Testování softwaru

Sémantická chyba, syntaktická chyba, chyba, debugging, ladění aplikace, výjimka, testování, jednotkové testy

Joseph Soukup – P4

- Nestandartní stavy programu
- Softwarové chyby
- Výjimky
- Testování

Nestandartní stavy programu

- Situace, kdy program produkuje nesprávné nebo neočekávané výsledky nebo se chová nežádoucím způsobem

Hardwarové chyby

- Chyby způsobené selháním hardware

Uživatelské chyby

- Chyby způsobené nekorektními vstupy od uživatele

Programátorské chyby

- Chyby způsobené nesprávně napsaným kódem programu

Programátorské chyby

Syntaktické

- Nesprávný zápis gramatiky programu, programový zápis je nesrozumitelný, u kompilovaných programů je hlášena během překladu
    - Věta nedává smysl

Sémantické

- Program se přeloží, ale končí v nestandartním stavu
    - Věta dává jiný smysl, než jsme chtěli

Neočekávané události

- Stane se něco, co programátor nečekal
    - Během věty nás někdo přerušil

Externí zdroje chyb programu

- Uživatel zadá špatný vstup
- Dojde k porušení hardware
- Neexistující soubor
    - Obecně I/O chyby
- Jiné chyby

Předcházení chyb

- Aktivní
- Ověření dat, stavů a prostředků před tím, než je použijeme, pracné

Zachycení chyb

- Pasivní
- K chybám necháme dojít, pak je řešíme

Výjimky

- Standartní mechanismus pro hlášení chyb

Try-catch

```
try {
  Nebezpečný kód
}
catch
{
  Obsluha všech chyb (= vyhozených výjimek)
}
```

Try-finally

```
try {
  Nebezpečný kód
}
finally
{
  Provede se vždy, ať už k výjimce došlo nebo ne
  např. zavírání souborů
}
```

Try-catch-finally

```
try {
  Nebezpečný kód
}
catch
{
  Obsluha všech chyb (= vyhozených výjimek)
}
finally
{
  Provede se vždy, ať už k výjimce došlo nebo ne
  např. zavírání souborů
}
```

Typy výjimek

```
try {
  Nebezpečný kód
}
catch (FileErrorException ex)
{
  Obsluha výjimky FileErrorException, v ex jsou
  o ní informace
}
catch (StupidUserException ex)
{
  Obsluha výjimky StupidUserException, v ex jsou
  o ní informace
}
```

Vlastnosti výjimek v C#

- Pokud výjimka není zachycena, probublá se do nadřízeného bloku
- Všechny výjimky jsou třídy odvozené od System.Exception
- Pokud dojde k vyhození výjimky, zbytek blogu se ignoruje
- Objekt výjimky obsahuje informace o chybě
- Výjimky jsou standartní způsob předávání informací o chybě nebo nestandardním stavu
- Třídy výjimek by měly být serializované

Vyhození výjimky

- Throw new Systém.Exception()
- Throw new Systém.ArgumentException(„Parameter cannot be null“, „original“)
- Throw new Systém.InvalidOperationException(„Logfile cannout be read-only“)
- Hlášení chyb z našich vlastních tříd

Vlastní výjimka

```
public class InvalidDepartmentException : System.Exception
{
  public InvalidDepartmentException() : base() { }
  public InvalidDepartmentException(string message) : base(message) { }
  public InvalidDepartmentException(string message, System.Exception inner)
: base(message, inner) { }
  protected InvalidDepartmentException(System.Runtime.Serialization.SerializationInfo info, Syste
m.Runtime.Serialization.StreamingContext context) : base(info, context) { }
}
```

Testování software

- Ověření kvality

Testování programátorem

- Unit
- Vrací metody nebo objekty to, co od nich očekáváme

Testování funkcionalit

- Feature
- Dělají části kódu, to, co mají z pohledu uživatele

Integrační testování

- Intergration
- Lze nové vlastnosti zaintegrovat do stávajícího kódu

Systémové testování

- Systém
- Je aplikace funkční

Akceptační testování

- Acceptance
- Ověření klientem

Pilotní testování

- Pilot
- Zkušební provoz

Jednotkové testování v C#

- V projektu máme metody, které chce otestovat
- Víme, co mají, za jakých okolností vracet
- Přidáme do solution nový projekt: C# Unit Test Project
- V testovacích metodách pak definujeme, co má kdy vycházet
- Arrange
  - Nastavení hodnot
- Act
  - Výpočet
- Assert
  - Tvrzení, které má být splněno

```
Assert.AreEqual(expected, actual, 0.001, "Account not debited correctly");
Assert.ThrowsException<System.ArgumentOutOfRangeException>(() =>
account.Debit(debitAmount));
StringAssert.Contains(e.Message,
BankAccount.DebitAmountExceedsBalanceMessage);
Assert.Fail("The expected exception was not thrown.");
```