

ՅՈՒՆԱՅԵՐՅ ԵԿԺՅՐԻՆԳՈՒՄՅ

ԽՆՅՊԵՐԿՈՑԿՑ ԻՐՅԹՈՑՎՑ Զ ՅՈՒՆԱՅԵՐՅԳՈՒՄ

ՈՒՆՑՅ ՄՈՒՊԵՐԵ

Otvoreno računarstvo

Programski jezici i otvorenost

- **povijest**
- prenosivost i skalabilnost
- smanjenje složenosti
- normiranost i licenciranje

Mario Žagar



The background features large, light green letters 'O' and 'R' that are partially visible, framing the central text.

Povijest

Motivacija

- Što radi kôd u četvrtom retku (C)?

```
char  dest[10]={'a','b','c','d','\0'};
char  src[10]={'e','f','g','\0'};
memcpy(dest, src, strlen(src));
*((char *) (dest+strlen(src)))=0;
```

- Terminira niz znakova *null*-znakom
 - očigledno????
 - mogućnost grješke???
 - redoslijed operatora???

Motivacija

- Što se dogodi u slučaju grješke (VBA)?

```
Private Sub NekiButton_Click()
    DoCmd.SetWarnings False
    On Error GoTo NekiButtonClick_Err
    Dim cmd As ADODB.Command
    cmd.Execute
    ...
NekiButtonClick_Exit:
    Exit Sub
NekiButtonClick_Err:
    Resume NekiButtonClick_Exit
End Sub
```

Motivacija



```
void f(void)
{
    void* s;
    s = malloc(50);
    return;
}
```

```
int main(void)
{
    while (1) f();
    return 0;
}
```

- Što će se dogoditi s memorijom?

Dosadašnji primjeri – jesu li ...?



- Neka od svojstava idealnog programskog jezika su:
 - jednostavnost
 - jasnoća
 - apstraktnost
 - izražajnost
 - fleksibilnost
 - efikasnost

Jesu li to svojstva dosadašnjih primjera????

Kratka povijest

- 1970-ih godina
 - Lisp dobiva *garbage collection*
 - Virtualni stroj za Pascalov *p-code*
 - B (Ken Thompson, Bell Labs), kasnije i C (Dennis Ritchie, Bell Labs)(tzv. K&R)
 - 1971. UNIX u assembleru za PDP-11
 - 1973. UNIX u C-u
 - nroff, troff - lijepi ispis teksta **temeljen na oznakama**
- 1979. *C with classes* (Bjarne Stroustrup, Bell Labs)
- 1983.-1985. C++ (Bjarne Stroustrup, Bell Labs)
- 1980-ih (sredinom) postoje:
 - **razredi (klase), iznimke, nasljeđivanje, sakupljanje smeća (*garbage collection*), virtualni stroj i C/C++ sintaksa**

Kratka povijest

- 1991. SUN, James Gosling
 - Oak, kasnije Java,
 - prva javna inačica u svibnju 1995.
- Microsoft – kupuje OmniVM
 - nastaje J++,
 - kasnije .NET i C#
- Koncepti postoje dugo vremena, tehnologije evoluiraju primjenom istih!

Strojni i zbirni jezici

- U samim počecima programiranja
 - programiranje u strojnom jeziku
 - svaka naredba = strojni kod
 - **npr. 00000101**
- Sljedeći korak
 - zbirni jezik (assembler)
 - mnemonička reprezentacija strojnog koda
 - **npr. 00000101 = DEC B (Zilog Z80)**

Makro naredbe

- Makro naredbe:

- Nameću normu pisanja i korištenja procedura
- Sprječavaju ponavljanje koda

- npr. "jump if greater than" makro (8051 assembler):

```
%*DEFINE (JGT(VALUE, LABEL))
    (CJNE A, # $VALUE+1, $+3      ; JGT
    JNC $LABEL)
```

- poziv – da li akumulator sadrži ASCII kod veći od "Z"?

```
%JGT('Z', GREATER_THAN)
```

- prevodi se u:

```
CJNE A, #5BH, $+3      ; JGT
JNC    GREATER_THAN
```

Viši programski jezici

- kodiranje u zbirnom jeziku
 - neefikasno, sporo, sklono grješkama
- rješenje
 - viši programski jezici
- poplava programskih jezika 1960-ih
 - preživjeli samo najpopularniji:
 - Algol
 - BASIC
 - Cobol
 - FORTRAN
 - Lisp
 - itd...

Viši programski jezici

- moćnija računala
 - potreba za moćnijim i izražajnijim programskim jezicima
- rješenje
 - nadogradnja postojećih novim mogućnostima
- zadržati kompatibilnost
 - npr. FORTRAN 2003 kompatibilan s prvim FORTRAN-om iz 1956.!!!).
- rezultat
 - dodatna kompleksnost jezika
- rješenje
 - počni od početka!

Viši programski jezici

- **BCPL (1960) -> B (1970.)-> C (1972.)**
- **C+1 = C++ (1985.)**
 - objektno orijentiran, fleksibilan, moćan, temelji se na C sintaksi
- C/C++ namijenjen pisanju systemske programske podrške (a NE aplikativne!):
 - efikasan
 - direktan pristup memoriji
 - prilagođen ograničenim resursima
 - jednostavan prevoditelj
 - jezični konstrukti se efikasno mapiraju u zbirni jezik
 - kombiniranje koda u C-u i koda u zbirnom jeziku

Oak



- Sun, **1992.**, Star7
 - prototip naprave za upravljanje kućnim uređajima
 - 15,24 cm (6") LCD *touchscreen*, bežični radio 200 kbps, grafičko sučelje u 4 MB FLASH memorije
 - SunOS + interpreter za novi jezik
 - **Oak**
 - razredi (klase) stižu bežično
 - putem radio veze



