

# Specifičnosti jezika C#

Programiranje korisničkih interfejsa

Bojan Furlan

# Provera aritmetičkog prekoračenja

- ▶ Podrazumevano se ne vrši provera

```
checked {  
    int number = int.MaxValue;  
    Console.WriteLine(++number);  
}
```

**OverflowException**

Baca se izuzetak,  
WriteLine se ne izvršava.

```
unchecked {  
    int number = int.MaxValue;  
    Console.WriteLine(++number);  
}
```

MaxValue + 1 < 0 ???

**-2147483648**

# foreach konstrukcija

- Kolekcija mora da implementira interfejs **System.Collections.IEnumerable**, tj. *GetEnumerator()* metod

- nizovi implementiraju System.IEnumerable, pa se i kroz njih može iterirati

```
ArrayList numbers = new ArrayList( );  
for (int i = 0; i < 10; i++ ) {  
    numbers.Add(i);  
}  
  
foreach (int number in numbers) {  
    Console.WriteLine(number);  
}
```

**0 1 2 3 4 5 6 7 8 9**

# switch konstrukcija

- Podržani tipovi : integer, char, enum ili **string**

```
switch (color)
{
    case "Black":
        ...
        break;
    case "Red":
    case "Blue":
        ...
        break;
    default:
        ...
        break;
}
```

# Meанизам prosleđivanja parametara

# Mehanizam prosleđivanja parametara

## Tri načina prosleđivanja parametara

<b>ulazni</b>	Po vrednosti (podrazumevano)
<b>ulazno - izlazni</b>	Po referenci
<b>izlazni</b>	Izlazni parametri (po referenci)

# Prosleđivanje po Referenci - ref

## ■ Prosleđuje se referenca na mem. lokaciju

- Mora se poklapati po tipu (za razliku od (&) u C++)
- Mora se dodeliti vrednost pre poziva

```
void MyMethod()  
{  
    int num1 = 7, num2 = 9;  
    Swap(ref num1, ref num2);  
    // num1 = 9, num2 = 7  
}  
void Swap(ref int x, ref int y)  
{  
    int temp = x; x = y; y = temp;  
}
```

# Izlazni Parameteri - out

- Kao ref, jedino što nije dozvoljeno dodeliti vrednost pre poziva

```
static void OutDemo(out int p, out int q)
{
    // ...
}

//...

int m,n;
OutDemo(out m,out n);
```



# Prosleđivanje promenljivog broja parametara

- Uvek se prosleđuje po vrednosti
- Deklariše se kao niz uvek na kraju liste parametara
- Type-safe za razliku od C++

```
static long Sum(params long[ ] v)
{
    long total, i;
    for (i = 0, total = 0; i < v.Length; i++)
        total += v[i];
    return total;
}
static void Main( )
{
    long x = Sum(63,21,84);
}
```

# Guidelines for Passing Parameters

## ■ Mechanisms

- Pass by value is most common
- Method return value is useful for single values
- Use **ref** and/or **out** for multiple return values
- Only use **ref** if data is transferred both ways

## ■ Efficiency

- Pass by value is generally the most efficient

# Naredba zaključavanja

- Ključna reč *lock* (*synchronized* u Javi)
  - ne može koristiti kao modifikator metoda
  - može se koristiti samo kao naredba

```
public void Metod()  
{  
    lock (NekiObjekat)  
    {  
        // naredbe koje zahtevaju sinhronizovani  
        // pristup objektu  
    }  
}
```