principu porovnávání aktuálního stavu DB a uloženého *datasetu* s obsahem očekávaným po provedení testované operace

DbUnit – ukázka (1)

```
public class SampleTest extends DatabaseTestCase {

  public IDatabaseConnection getConnection() throws
        Exception {

    Connection jdbcConnection = ...;
    return new DatabaseConnection(jdbcConnection);
  }

  protected IDataSet getDataSet() throws Exception {

    return new FlatXmlDataSet(
        new FileInputStream(new File("dataset.xml")));
  }
```

DbUnit – ukázka (2)

```
  public void testMethod() throws Exception {

    // zde se provadi testovany kod modifikujici DB

    // vytazeni dat z databaze
    IDataSet dbDataSet = getConnection().createDataSet();
    ITable dbTable = dbDataSet.getTable("TESTED_TABLE");

    // nacteni ocekavanych dat
    IDataset expDataset = new FlatXmlDataSet(new File(…));
    ITable expTable = expDataSet.getTable("TESTED_TABLE");

    // overeni zda aktualni tabulka vyhovuje ocekavane
    Asserion.assertEquals(expTable, dbTable);
  }
}
```

DbUnit – zhodnocení

+ vhodné pro aplikace využívající relační databáze
+ možno nasadit i při použití ORM (Hibernate)
+ DBMS může obsahovat uložené procedury
– problémy při změnách datového schématu
– pomalé (před každým testem se připravuje výchozí stav databáze)

Co je Mock objekt?

= speciální verze objektu určena pro testování

- má stejné chování jako skutečný objekt, ale je to jen **atrapa** bez vazeb na další objekty
- před vlastním provedením testu se definuje ***očekávané*** chovaní všech zástupných (Mock) objektů, které se podílejí na testované funkčnosti
- vhodné zejména pro *test-driven development*

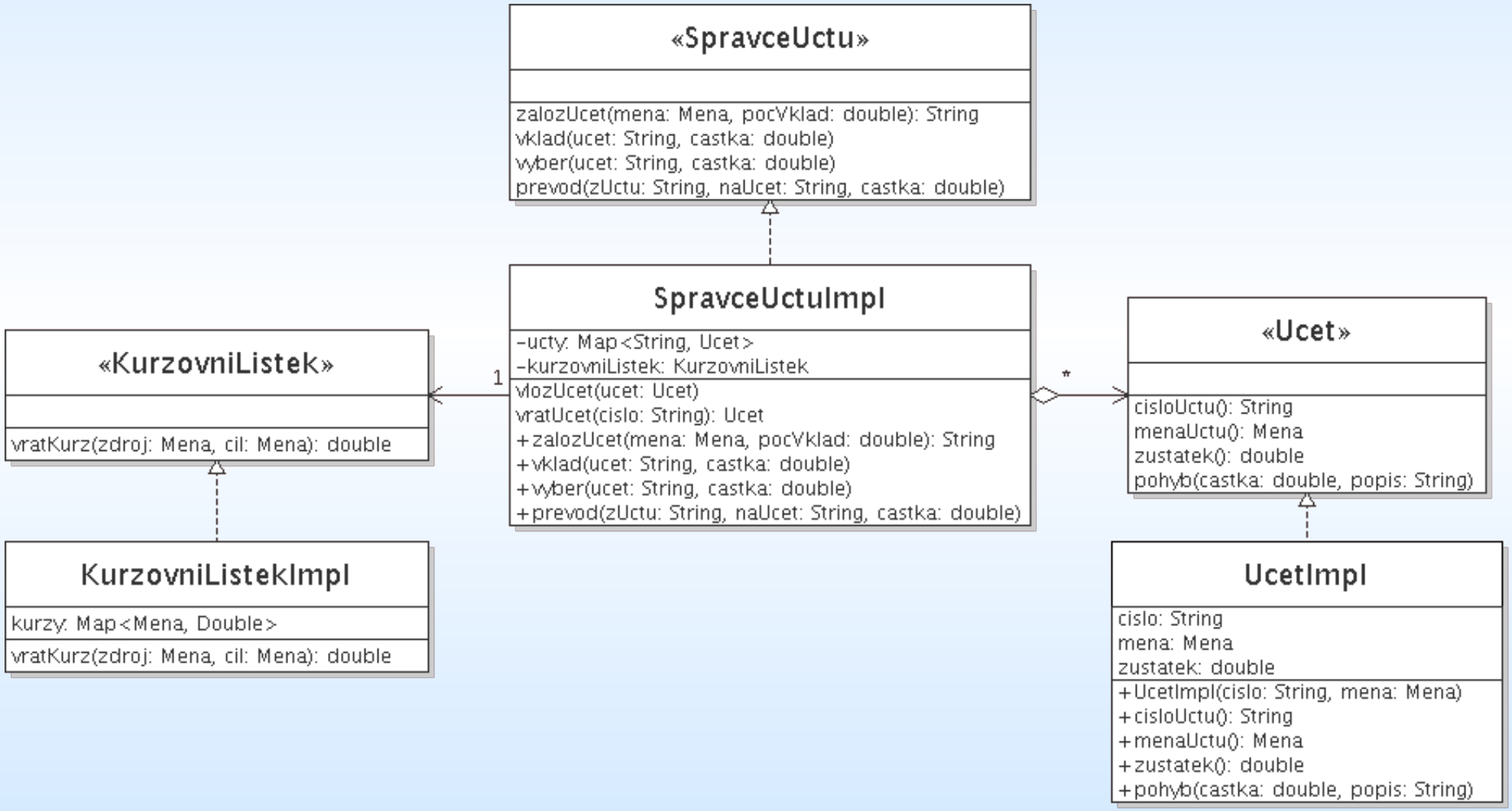
„Ruční“ mock objekt

```
// vytvoreni hard-coded mock objektu
KurzovniListek kurzovniListekMock = new KurzovniListek() {

  public double vratKurz(Mena zdrMena, Mena cilMena) {

    if (zdrMena != Mena.EUR || cilMena != Mena.CZK)
      throw new IllegalArgumentException(
          "zdrMena != EUR || cilMena != CZK");

    return 28.51;
  }
};

// vytvoreni spravce uctu
spravceUctu = new SpravceUctuImpl(kurzovniListekMock);
```