Java



- Star7 – namijenjen digitalnoj kablskoj televiziji
 - tržište nespremno za potpuno interaktivnu televiziju
- **WWW** mora biti
 - pouzdan, siguran, neovisan o arhitekturi
 - osobine novog programskog jezika
- preorijentacija projekta na World Wide Web
 - preglednik WebRunner (kasnije HotJava)
 - skidanje i izvršavanje razreda s Interneta
- Javin applet
 - prvi dinamički sadržaj unutar Web preglednika



Java



- Novi programski jezik savršeno odgovara WWW-u
- Web preglednik
 - uz prikazivanje statičnih slika i teksta, po prvi put može izvršavati aplikacije (pisane u Javi)
- 1995. Netscape ugrađuje Javu u svoj preglednik
- 1996. Sun izdaje Java Development Kit (JDK)
- Java se razvija i raste
 - ideja prenosivosti i skalabilnosti
 - PicoJava
 - Java ME
 - Java SE
 - Java EE
 - realtime Java



Applet - Nyquistov teorem


Microsoft



- ...sredinom 1990-ih:
 - Windows 95 + Office
 - najpopularniji OS i uredski paket na PC-ima
 - Microsoft ignorira Internet
 - spajanje na Mrežu otkriva probleme sigurnosti unutar OS-a,
 - kao i makro mogućnosti kod Office aplikacija (Word)
 - Windows API neprimjenjiv na malim uređajima
 - Windows CE uređaji spori

- ...sredinom 90-ih:
 - *open source* i *free software* pokreti
 - potiču kooperativni razvoj programske podrške
 - prijetnja komercijalnim aplikacijama
 - **Javini razvojni alati (JDK) za Windowse**
 - omogućeno pisanje aplikacija u elegantnom jeziku s ugrađenom sigurnošću i spajanjem na Internet

- Uspjeh Jave - reakcije:
 - Microsoft kupuje OmniVM,
 - nastaje J++,
 - 2002. g. izlazi Visual Studio .NET
 - nova arhitektura (svi programski jezici izvode se na istom virtualnom stroju),
 - novi programski jezik (C#)
 - C# - autor Anders Hejlsberg (Turbo Pascal, Borland Delphi)

The background of the slide features large, light green letters 'O' and 'R' that are partially visible and cut off by the edges of the frame. The text 'Prenosivost i skalabilnost' is centered over these letters.

Prenosivost i skalabilnost

Definicije

- **Prenosivost**
 - mogućnost izvršavanja istog programa na različitim platformama/operacijskim sustavima
- **Skalabilnost**
 - mogućnost pisanja aplikacija za ugrađena računala i za poslužitelje istim alatima odnosno programskim sučeljem
 - mogućnost izvođenja iste aplikacija na malim i velikim računalima

Prenosivost

- Doba assemblera
 - programski jezik je osobina procesora na kojem se izvodi (interna arhitektura, skup naredaba, itd...)
- Normirani C
 - prevodi se korektno na različitim platformama
 - knjižnice funkcija (pristup mreži, korištenje niti) često nisu normirane
- Java
 - prva donijela platformsku neovisnost cijelog jezika i pripadnih biblioteka (potpuna i zaokružena cjelina)
- Kako?
 - Java se ne prevodi u strojni kod ciljne platforme

