WI

Zaawansowane techniki programowania C#

Wykład 2

Piotr Błaszyński

Wydział Informatyki Zachodniopomorskiego Uniwersytetu Technologicznego

18 października 2017

Wyjątki

- służą do informowania o **niepowodzeniu** pewnego elementu programu,
- dawniej używano tzw. wartości osobliwej (takiej, która czymś się wyróżniała) do informowania, że funkcja się nie powiodła,
- więc już dawno (w pewnej formie przed 1970 rokiem) wymyślono wyjątki,

Wyjątki

- w skrócie, w momencie wystąpienia wyjątku, wykonywanie kodu jest przerywane i następuje skok do najbliższego miejsca obsługi wyjątku
  - w C# jest to najbliższe pasujące catch
  - najbliższe oznacza najbliżej położone w kodzie aktualnej funkcji/metody,
  - jeśli w aktualnie wykonywanej funkcji nie ma catch: to przeszukiwana jest funkcja ją wywołująca, zaczynając od momentu wywołania aktualnej funkcji,
  - jeżeli tam też nie ma to są przeszukiwane kolejne wywołania w górę, aż do najwyższego poziomu
- jeżeli nic (pasujące catch) nie zostanie znalezione, to komunikat jest wyświetlany (w uproszczeniu, bo to zależy jeszcze w jaki sposób uruchomiliśmy program).

Wyjątki - podstawowa składnia


```
Try
{
    //kod programu
}
//obsługa sytuacji wyjątkowych,
//catch(0..n) albo finally
```

Wyjątki - Łapanie wszystkich wyjątków


```
Try
{
    //kod programu
}
catch
{
    //obsługa
}
```

Wyjątki - Łapanie konkretnych wyjątków

Wyjątki - Łapanie konkretnych wyjątków (dostajemy obiekt wyjątku)

```
Try{
    //kod programu
}
catch (System.NullReferenceException e){
    //obsługa (można korzystać z obiektu e
}
catch (System.Exception e){
    //pozostałe wyjątki
}
```

Wyjątki - Łapanie konkretnych wyjątków

Wyjątki - Łapanie konkretnych wyjątków (dostajemy obiekt wyjątku)

```
Try{
    //kod programu
}
catch (System.NullReferenceException e){
    //obsługa (można korzystać z obiektu e
}
catch (System.Exception e){
    //pozostałe wyjątki
}
finally{
    //to co musi się wykonać zawsze
}
```

Wyjątki - Przekazanie wyjątku dalej

Przekazanie wyjątku dalej (ale kod w sekcji finally się wykona):

```
Try{
    //kod programu
}
finally{
    //to co musi się wykonać zawsze
}
```

Wyjątki - Przekazanie wyjątku dalej

Przekazanie wyjątku dalej przez jego rzucenie (rethrow):

```
Try{
    //kod programu
}
catch (System.Exception e){
    //jakiś kod
    throw e;
}
```

Wyjątki - Przekazanie wyjątku dalej

Przekazanie wyjątku dalej przez jego rzucenie (rethrow):

```
Try{
    //kod programu
}
catch {
    throw;
}
```

Wyjątki - Przekazanie wyjątku dalej

Przekazanie wyjątku dalej przez jego obudowanie w nowy obiekt:

```
Try{
    //kod programu
}
catch (NullReferenceException e){
throw new NullReferenceException("Obiekt A nie jest
    ustawiony", e);
}
```

Wyjątki - Rzucenie wyjątku

Rzucenie wyjątku:

```
//kod programu dotyczący m.in. wartości a i param
if (a==null)
{
throw new NullReferenceException("Obiekt a nie jest
    ustawiony", param);
}
```

Wyjątki - Własne wyjątki

Własne wyjątki (oparte na dziedziczeniu):

```
public class OwnException:System.
    ApplicationException
{
//dodatkowe elementy
//(np. informacje o obiekcie będącym przyczyną wyją
    tku
    public OwnException(...jakieś parametry...,
        string S) : base(S)
    {
        //operacje wykorzystujące parametry
    }
}
```

Wyjątki - problemy

- Wyjątków trzeba się „spodziewać”, a ...
- ... Nikt nie spodziewa się hiszpańskiej inkwizycji.