Programování do rozhraní

- pomáhá rozmyslet si a minimalizovat vazby jednotlivých částí systému
- vazby přes interface jsou obecnější
- použití rozhraní je nejvhodnější způsob vytváření Mock objektů
- rozhraní jsou vhodná pro kontejnery na principu *Inversion of Control* (PicoContainer, Spring)

Příklad – UML (programování do interfaců)



SpravceUctuImpl:
metoda prevod

```
public void prevod(String zUctu, String naUcet, double castka) ... {

  if (castka <= 0.0)
    throw new IllegalArgumentException("castka <= 0.0");

  Ucet zdrUcet = vratUcet(zUctu);
  Ucet cilUcet = vratUcet(naUcet);
  Mena zdrMena = zdrUcet.mena();
  Mena cilMena = cilUcet.mena();

  if (zdrMena.equals(cilMena)) {
    // prevod ve stejne mene
    zdrUcet.pohyb(-castka, "prevod na ucet " + naUcet);
    cilUcet.pohyb(castka, "prevod z uctu " + zUctu);

  } else {
    // prevod na jinou menu
    double kurz = kurzovniListek.vratKurz(zdrMena, cilMena);
    zdrUcet.pohyb(-castka, "prevod v jine mene na ucet " + naUcet);
    cilUcet.pohyb(castka*kurz, "prevod z uctu " + zUctu);
  }
}
```

rMock – úvod

- vznikl v roce 2005
- rozšiřuje JUnit
- je to framework na testování pomocí Mock objektů (vytváření a verifikace jejich stavu)
- základní princip funkčnosti:
  1. vytvoření mock objektů (interceptorů)
  2. volání a modifikace očekávaných metod
  3. přepnutí do režimu ověření
  4. spuštění testů nad testovanou třídou
  5. vložení assertů pokud to je nutné
  6. automatické ověření mock objektů

rMock – ukázka (1)

```
protected void setUp() throws Exception {

  // vytvoreni potrebnych mock objektu
  kurzovniListekMock = (KurzovniListek)
      mock(KurzovniListek.class, "kurzovniListek");
  // vytvoreni spravce uctu
  spravceUctu = new SpravceUctuImpl(kurzovniListekMock);

  // zalozeni testovacich uctu
  CZK_UCET1 = spravceUctu.zalozUcet(Mena.CZK, 100.0);
  CZK_UCET2 = spravceUctu.zalozUcet(Mena.CZK, 200.0);
  EUR_UCET = spravceUctu.zalozUcet(Mena.EUR, 10.0);
  UCET_100_CZK =
      ((SpravceUctuImpl)spravceUctu).vratUcet(CZK_UCET1);
  UCET_200_CZK =
      ((SpravceUctuImpl)spravceUctu).vratUcet(CZK_UCET2);
  UCET_10_EUR =
      ((SpravceUctuImpl) spravceUctu).vratUcet(EUR_UCET);
}
```