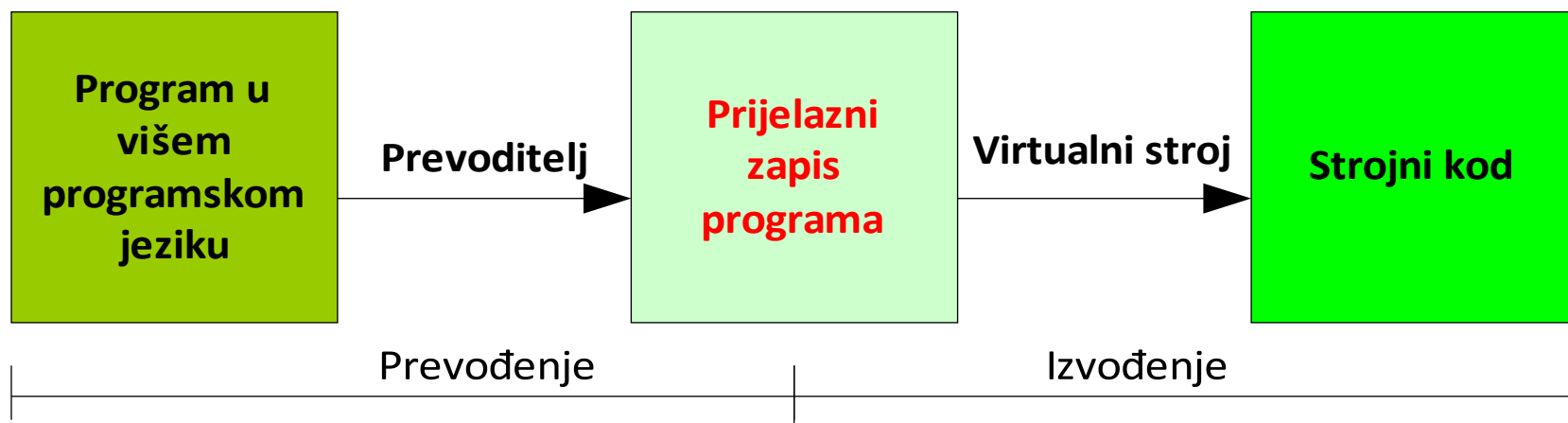
Prenosivost



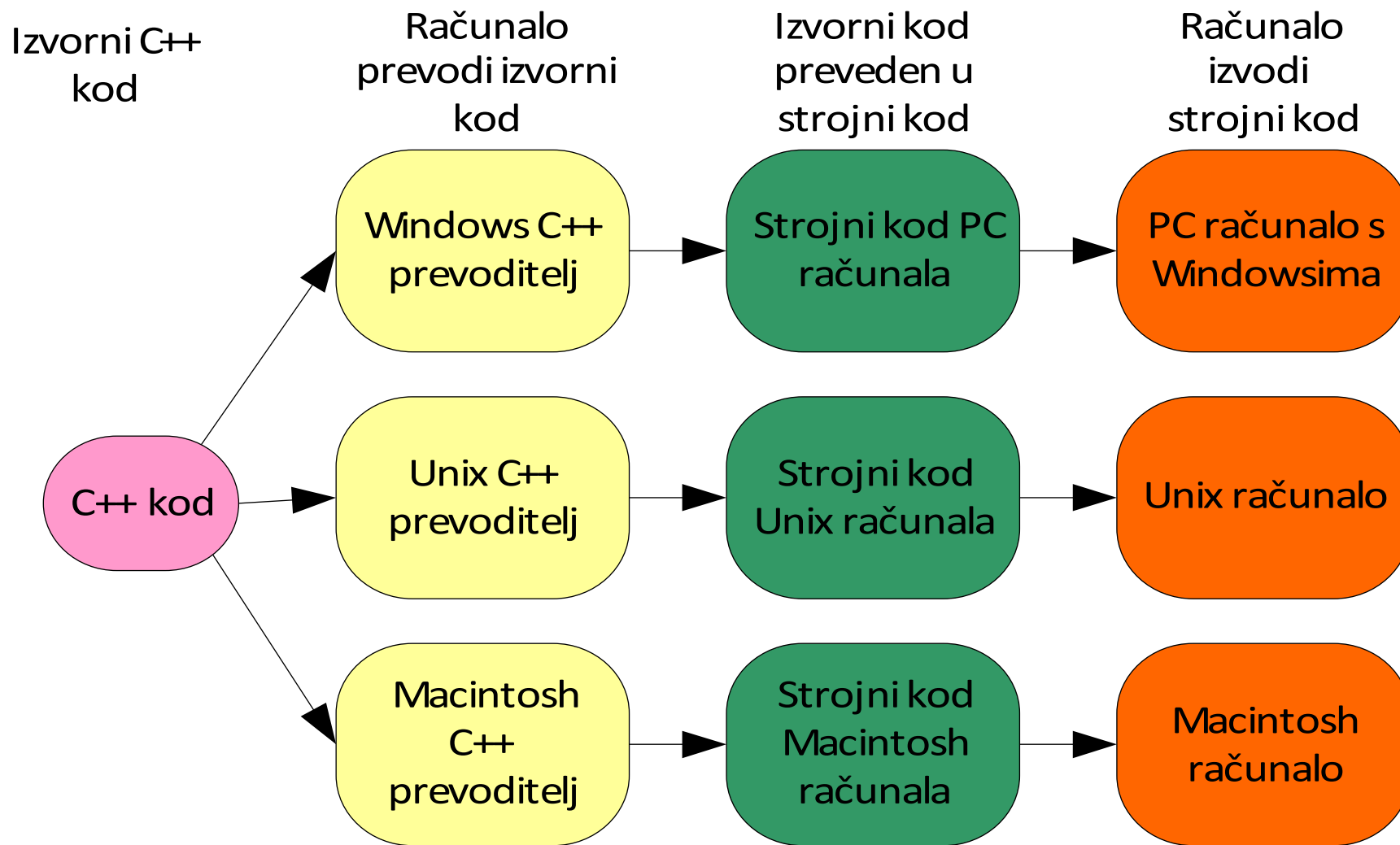
Stari pristup:



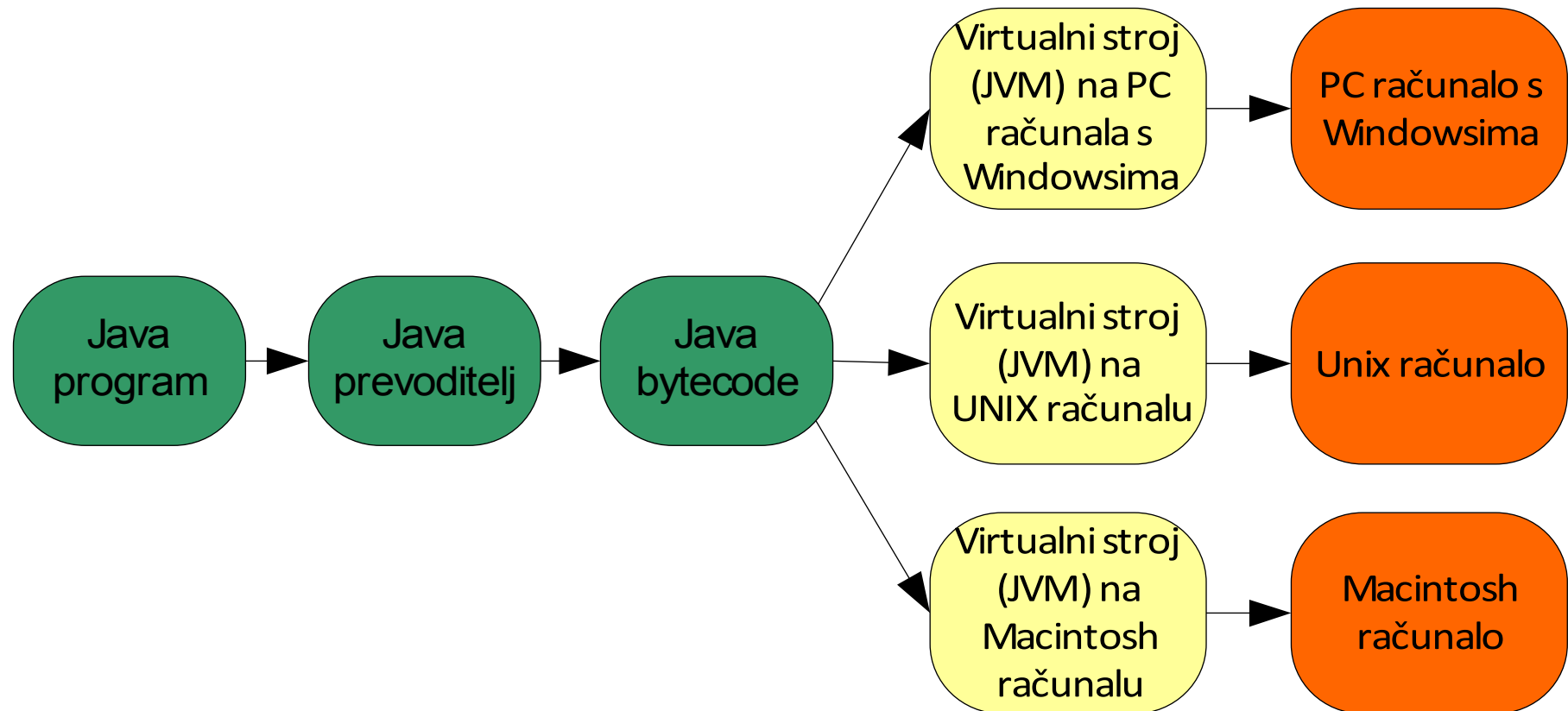
Novi pristup:



Stari pristup - Primjer



Novi pristup - Primjer



Prevođenje i tumačenje

- Viši programski jezik se:
 - prevodi u strojni kod ili
 - interpretira
- Tumačenje (interpretacija) je sporo
 - prevođenje programa u prijelazni (intermediate) zapis
- Prijelazni zapis se:
 - prevodi u strojni kod ili
 - interpretira
- Prijelazni zapis omogućuje neovisnost o platformi

Prijelazni zapis programa

- Generira se
 - jednom (*Java bytecode*),
 - prije svakog izvođenja (Perl, Ruby) ili
 - prilikom svake promjene izvornog koda prije izvođenja (Python)
- Neovisan je o platformi na kojoj se izvodi
- Lakši za otkrivanje grješaka
 - lakši je povratak do izvornog koda
- Omogućuje dinamičko određivanje
 - dosega (*scoping*) varijable
 - tipa (*typing*) varijable

Prijelazni zapis programa

- prijelazni zapis je:
 - orijentiran na stog
 - objektni
 - niže razine apstrakcije (od izvornog koda)
 - prilagođen izvođenju unutar virtualnog stroja
- Java
 - **bytecode**
- jezici .NET platforme
 - **MSIL/CIL** (**M**icrosoft/**C**ommon Intermediate Language)

Javin *bytecode*



- public class Zbroji
{
 public static void main(String[] args)
 {
 int a=42,b=24,c;
 c=a+b;
 }
}
- prevođenje Zbroji.java -> Zbroji.class
 javac Zbroji.java
- prevođenje u suprotnom smjeru (disassembler)
 Zbroji.class -> JVM assembler
 javap -c Zbroji ->

Bytecode



Compiled **from** "Zbroji.java"

```
public class hr.rasip.or.primjer.Zbroji
  extends java.lang.Object{
  ...
  public static void
  main(java.lang.String[]);
```

HEX:	Code:
10 2A	0: bipush 42
3C	2: istore_1
10 18	3: bipush 24
3D	5: istore_2
1B	6: iload_1
1C	7: iload_2
60	8: iadd
3E	9: istore_3
B1	10: return
	}

MSIL/CIL



```
class Zbroji
{
    static void Main(string[] args)
    {
        int a = 42, b = 24, c;
        c = a + b;
    }
}
```

- prevođenje -> Zbroji.exe
- ildasm Zbroji.exe ->

MSIL/CIL



- .method **private** hidebysig
static void Main(string[] args) cil managed
{
 .entrypoint // Code size 12 (0xc)
 .maxstack 2
 .locals init ([0] int32 a, [1] int32 b,
 [2] int32 c)
 IL_0000: nop
 IL_0001: ldc.i4.s 42
 IL_0003: stloc.0
 IL_0004: ldc.i4.s 24
 IL_0006: stloc.1
 IL_0007: ldloc.0
 IL_0008: ldloc.1
 IL_0009: add
 IL_000a: stloc.2
 IL_000b: ret
} // end of method Zbroji::Main

