Wyjątki w C# - skrypt:

Zajęcia laboratoryjne przeprowadzane są na komputerach z systemem operacyjnym Windows 7 z wykorzystaniem oprogramowania Visual Studio 2010 w wersji Ultimate. Poniżej omówiono kilka często wykorzystywanych skrótów:
VS – Visual Studio
LPM – Lewy przycisk myszy
PPM – Prawy przycisk myszy

Zagadnienie 1: Wyjątki
Wszystkie błędy występujące w programach napisanych w języku C# są propagowane za pomocą mechanizmu zwanego wyjątkami. Wyjątki są zgłaszane w momencie jak występuje błąd i pozwalają ten błąd naprawić (obsłużyć). Od momentu zgłoszenia wyjątku jest on propagowany zgodnie ze stosem wywołań do czasu aż zostanie on obsłużony przez instrukcję catch. W przypadku braku obsługi zgłoszonego wyjątku zostaje on obsłużony w sposób domyślny przez system wyświetlając okno dialogowe informujące o wystąpieniu niepoprawnego stanu podczas wykonania aplikacji.
Wyjątki reprezentowane są poprzez klasy dziedziczące z klasy `Exception`. Klasy te określają typ błędu oraz posiadają informacje niezbędne do poprawnego jego obsłużenia. Wywołanie wyjątku, wymaga najpierw utworzenia klasy wyjątku, ustawienia jej parametrów i ostatecznie wywołania go za pomocą słowa kluczowego `throw`.

Przykład:
```
class MyException : Exception
{
	public MyException(string msg)
	{

	}

}
...
	private static void TestThrowException()
	{
		MyException ex =
			new MyException("Custom exception in TestThrowException()");

		throw ex;
	}

...
```

Po zgłoszeniu wyjątku środowisko wykonawcze sprawdza czy kod generujący wyjątek znajduje się wewnątrz bloku try. Jeśli tak, to następuje poszukiwanie odpowiedniego bloku catch obsługującego wyjątek. Jeśli nie ma takiego bloku wyjątek propagowany jest dalej zgodnie z aktualnym stosem wywołań. Po obsłużeniu wyjątku wykonanie programu jest wznawiane kolejną instrukcją po bloku catch. Należy pamiętać, że bloki catch są sprawdzane kolejno zatem w pierwszej kolejności obsługuje się najbardziej szczegółowy wyjątki stopniowo przechodząc do bardziej ogólnych.
Poniżej zamieszczono przykład:

```
static void TestCatch()
{
	System.IO.StreamWriter sw = null;
	try
	{
		sw = new System.IO.StreamWriter(@"C:\test\test.txt");
		sw.WriteLine("Hello");
	}

	catch (System.IO.FileNotFoundException ex)
	{
		System.Console.WriteLine(ex.ToString());  // put the more specific exception
first
	}

	catch (System.IO.IOException ex)
	{
		System.Console.WriteLine(ex.ToString());  // put the less specific exceptions
last
	}
	finally
	{
		sw.Close();
	}

	System.Console.WriteLine("Done");  // this statement is executed after the catch
block
}
```

W zamieszczonym powyżej przykładzie znajduje się również blok finally. Często istotnym jest aby pewien kod wykonywany był niezależnie czy wyjątek wystąpił czy nie. W tym celu kod taki należy umieścić w bloku finally.

Wyjątki – podsumowanie:

- Wyjątki to typ, który dziedziczy po klasie Exception.
- Używaj bloku try wszędzie tam gdzie może wystąpić wyjątek.
- Obsługa wyjątków jest wykonywana poprzez blok catch. Jeśli wystąpi wyjątek sterowanie przekazywane jest do najbliższego na stosie wywołań bloku catch obsługującego zgłoszony wyjątek.
- Jeśli zgłoszony wyjątek nie jest obsługiwany – wówczas program się kończy komunikatem o błędzie.
- Przechwytuje się jedynie wyjątki, które można obsłużyć, pozostawiając aplikację w stanie spójnym.
- Jeżeli blok catch definiuje zmienną typu Exception, wówczas można jej użyć w celu uzyskania dokładniejszych danych na temat warunków jego wystąpienia.
- Obiekt Exception zawiera informacje o stosie wywołań oraz wiadomość tekstową dotyczącą wystąpienia błędu.
- Kod w bloku finally jest wykonywany niezależnie od tego czy wyjątek został zgłoszony czy nie. Używa się go do sprzątania np. zamykania otwartych plików czy strumieni.

Zagadnienie 2: Zasady zgłaszania wyjątków

Kiedy należy zgłaszać wyjątki:

- wykonywana metoda nie może zakończyć działania w sposób poprawny realizując zlecone jej zadanie
- niepoprawne odwołanie do obiektu biorąc pod uwagę jego stan
- argument wywołania metody powoduje wyjątek

Kiedy NIE należy używać wyjątków:

- zmiana wykonania programu w przypadku poprawnego jego działania np. obsługa natrafienia na znak EOF podczas czytania pliku.
- sytuacja zwracania wartości funkcji.
- nie zaleca się zgłaszania wyjątków: System.Exception, System.SystemException, System.NullReferenceException oraz System.IndexOutOfRangeException
-

Zagadnienie 3: Pełna definicja klasy Exception

Aplikacje w języku C# mogą zgłaszać wyjątki systemowe lub zdefiniowane przez programistę. Wyjątki definiowane przez programistę powinny zawierać minimalnie cztery konstruktory: konstruktor domyślny, pozwalający na ustawienie wiadomości tekstowej, ustawiający wiadomość tekstową oraz wyjątek wewnętrzny i ostatni, którym się na razie nie będziemy zajmować (można pomijać).

Przykład:

```
public class InvalidNumberException : System.Exception
{
	public InvalidNumberException() : base() { }
	public InvalidNumberException(string message) : base(message) { }
	public InvalidNumberException(string message, System.Exception inner) :
}
```

Linki:

[1] MSDN: http://msdn.microsoft.com/
[2] C# Practical Learning: http://www.functionx.com/csharp/
[3] C# for beginners: http://www.csharphelp.com/2006/12/c-tutorial-for-beginners/
[4] C# tutorial: http://csharpcomputing.com/Tutorials/
[4] C# tutorials: http://csharp.net-tutorials.com/