rMock – ukázka (2)

```
public void testPrevod_EURtoCZK() throws Exception {

  // definice ocekavaneho chovani mock objektu
  kurzovniListekMock.vratKurz(null, null);
  modify().args(is.eq(Mena.EUR), is.eq(Mena.CZK))
      .returnValue(28.51);

  // prepnuti do rezimu verifikace
  startVerification();

  // registrovani potrebnych uctu
  ((SpravceUctuImpl) spravceUctu).vlozUcet(UCET_100_CZK);
  ((SpravceUctuImpl) spravceUctu).vlozUcet(UCET_10_EUR);

  // zavolani testovane funkce a overeni vysledku
  spravceUctu.prevod(EUR_UCET, CZK_UCET1, 1.0);
  assertThat(UCET_10_EUR.zustatek(), is.eq(9.0));
  assertThat(UCET_100_CZK.zustatek(), is.eq(128.51));
}
```

rMock – možnosti

- vytvoření mock objektu (mock, intercept)
  - z rozhraní nebo implementace (nedoporučeno)
- nastavení očekávaných volání
  - voláním odpovídající metody (nezáleží na hodnotách argumentů)
- modifikace volání
  - validace argumentů
    assertThat(arg, is.eq(10)) X assertEquals(arg, 10)
  - modifikace návratové hodnoty
  - nastavení počtu volání (implicitně 1x)
  - nastavení očekávané výjimky
- definování sekcí volání (uspořádané, neuspořádané)
- dynamické vytváření testovacích balíků

rMock – ukázka (3)

```
private Ucet mockUcet(String cislo, Mena mena) {

  Object[] args = {cislo, mena};
  Ucet ucet = (Ucet) intercept(UcetImpl.class, args, "UcetImpl_" + cislo);

  ucet.cisloUctu(); modify().forward();
  ucet.mena(); modify().forward();

  return ucet;
}

public void testPrevod_MockUcty() throws Exception {

  // vytvoreni mock uctu
  Ucet ucet100CzkMock = mockUcet(CZK_UCET1, Mena.CZK);
  Ucet ucet10EurMock = mockUcet(EUR_UCET, Mena.EUR);

  // definice ocekavaneho chovani mock objektu
  kurzovniListekMock.vratKurz(null, null);
  modify().args(is.eq(Mena.EUR), is.eq(Mena.CZK))
      .returnValue(28.51);
```

rMock – ukázka (4)

```
  beginSection(s.ordered("Prevod na jinou menu"));
  {
    ucet10EurMock.pohyb(0.0, "");
    modify().args(is.eq(-1.0), is.startingWith(
        "prevod v jine mene na ucet")).forward();
    ucet100CzkMock.pohyb(0.0, "");
    modify().args(is.eq(28.51), is.startingWith(
        "prevod z uctu")).forward();
  }
  endSection();

  startVerification();

  // registrovani potrebnych uctu
  ucet100CzkMock.pohyb(100.0, "pocatecni vklad");
  ucet10EurMock.pohyb(10.0, "pocatecni vklad");
  ((SpravceUctuImpl) spravceUctu).vlozUcet(ucet100CzkMock);
  ((SpravceUctuImpl) spravceUctu).vlozUcet(ucet10EurMock);

  // vlastni test prevodu
  spravceUctu.prevod(EUR_UCET, CZK_UCET1, 1.0);
}
```

JMock

- jeden z prvních frameworků pro Mock objekty
- široké možnosti pro definování očekávaného chování Mock objektů
- nevýhoda v zadávání názvů metod jako text

```
mock.expects(once()).method("m").with(eq(10));
```

Kdy psát testy?

1. implementace klíčových funkcí
2. oprava chyby (již se nebude opakovat)
3. úpravy existujícího programu
   - implementace nové vlastnosti
   - refactoring
4. test-driven development (testy jako specifikace programu)

Shrnutí

- JUnit základ většiny frameworků pro testování
- rMock je velmi vhodný nástroj pro integrační testování
- vyvíjíme-li aplikaci skládající se z více komponent je velmi vhodné nasadit kontejner na principu *Inversion of Control* (snadná záměna netestovaných částí systému za Mock objekty)

Odkazy

JUnit, CppUnit, TestNG

- http://www.junit.org/
- http://cppunit.sourceforge.net/cppunit-wiki/FrontPage
- http://testng.org/

Mock objekty

- http://en.wikipedia.org/wiki/Mock_Object
- http://www.mockobjects.com/

rMock, JMock, EasyMock

- http://rmock.sourceforge.net/
- http://www.jmock.org/
- http://www.easymock.org/

Konec