Bytecode vs CIL

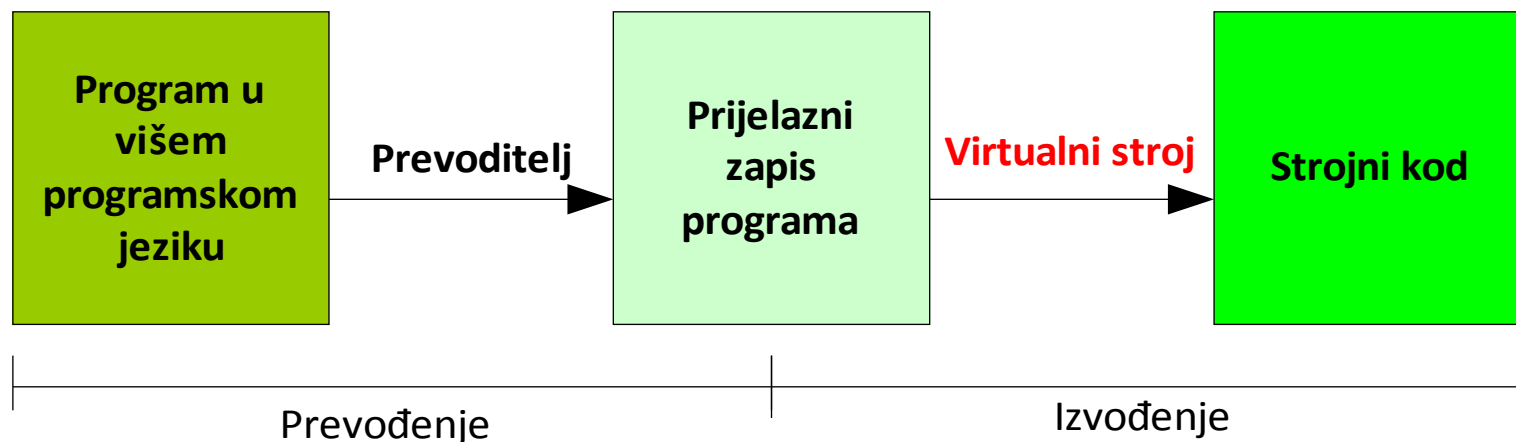


```
0:    bipush    42
2:    istore_1
3:    bipush    24
5:    istore_2
6:    iload_1
7:    iload_2
8:    iadd
9:    istore_3
10:   return
```

```
IL_0001:  ldc.i4.s  42
IL_0003:  stloc.0
IL_0004:  ldc.i4.s  24
IL_0006:  stloc.1
IL_0007:  ldloc.0
IL_0008:  ldloc.1
IL_0009:  add
IL_000a:  stloc.2
IL_000b:  ret
```

Virtualni stroj

- Prijelazni zapis – kako do strojnog koda?
 - interpretacijom
 - dinamičkim prevođenjem (Just In Time -JIT)
- programska podrška koja izvodi prijelazni zapis stvara privid izvođenja na stvarnom sklopovlju - program se izvršava na virtualnom stroju



Virtualni stroj

- Virtualni stroj upravlja:
 - memorijom (objektima na hrpi - heapu)
 - sinkronizacijom niti (thread)
 - iznimkama
 - sažimanjem praznina (garbage collection)
 - sigurnošću

CLR (Common Language Runtime)



- Microsoftova referentna implementacija virtualnog stroja .NET platforme
- uvijek prevodi prijelazni zapis (CIL) u strojni kod
- implementacija CLI specifikacije (normirane po ECMA-335 i ISO/IEC 23271)

CLR (Common Language Runtime)



- CLI specifikacija određuje pravila koja programski jezik mora zadovoljiti da bi ga CLI sukladna implementacija mogla prevesti i izvesti, i to:
 - skup tipova i operacija (CTS-Common Type System)
 - meta podatke o strukturi programa, neovisne o jeziku
 - pravila za programski jezik (CLS – Common Language Specification)
 - sustav izvođenja (VES-Virtual Execution System)

HotSpot VM

- Sunova referentna implementacija Javinog virtualnog stroja
- interpretira bytecode i po potrebi ga prevodi u strojni kod
- Sunov virtualni stroj pronalazi dijelove programa koji se često izvode – **hot spot**-ove, te ih prevede u strojni kod nakon određenog broja izvođenja
- klijentski (Client VM) – brzo pokretanje i minimalno zauzeće memorije
- poslužiteljski (ServerVM) – čim brže izvođenje

HotSpot VM

```

public static void main(String[] args) {
    // Create an ITimer using the Factory class:
    final ITimer timer = TimerFactory.newTimer();
    for (int repeat = 0; repeat < 25; ++repeat)
    {
        timer.start();
        sum(100);
        timer.stop();
        System.out.println(repeat+":\t"+ timer.getDuration());
        timer.reset();
    }
}

public static int sum(int n) {
    if (n <= 1)
        return 1;
    else
        return n + sum(n - 1);
}

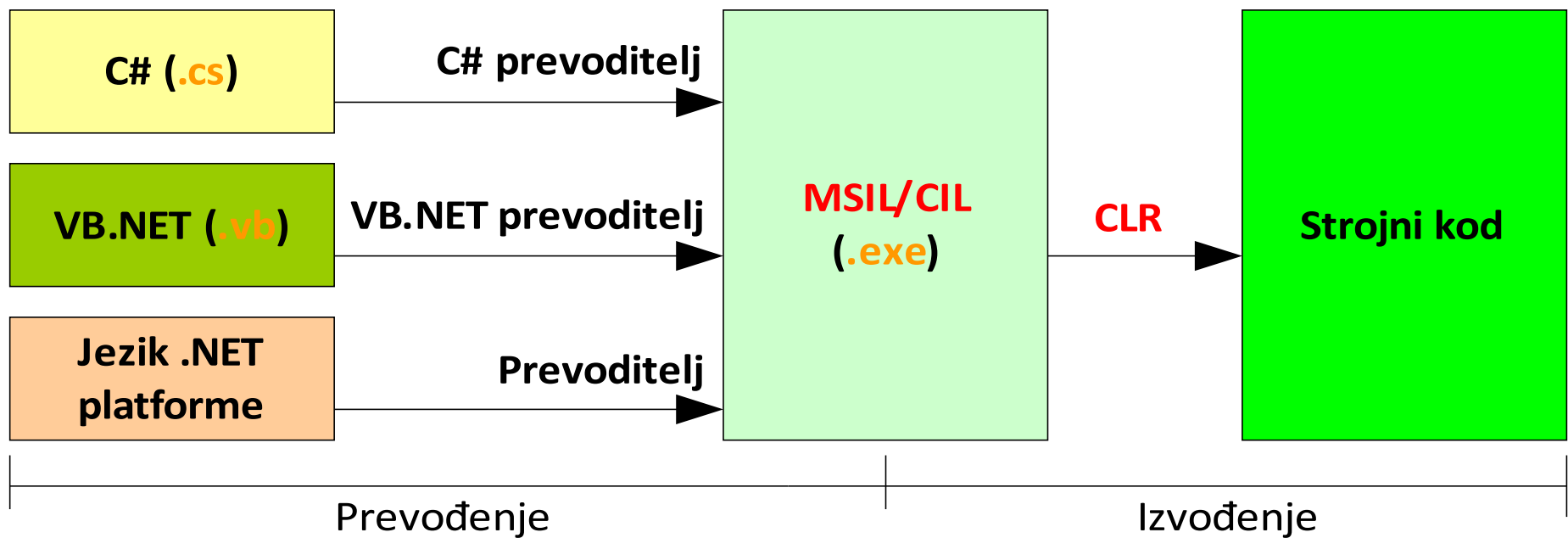
```