Wyjątki - problemy

- Wyjątków trzeba się „spodziewać”, a ...
- ... Nikt nie spodziewa się hiszpańskiej inkwizycji.

Wyjątki - problemy

- Dużo programistów używa wyjątków do „naprawiania” problemów.
- często ignorowane jest informowanie użytkownika, a można (a w zasadzie trzeba) to zrobić na 2 sposoby:
  - poprzez wyświetlenie użytkownikowi informacji w bardziej czytelnej formie niż komunikat wyjątku,
  - zapisanie informacji w pewnym miejscu (np. do logu programu).

System.IO - Obsługa danych

- Stream - Abstrakcyjna klasa bazowa Stream umożliwia odczyt i zapis bajtów.
- FileStream - dokłada do podstawowego zachowania Stream, swobodny dostęp do plików poprzez metodę. Wspiera także operacje synchroniczne i asynchroniczne.
- MemoryStream - Niebuforowany strumień którego dane (kapsułkowane przez niego) są dostępne bezpośrednio w pamięci. Używany jako tymczasowy bufor.

System.IO - Obsługa danych

- BufferedStream - Rodzaj strumienia który dokłada buforowanie do innego strumienia, na przykład NetworkStream. (FileStream ma wbudowane buforowanie, a MemoryStream nie potrzebuje.) Obiekt BufferedStream może wspierać niektóre typy strumieni dla poprawienia wydajności odczytu i zapisu.
- TextReader - Abstrakcyjna klasa bazowa dla klas: StreamReader, StringReader. Klasa (abstrakcyjna) Stream jest przeznaczona do odczytu i zapisu bajtów, natomiast implementacja TextReader jest zaprojektowana do odczytu znaków Unicode.

System.IO - Obsługa danych

- StreamReader – Odczytuje znaki ze strumienia (Stream), przy użyciu wybranego kodowania (Encoding) w celu konwersji znaków z i do bajtów.
- StringReader – odczytuje znaki z obiektu typu String. Dzięki StringReader możliwe jest potraktowanie łańcucha znaków tak jak strumienia.

System.IO - Obsługa danych

- TextWriter - Klasa bazowa (abstrakcyjna) dla klas: StreamWriter, StringWriter. Klasa (abstrakcyjna) Stream jest przeznaczona do odczytu i zapisu bajtów, natomiast implementacja TextWriter jest zaprojektowana do zapisu znaków Unicode.
- StreamWriter - Zapisuje znaki do strumienia (Stream), przy użyciu wybranego kodowania (Encoding) w celu konwersji znaków do bajtów. Dzięki StringWriter możliwe jest potraktowanie łańcucha znaków tak jak strumienia.
- BinaryReader – Odczytuje binarne dane ze strumienia.
- BinaryWriter - Zapisuje binarne dane do strumienia.

System.IO - Ścieżki

- System.IO.Directory – przestrzeń nazw do obsługi operacji związanych ze strukturą katalogową plików (pobranie listy plików, dodanie i usunięcie katalogu, itd.)
- System.IO.Path – klasa do obsługi operacji na ścieżkach do plików (pobranie samej nazwy, samego rozszerzenia, itd.)

System.IO - Uprawnienia

Uprawnienia – FileIOPermission. W przykładzie poniżej obiekt f2 opisuje uprawnienia do odczytu `C:\testR.txt` a później również może opisywać uprawnienia do odczytu i zapisu do pliku `C:\testRW.txt`.

```
FileIOPermission f2 = new FileIOPermission(
    FileIOPermissionAccess.Read, "C:\\testR.txt");
f2.AddPathList(FileIOPermissionAccess.Write |
    FileIOPermissionAccess.Read, "C:\\testRW.txt");
```