WI

Zaawansowane techniki programowania C#

Wykład 3

Piotr Błaszyński

Wydział Informatyki Zachodniopomorskiego Uniwersytetu Technologicznego

26 października 2017

Nowości w środowisku

- VS.NET 2005 - .NET Framework 2.0
- Edit and Continue,
- Wizualizatory,
- Gotowe bloki kodu (snippets), otaczanie bloku kodu (surround),
- Projektant klas,
- Automatyczne testowanie (w wersji Team Edition),
- Refaktoring,
- inne usprawnienia,

Nowości w języku

- Dzielenie implementacji na części (partial types),
- Aliasy (aliases),
- Klasy statyczne (static classes),
- Różny dostęp do właściwości (property access modifiers),
- Obsługa wartości null ($Nullable < T >$, $T?$),
- Zmiany w obsłudze delegacji (delegate),
- Ułatwienia w implementacji iteratorów (yield),
- Programowanie generyczne, abstrakcyjne typy danych (generics)

Dzielenie implementacji na części (partial types)

- Słowo kluczowe partial,
- Zastosowanie:
  - Oddzielenie interfejsu od implementacji (jak w C++)
  - Podział na mniejsze pliki,
  - Oddzielenie części automatycznie generowanej od właściwego kodu
- using odnosi się tylko do pliku, w którym wystepuje,
- Brak gwarancji na kolejność wykonywania inicjalizacji zmiennych, nie wolno na tym polegać
- Modyfikatory dostępu (public, private itd.) nie mogą być różne, wystarczy określić dostęp w jednym miejscu

Aliasy (aliases)

- Pozwalają na umieszczenie tych samych nazw w obrębie jednego assembly,
- Mało sytuacji, kiedy są rzeczywiście potrzebne,
- Alias global określający główną przestrzeń nazw

Klasy statyczne (static classes)

- Nie ma potrzeby implementacji konstruktora,
- Pożyteczne przy klasach użytkowych (np. System.Math),
- Nie mogą być dziedziczone, nie mogą zawierać składowych dla instancji,

Różny dostęp do właściwości (property access modifiers)

- Tylko dla metody get lub set, nie dla obu jednocześnie,
- Dokładamy modyfikator zwiększający „restrykcyjność”

Obsługa wartości null (*Nullable* $<$ *T* $>$, *T*?)

- Typ? – gotowe typy z obsługa wartości null,
- *System.Nullable* $<$ *Typ* $>$ - dodaje do typu obsługę wartości null,
- Operator ?? – jeśli wartość wybierana jest null-em, przypisuje inną wartość: c = a??b jest równoznaczne z: if(a==null) then c=b; else c=a;

Zmiany w obsłudze delegacji (delegate)

- Anonimowe metody
- Drugie znaczenie słowa kluczowego delegate,
- Dostęp do zmiennych

Ułatwienia w implementacji iteratorów (yield),

- yield return
- yield break

Programowanie generyczne, abstrakcyjne typy danych (generics)

- Szybsze znajdowanie błędów, już w czasie kompilacji,
- Szybsze działanie programu, nie ma potrzeby poszukiwania odpowiedniego typu,
- Gotowe podstawowe struktury danych,

```
public class Dictionary<T,K>
where T: IComparable<T>
where K: BaseClass, ICloneable
{ //...
}
```

Programowanie generyczne, abstrakcyjne typy danych (generics)

- *System.Collections.Generic* - wcześniej *System.Collections*
- *Comparer* $<T>$ - *Comparer*
- *Dictionary* $<K, T>$ - *HashTable*
- *List* $<T>$, *LinkedList* $<T>$ - *ArrayList*
- *Queue* $<T>$ - *Queue*
- *SortedDictionary* $<K, T>$ - *SortedList*
- *Stack* $<T>$ - *Stack*