→ → **mjerjenja trajanja izvođenja funkcije**

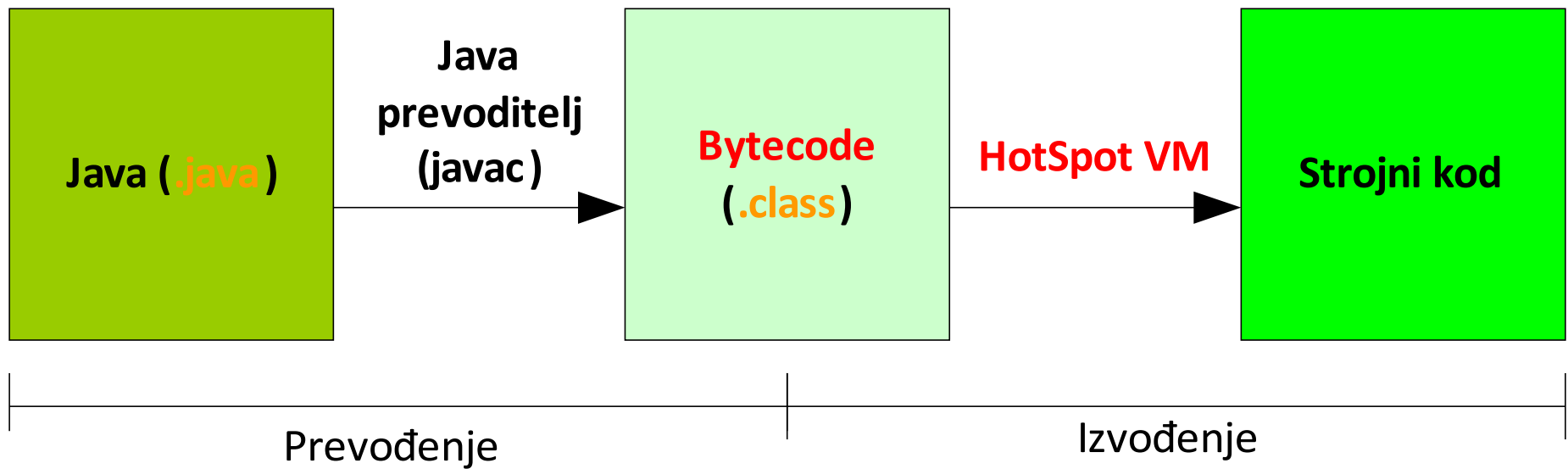
→ → **rekurzivna funkcija**

hotspot.jar

CLR (Common Language Runtime)



HotSpot VM



Biblioteke osnovnih razreda

- većina programskih jezika ima osnovnu biblioteku (*core library*), koja obično uključuje:
 - osnovne strukture podataka
 - mehanizme ulaza i izlaza (zaslon, datoteka)
 - osnovne algoritme
- moderni programski jezici žele postići prenosivost
 - ukloniti ovisnost o operacijskom sustavu
 - ukloniti ovisnost o postojećim bibliotekama funkcija

Biblioteke osnovnih razreda

- dobro definiran i zaokružen skup osnovnih paketa, biblioteka razreda, sučelja i tipova
- temelj svake komponente, kontrole i aplikacije programske platforme

Biblioteke osnovnih razreda

- Java - Java Foundation Classes (JFC)
 - .NET - Base Class Library (BCL)
-
- | | |
|----------------------|------------------------|
| • System | • java.lang |
| • System.IO | • java.io |
| • System.Net | • java.net, javax.net |
| • System.Threading | • java.util.concurrent |
| • System.Collections | • java.util.collection |
| • System.Reflection | • java.lang.reflect |
| • System.Windows | • javax.swing |

Javini appleti

- "Živi" sadržaj na Webu – od samog početka Weba
- Javina aplikacija
 - samostalni program
- Javin applet
 - namijenjen izvođenju unutar preglednika koji podržava Javu
- Applet:

AppletDemo.html

 - dolazi preko mreže
 - ima restrikcije pristupa resursima računala, mrežne komunikacije
 - nasljeđuje razred *java.applet.Applet*
 - životni vijek određen stranicom koja ga prikazuje

Javini appleti - Primjer(1/4)



```
import java.applet.Applet;
import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics;

public class AppletDemo extends Applet
{
    int brojKvadratica, delay;
    public void init() {
        setForeground (Color.green);
        brojKvadratica=Integer.parseInt
            (getParameter("BrojKvadratica"));
        delay=Integer.parseInt(getParameter("Delay"));
        System.out.println("numK="+brojKvadratica+"
            delay="+delay);
    }
    public void start() {}
    public void stop() {}
    public void destroy() {}
    ...
}
```

Javini appletti - Primjer(2/4)



```
...  
public void paint(Graphics g)  
{  
    int a=10; int b=20; int h=30; int v=40;  
    for (int i=1;i<brojKvadratica;i++) {  
        g.drawRect(a+i,b+i,h+i,v+i);  
        try{  
            Thread.sleep(delay);  
        } catch (Exception e) {}  
    }  
}  
}
```


