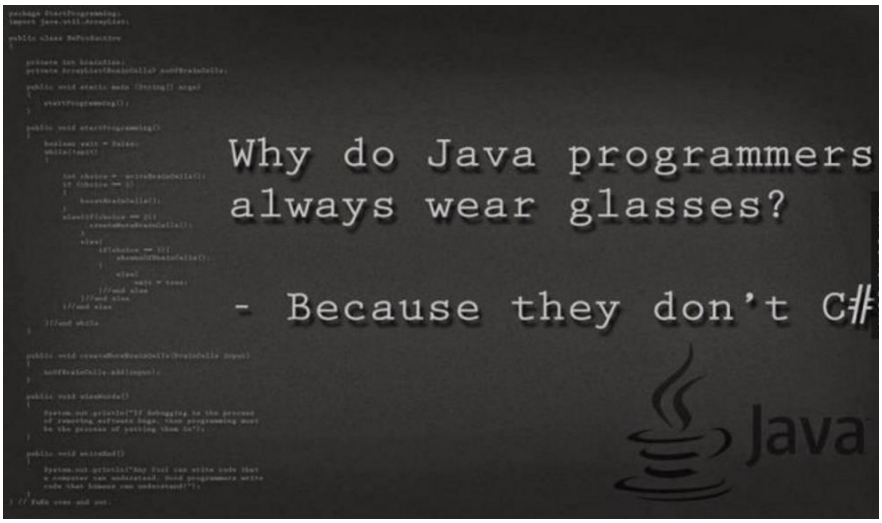
UVOD - OD JAVE DO C#



ELEMENTARNE RAZLIKE

Veliki broj Java/C# razlika su uglavnom preimenovane ključne reči i razlike u konvencijama imenovanja. Neke od tih jezičkih razlika su:

- Java kod se prevodi u .class fajlove koji se opciono mogu pakovati u .jar fajlove. Ovi fajlovi se mogu izvršiti pomoću Java virtuelne mašine. C# kod se prevodi u biblioteke (.dll fajlove) i izvršne fajlove (.exe fajlove). Izvršni fajlovi se mogu izvršiti direktno iz Windows operativnog sistema kao i ostali fajlovi. Biblioteke se ne mogu izvršiti one se samo koriste u izvršnim fajlovima. Slično kao i kod Jave, da bi se izvršavao C# kod na računaru je potrebno instalirati odgovarajući .Net okvir.
- I u Javi i u C# jeziku se klase mogu podeliti po grupama. U jednoj grupi se ne mogu naći dve klase sa istim imenom. U Javi se grupe klasa nazivaju paketi (npr. **package** java.util), dok C# ima imena prostora (npr. **namespace** System.Collection)
- Kada u jednoj grupi želite da koristite klase iz druge grupe one se neće videti. Da bi se u jednoj grupi koristile klase iz druge grupe morate eksplicitno definisati koje grupe se koriste. U Javi se to radi tako što se importuje paket (npr. **import** java.util), dok se u C# se koristi (**using** System.Collection)
- U Javi imena paketa i metoda se obično pišu malim slovom, dok se u C# jeziku pišu velikim početnim slovom.
- Obično se interfejsima u C# jeziku dodaje slovo I kao prefiks - na primer IComparable, IList i slično.
- Ako želite da ne dozvolite da se neke klasa dalje nasleđuje u Javi ćete koristiti **final** dok ćete u C# jeziku koristiti **sealed**.

Sličnost je najlakše predstaviti u konkretnom primeru. U sledećem kodu je prikazana jednostavna Java konzolarna aplikacija:

```
package aplikacija;
import  java.util.*;
import java.lang.*;

public final class Primer
{
    public static void main(string[] args)
    {
        System.out.println( (new DateTime(2012, 3, 27)).toString() );
    }
}
```

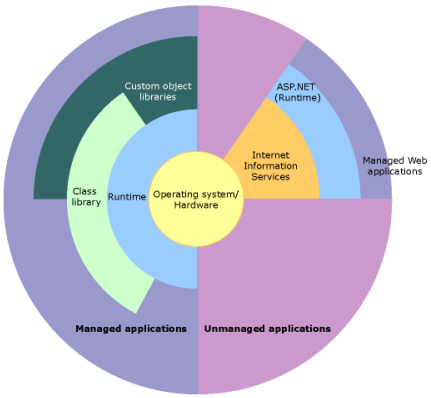
```
namespace aplikacija
{
    using System;
    using System.Collection;

    public sealed class Primer
    {
        public static void Main(String[] args)
        {
            Console.WriteLine( (new Date(2016, 9, 28)).ToString() );
        }
    }
}
```

Kao što možete da vidite osnovna sintaksa je skoro identična. Ako naučite razlike izmedu jezika lako možete da pređete sa jednog jezika na drugi.

