

## 6.Структури от данни в .NET

Част втора Generics



## Съдържание (1/2)

- Какво е Generic?
- Разлики между Generics и шаблоните в C++
- Деклариране
  - Generic класове
  - Generic методи
- Повече от един параметър
- Ограничения



## Съдържание (2/2)

- Как да използваме Generic класове и методи?
- Generic интерфейси IEnumerable<T>,
   ICollection<T>, IList<T>, IDictionary<T,V>
- Важни Generic класове
- Generic колекции и списъци vs. нетипизирани колекции и спъсъци?



#### Generic?

#### • В превод:

#### **GENERIC**

- 1. родов
- 2. общ
- 3. обширен, с широко приложение

• В програмирането



## Generics vs. Templates (1/2)

Generics в C# не са толкова гъвкави като шаблоните в C++.

- Само типове могат да са параметри
- Не се поддържат специализации
- Не се поддъжат частични имплементации



## Generics vs. Templates (2/2)

- Не се поддъжат стойности по подразбиране
- Кодът трябва да е валиден за всеки тип, с който може да бъде заместен параметъра
- Получаването на крайния (CLR) тип се получава по време на изпълнение



- С помощта на Generics можем да имплементираме функционалност без да конкретизираме тип
- Типът който не конкретизираме е параметър
- Могат да се параметризират както класове така и методи



- Един клас е generic, като след името му следва <T>, където Т е параметър
- В рамките на класа това име означава тип

```
class GenericClass<T>
{
    public static T GetValue()
    {
       return default(T);
    }
}
```



- Поддъжат се и generic методи в рамките на клас който или е, или не е generic
- Това става като след името на метода следва <T>, където Т е име на параметър

```
class NonGenericClass
{
    public static T GetGenericValue<T>()
    {
        return default(T);
    }
}
```

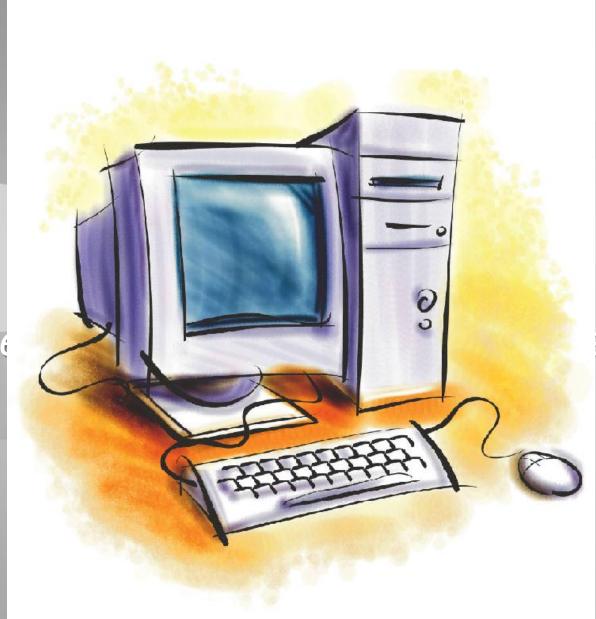


 Обръщение - <T> след името, където Т е име на тип

 Обръщения към дефинираните на предходния слайд класове/методи

```
GenericClass<int>.GetValue();
NonGenericClass.GetGenericValue<int>();
```





se u



#### По-сложни сценарии

• Повече параметри - <T, R, E, ...>

 Ограничения на типове, които могат да заместят параметъра – оказват се посредством ключовата дума where



## Ограничения на параметрите

- where T: struct
- where T: class
- where T: new()
- where T: <base class name>
- where T: <interface name>
- where T: U





тър;

(поє



## Generic интерфейси?

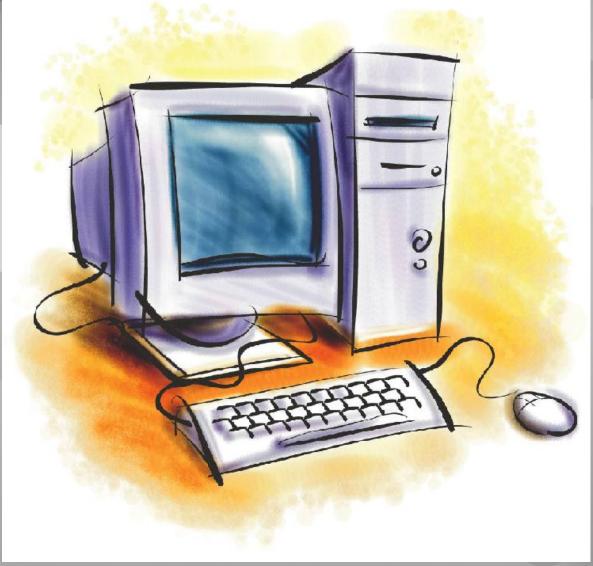
 Generic могат да бъдат не само методите и класове, но и интерфейсите

- IEnumerable<T> vs. IEnumerable
- ICollection
   T> vs. Icollection

• Всъщност System.Collections.Generic е новата версия на System.Collections...









## Generic колекции за всички!

Generic колекциите са силно типизирани. От това следват следните неща:

- Сигурни са не могат да съдържат елемент, който не е от посочения тип
- Не предизвикват излишен boxing и unboxing на елементите си
- Не ни е нужно да правим преобразования
- Съдържат повече функционалност (понеже са създадени по-късно)



# Въпроси?