Javini appleti - Primjer(3/4)



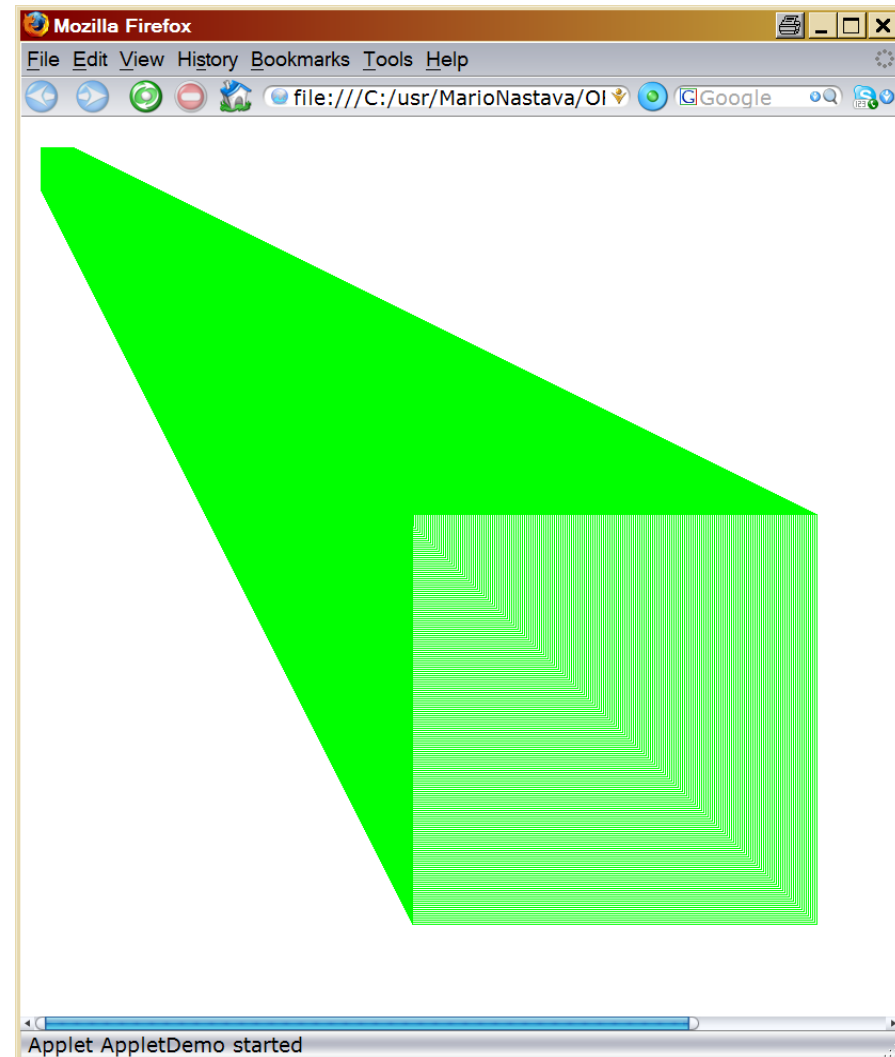
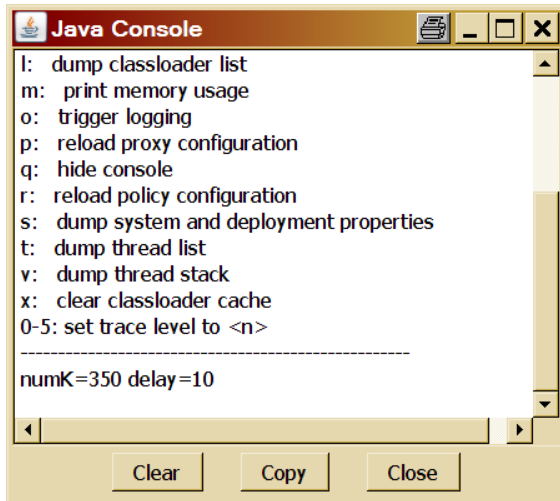
AppletDemo.html

```
<HTML>
  <BODY>
    <applet code="AppletDemo.class"
              width=1000 height=800>
      <PARAM NAME=BrojKvadratica VALUE="350">
      <PARAM NAME=Delay VALUE="10">
    </applet>
  </BODY>
</HTML>
```

Ime razreda

Parametri

Javini appleti - Primjer(4/4)





Smanjenje složenosti

Svojstva idealnog programskog jezika

- jednostavnost
 - uniformnost
 - ortogonalnost
 - jasnoća
 - sigurnost
 - modularnost
 - apstrakcija
 - izražajnost
 - fleksibilnost
 - efikasnost
 - sličnost sintakse s postojećim jezicima
-
- Idealan programski jezik dopušta programeru da se usredotoči na konkretan problem, a ne na "začkoljice" programskog jezika

Automatsko upravljanje memorijom



- Dinamičko alociranje memorije
 - C programer mora ručno tražiti i osloboditi memorijski prostor
- Moderni jezici:
 - automatsko upravljanje memorijom:
 - objekti se alociraju na programskoj hrpi (*heap*)
 - novi objekt instancira se ključnom riječi *new*
 - dio virtualnog stroja (*garbage collector*) uništava objekte koji se više ne koriste
 - u jeziku nema primitiva za izravno upravljanje memorijom
 - zauzimanje i oslobađanje memorije u C-u *malloc()* i *free()*