.NET OKVIR

Microsoft .NET Framework (Slika 1) je softverska platforma koja može biti instalirana na računarima koje pokreće *Microsoft Windows* operativni sistem (postoji i verzija za Linux - http://www.monodevelop.com/). On uključuje veliki broj gotovih biblioteka kodova za uobičajene probleme u programiranju i virtuelnu mašinu koja upravlja izvršavanjem programa pisanih specijalno za *.NET Framework*. Dostupan je na svim programskim jezicima koji .Net Framework podržava (VB.NET, C#...). Više o *Microsoft .Net Framefork*-u možete pročitati ovde.



Slika 1 – Arhitektura .NET okvira

Microsoft .NET Framework je komponenta koja omogućava izvršavanje programa, dok je za razvoj programa potrebno instalirati i *Microsoft SDK* (engl. Microsoft Softwere Development Kit). Radi lakšeg razvoja aplikacija potrebno je instalirati i Visual Studio koji predstavlja veoma moćno razvojno okruženje koje olaksava razvoj .NET aplikacija.

RAZLIKE U JEZIČKIM KONSTRUKCIJAMA

Pored elementarnih preimenovanih jezičkih konstrukcija neki elementi se malo više razlikuju. Ovde će biti prikazani neki primeri jezičkih konstrukcija koje se malo više razlikuju i ne mogu se svrstati u jednostavno preimenovanje.

ČITANJE SA KONZOLE

I Java i C# imaju jednostavne metode za ispis na konzolu System.out.println() odnosno Console.WriteLine(), međutim Java nema ekvivalentnu metodu za čitanje. Ako želite da čitate nešto sa konzole moraćete da uzmete System.inn polje, konvertujete ga u InputStreamReader, a onda njega u BufferedReader, pa onda da koristite readLine metodu ovog objekat kao što je prikazano u sledećem primeru:

```
BufferedReader console = new BufferedReader(new InputStreamReader(System.in));
String linija = console.readLine();
```

Pošto je u konzolarnim aplikacijama ovo česta operacije, u C# vam obezbeđuje metodu klase Console koja bez ikakvog konvertovanja vraća string sa konzole:

```
String linija = Console.ReadLine();
```

Jedan od razloga zašto je Microsoft napravio Jezik koji je sličan Javi ali pod drrugim imenom je upravo mogućnost da ga menja i uvodi ovakve male ali značajne olakšice a da ne zavisi od Java standarda. U nastavku ćete videti još nekoliko zanimljivih razlika koje će vam olakšati rad.

NASLEĐIVANJE

Oba jezika podržavaju nasleđivanje jedne osnovne klase i implementaciju proizvoljnog broja interfejsa samo se sintaksa malo razlikuje. U javi biste mogli da izvedete klasu kao u sledećem primeru:

```
class Trougao extends GeometrijskaFigura
    implements Cloneable, Comparable
    {

    }
```

U **extends** sekciji možete da stavite jednu klasu a u **implements** proizvoljan broj interfejsa. C# ekvivalent je prikazan u sledećem primeru:

```
class Trougao: GeometrijskaFigura, Cloneable, Comparable
{

}
```

Kao što možete videti u C# jeziku je dovoljno nabrojati klase i interfejse. Kompajler će sam proveriti da li je samo jedan element u listi klasa a ostali interfejsi.

