Jazyk C# a platforma .NET

Katedra softwarového inženýrství
Fakulta informačních technologií
České vysoké učení technické v Praze

Interface IEnumerable

BI-DNP

```
using System;
using System.Collections;

// priklad mirne modifikovany z Microsoft Help

// interface IEnumerable slouzi napr. pro realizaci cyklu foreach

// pomocna trida (jeji instance tvori prvky interniho pole
// tridy ("kolekce") People
public class Person {
    public Person(string fName, string lName) {
        this.firstName = fName;
        this.lastName = lName;
    }

    public string firstName;
    public string lastName;
}

// verze s detailni implementaci IEnumerable a IEnumerator

/*
public class People : IEnumerable {
    private Person[] _people;

    // konstruktor (vlozi prvky sveho parametru (pole)
    // do pole _people)
    public People(Person[] pArray) {
        _people = new Person[pArray.Length];

        for (int i = 0; i < pArray.Length; i++) {
            _people[i] = pArray[i];
        }
    }

    // metoda GetEnumerator (jedina metoda interface IEnumerable)
    // vraci instanci libovolne tridy, ktera implementuje
    // interface IEnumerator

    // IEnumerable a IEnumerator mohou byt i generiky
    // (s parametrickym typem <T>)

    IEnumerator IEnumerable.GetEnumerator() {
        return new PeopleEnum(_people); // PeopleEnum definovano dale
    }
}
```

```
// PeopleEnum - trida, ktera implementuje interface IEnumerator
// (ma metody Reset, MoveNext a Current)
// tyto metody staci k tomu, aby kompilator byl schopen
// realizovat foreach cyklus

public class PeopleEnum : IEnumerator {
    public Person[] _people;

    // Enumeratory jsou nastaveny na pozici pred 1. element,
    // dokud neni poprve zavolana metoda MoveNext()
    int position = -1;

    public PeopleEnum(Person[] list) {
        _people = list;
        // zde zjednodusuji - priradim jen pointer
        // (pole potom lze zvenku zmenit!!)
    }

    public bool MoveNext() {
        position++;
        return (position < _people.Length);
    }

    public void Reset() {
        position = -1;
    }

    public object Current {
        get  {
            // vraci aktualni prvrk
            try {
                return _people[position];
            }
            catch (IndexOutOfRangeException) {
                throw new InvalidOperationException();
            }
        }
    }
}
*/
```

```
// verze pouzivajici yild

public class People : IEnumerable
{
    private Person[] _people;
    public People(Person[] pArray)
    {
        /*
        _people = new Person[pArray.Length];

        for (int i = 0; i < pArray.Length; i++) {
            _people[i] = pArray[i];
        }
        */

        _people = pArray;  // take funguje, ale nekopiruje prvky
    }

    public IEnumerator GetEnumerator() {
        for (int i = 0; i < _people.Length; i++) {
            // pomoci yield je nahrazena predchozi pracna realizace
            yield return _people[i];
        }
    }

    // nebo IEnumerable neimplementovat pro People, ale pouzit takto:
    /*
    public IEnumerable GetPeople() {
        for (int i = 0; i < _people.Length; i++) {
            yield return _people[i];
        }
    }

    // potom cyklus pisi takto:
    foreach (Person p in peopleList.GetPeople()) { … }
    */
}
```

```
class App
{
    static void Main()
    {
        Person[] peopleArray = new Person[3]
        {
            new Person("Franta", "Novak"),
            new Person("Venca", "Vopicka"),
            new Person("Eulalie", "Prekrasna"),
        };

        People peopleList = new People(peopleArray);

        // trida, ktera implementuje interface
        // IEnumerable je pouzitelna v cyklu foreach:

        // pro variantu implementace IEnumerable pro People
        foreach (Person p in peopleList)
        // pro variantu s GetPeople
        // foreach (Person p in peopleList.GetPeople())
               Console.WriteLine(p.firstName + " " + p.lastName);

        Console.ReadLine();

    }
}

// Vystup:

// Franta Novak
// Venca Vopicka
// Eulalie Prekrasna
```