Automatsko upravljanje memorijom



- Prednosti – česte programerske grješke poput:
 - nedopuštenog pristupa memoriji (npr. pristup polju izvan granica polja)
 - korištenje neinicijaliziranog pokazivača (tj. reference)
 - visećeg pokazivača/reference (*dangling pointer*)
 - curenja memorije (*memory leak*)
 - dvostrukog oslobađanje istog memorijskog prostora
- koje se teško uočljive, a mogu dovesti do gubitka podataka ili nepredvidivog ponašanja, ne mogu se dogoditi
- Loše napisan program ne može srušiti operacijski sustav(ne pristupa više sistemskoj memoriji izravno)

Određivanje tipova

- u C-u:

```
int a=5;
float b=a;
```
- ili (C++):

```
Kruska *kruska;
Jabuka *jabuka = (Jabuka *) kruska;
```
- C i C++ -> jezici s slabom tipizacijom -> ne prijavljuju "grješke" ovog tipa
- jezici sa strogim određivanjem (*strongly typed*) tipa (C#, Java) – grješka prilikom prevođenja

Iznimke

- Cilj – omogućiti kontrolu nad nepredviđenim situacijama:
 - biti svjestan mogućnosti nastanka pogreške
 - pokušati oporavak od pogreške
 - upozoriti korisnika opisnom porukom ako sve ostalo propadne
- Iznimke
 - objekti koji signaliziraju grješku, razmjenjuju se između klasa
 - mehanizmi ugrađeni u sam jezik

Iznimke



```
File postavke = new File("postavke.cfg");
try{
    procitajPostavke(postavke);
}catch (FileNotFoundException e){
    //nema datoteke?
}catch (EOFException e){
    //procitan kraj datoteke
}catch (ObjectStreamException e){
    //datoteka je ostecena?
}catch (IOException e){
    // neka druga I/O grjeska
}
```

Constructor Detail

FileInputStream

```
public FileInputStream(String name)
    throws FileNotFoundException
```

```
public void procitajPostavke(File datoteka)
    throws IllegalArgumentException,
        FileNotFoundException, IOException
{
    if (datoteka == null) {
        throw new IllegalArgumentException
            ("Nema datoteke s postavkama");
    } ...
    InputStream in = new FileInputStream(filename);
    ...
}
```

Niti - Dretve - *Threads*

- Nekad – POSIX niti i semafori:

```
int pthread_create(pthread_t *thread, const pthread_attr_t
    *attr, void *(*start_routine)(void *), void *arg);
int pthread_join(pthread_t cekana_dr, void **stanje);
```

```
int semget(key_t key, int nsems, int flags);
int semctl(int semid, int semnum, int cmd, union semun
    arg);
```

- Sad – niti i osnovni sinkronizacijski mehanizmi dio programskog jezika:

```
public class HelloThread extends Thread {
    public void run() { System.out.println("Bok!"); }
    public static void main(String args[]) {
        (new HelloThread()).start();
    }
}
```



Normiranje i licenciranje

Otvorenost i prog. inženjerstvo

- Normiranje – ANSI C ili C89, normiran po ANSI X3.159-1989 normi – svaki prevoditelj sukladan normi korektno će prevesti programski kod napisan po normi, na bilo kojoj platformi
- Biblioteka funkcija koja dolazi s izvornim kodom bit će lakša za primjenu od biblioteke koja se distribuira samo u binarnom obliku, zato jer:
 - programer shvaća što se događa "ispod"
 - dobro komentiran izvorni kod je najbolja dokumentacija
 - manjkavosti biblioteke se mogu zaobići ili ispraviti (ovisno o licenci)

Java i norme



- Java platforma – *de facto* norma
- Pokušaj formalne normizacije Java platforme 1997. (ISO) i 1999. (ECMA)
 - Sun povukao zahtjev oba puta
- Cilj normiranja – sačuvati kompatibilnost raznih implementacija (nakon prihvata norme, o tom se brine normizacijsko tijelo - ISO/ECMA)
- Sun u oba slučaja odlučio sam kontrolirati daljnji razvoj platforme

Java i norme



- Sun kontrolira Java platformu:
 - zajedničkim donošenjem novih specifikacija kroz Java Community Process (JCP) – zainteresirane tvrtke, pojedinci i djelatnici Sun-a sudjeluju u definiranju i implementaciji novih dijelova Java platforme
 - testiranjem (da bi neki proizvod bio sukladan, mora proći testiranje, svaki dio platforme definira svoj Technology Compatibility Kit - TCK).
 - licenciranjem (Sun i dalje zadržava vlasništvo nad binarnim kodom platforme)

Uvijek otvorena, sada i slobodna



- Izvorni kod temeljnih Java biblioteka (BCL – Base Class Library) oduvijek dio Java SDK-a (*rt.jar*), ali pod *gledaj-i-ne-mijenjaj* licencom
- krajem 2006., izvorni kod Java platforme objavljen pod GPLv2 licencom, i to
 - J2ME
 - J2SE
 - J2EE (dio)
 - javac
 - HotSpot

Platforma .NET i norme

- otvorene norme ECMA-335 i ISO/IEC 23271 definiraju CLI (Common Language Infrastructure):
 - skup tipova i operacija (CTS-Common Type System)
 - meta podatke o strukturi programa, neovisne o jeziku
 - pravila za programski jezik (CLS – Common Language Specification)
 - sustav izvođenja (VES-Virtual Execution System) - prevodi CIL u strojni kod
- norme omogućile izradu novih implementacija:
 - Mono (GPL, Linux, Windowsi, Solaris, Mac OS X)
 - Portable.NET /DotGNU

Otvorenost i .NET

- *shared source*
 - Microsoftova inicijativa nudi inačicu implementacije CLI specifikacije (Shared Source CLI) pod licencom koja omogućuje uvid u izvorni kod
 - polazna točka za ostale implementacije.

A large, stylized green letter 'R' is positioned on the left side of the slide, partially cut off by the edge.

Pitanja?