POZIVANJE KONSTRUKTORA NADKLASE

- **Java**

```
class Zivotinja
{
        String ime;

        Zivotinja(String ime)
        {
                this.ime = ime;
        }
}

class Pas extends Zivotinja
{
        String rasa;

        Pas(String rasa, String ime)
```

```
        {
                super(ime);
                this.rasa = rasa;
        }

}
```

- **C#**

```
public class Zivotinja
{
    String ime;

    public Zivotinja(String ime)
    {
        this.ime = ime;
    }
}

class Pas : Zivotinja
{
        String rasa;

        Pas(String rasa, String ime) : base(ime)
        {
                this.rasa = rasa;
        }

}
```

FOREACH PETLJA

U Javi, for petlja može da se koristi ili u standardnom obliku ili za prolazak kroz kolekciju promenljivih kao što je prikazano u sledećem listingu:

```
double[] brojevi = {1.2, 3.0, 0.8};
int zbir = 0;
for (double broj : brojevi)
{
    zbir += broj;
}
```

C# ne omogucava da se for petlja koristi na ovaj način ali ima posebnu foreach petlju koja je ekvivalentna ovoj konstrukciji:

```
double[] brojevi = {1.2, 3.0, 0.8};
int zbir = 0;
foreach (double broj in brojevi) {
    zbir += broj;
}
```

PROMENLJIVI BROJ ARGUMENATA METODE

U Javi možete deklarisati metodu koja prihvata proizvoljan broj parametara odredenog tipa. Na primer ako želite da deklarišete funkciju koja sabira proizvoljan broj celobrojnih argumenata, u Javi je možete napisati na sledeći način:

```
static int sum(int ... numbers)
{
    int total = 0;
```

```
        for (int i = 0; i < numbers.length; i++)
        total += numbers [i];

        return total;
    }
```

Promenljiva numbers sadrži niz brojeva koji će biti sabrani. C# ekvivalent je prikazan u sledećem listingu:

```
    static int Sum(params int[] numbers)
    {
       int total = 0;
       for (int i = 0; i < numbers.Length; i++)
            total += numbers [i];
        return total;
    }
```

U oba jezika pozivi metoda su manje više identični:

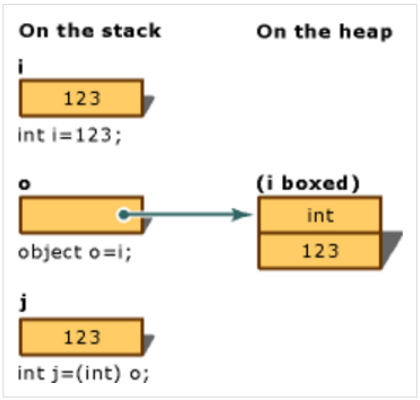
```
    int s1 = Sum(1,2,3);
    int s2 = Sum(1,2,3,9,8,7);
```

BOKSOVANJE

I Java i C# vam omogućavaju automatsku konverziju prostih vrednosnih tipova u referentne i obrnuto kao što je prikazano u sledećem primeru:

```
    int i = 123;
    Object o = i;
    x=321;
    int j = (int)o;//j==123
```

Kada definišete neku celobroju promenljivu (npr. lokalnu promenljivu neke metode koja se nalazi na steku) i dodelite je nekom objektu, i Java i C# će napraviti novi objekat tipa java.lang.Integer odnosno System.Int32, staviti ih na hip i postaviti referencu objektu. Kada se objekat na hipu dodeli nekoj drugoj promenljivoj, vrednost će biti kopirana u njega.



Primetite da kada se promenljiva prebaci na hip postaje potpuno nepovezana od originala. Kada promenite originalnu vrednost u promenljivoj

Ovaj proces se u C# jeziku naziva boxing odnosno unboxing.

Jedina razlika je u činjenici da se u Javi objekat na hipu predstavlja kao lava.lang.Integer, dok se u C# jeziku predstavlja kao System.Int32.

```
static void NullMe(Int32 n)
{
    n = 0;
}


static void NullMe(Object n)
{
    n = 0;
}

static void Main(string[] args)
{
    int x = 1;
    Object o = x;
    x=2;

    Console.WriteLine("o("+ o.GetType().FullName + ") = " + o);//1
    Console.WriteLine("x = " + x);//2

    System.Int32 n = (System.Int32)o;
    System.Int32 m = n;
    n = 5;

    Console.WriteLine("o = " + o);//1
    Console.WriteLine("n = " + n);//5
    Console.WriteLine("m = " + m);//1

    NullMe(n);
    Console.WriteLine("n = " + n);//5

    NullMe(o);
    Console.WriteLine("o = " + o);//1
}
```

Primetite da se boksovani objekti prenose po vrednosti kada se predaju funkcijama. Funkcije NullMe postavljaju predate vrednosti na nulu, ali se time menja samo vrednost predata kao argument, dok originalne vrednosti u pozivaocu ostaju nepromenjene.