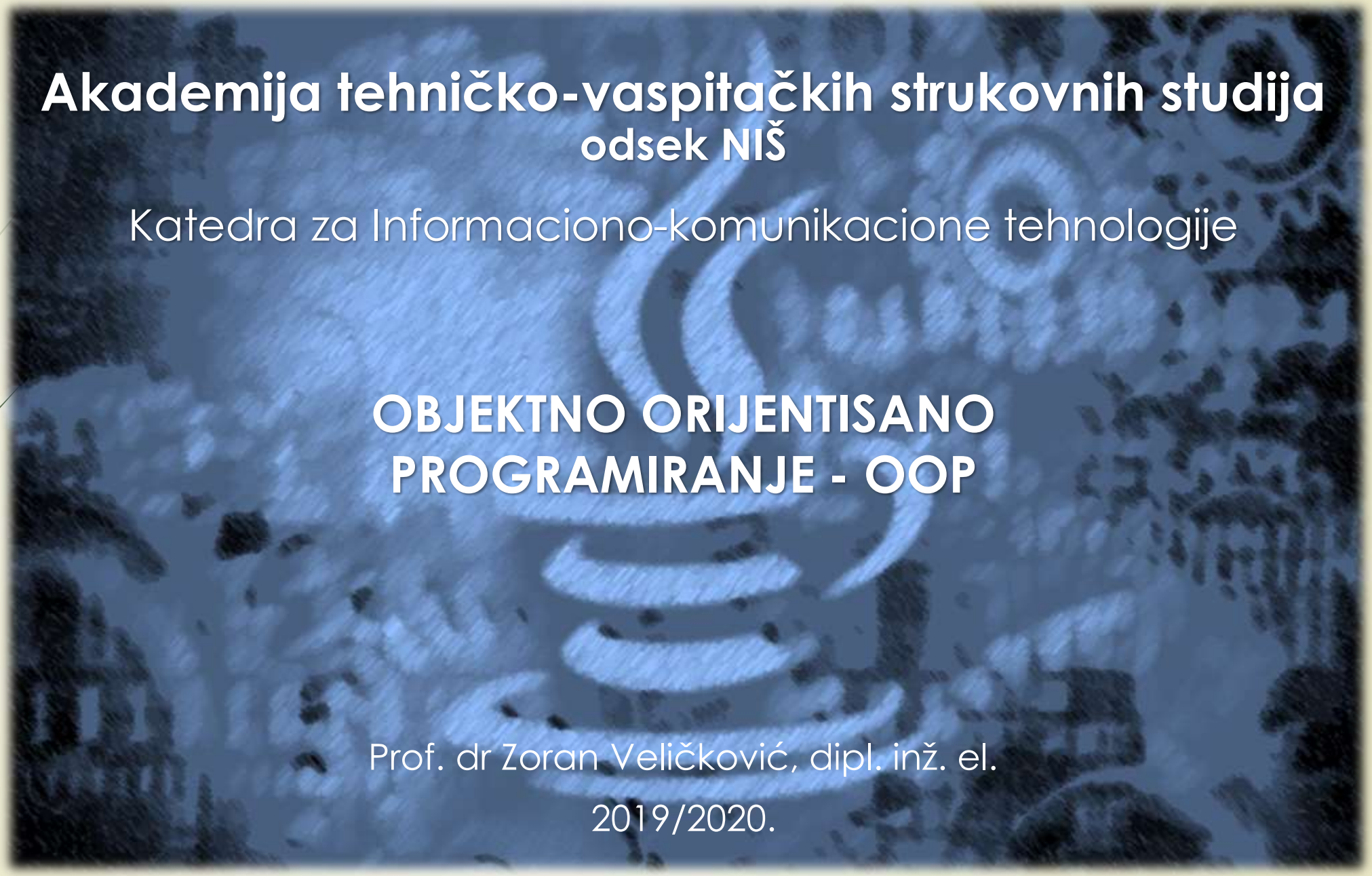


Akademija tehničko-vaspitačkih strukovnih studija
odsek NIŠ

Katedra za Informaciono-komunikacione tehnologije

OBJEKTNO ORIJENTISANO PROGRAMIRANJE - OOP

Prof. dr Zoran Veličković, dipl. inž. el.

2019/2020.

Prof. dr Zoran Veličković, dipl. inž. el.

OBJEKTNO ORIJENTISANO PROGRAMIRANJE - OOP

Osnove OO jezika - JAVA

(3)

Sadržaj

- **NASTANAK JAVE**
  - Ciljevi OO programskog jezika Java
  - Nastanak i razvoj Jave
  - IDE - Eclipse
- **OSNOVNE KARAKTERISTIKE JAVE**
  - Java i OOP
  - Kapsuliranje u Javi
  - Java i nezavisanost od platforme
  - Javina virtuelna mašina JVM
- **JAVA I BYTECODE**
  - Softverske platforme Jave: JEE, JSE, JME
- **JDK, JRE, JVM i JIT**
  - Java JDK: setovanje J2SE 5
- **RAZVOJ JAVA APLIKACIJE**
  - Konzolni kompajler: javac
  - Java: .java i .class datoteke
  - Java: izvršenje programa
- **JAVA-PRVI PROGRAM**
  - Hello world
  - Upravljanje programskim izvršenjem

Ciljevi programskog jezika Java

- Dizajneri Jave su sebi ostvarili nekoliko **VAŽNIH CILJEVA** (programeskih snova) koji su inspirisani **NOVIM DEŠAVANJIMA** na polju **RAČUNARSKE TEHNIKE**.
- U tom smislu se za programski jezik **JAVA** može istaći:
  - Nastao je kao težnja programera za **EFIKASNIM PROGRAMSKIM JEZIKOM** koji će prirodno raditi u **MREŽNOM OKRUŽENJU**;
  - Programski jezik koji će podržati **KLIJENT-SERVER** tehnologiju i razrešiti probleme **DISTRIBUIRANIH APLIKACIJA**, odnosno, korišćenje **UDALJENIH METODA I PODATAKA**;
  - Treba da ima podršku za **NOVE PRIMENE** i poseduje osobinu **PRENOSIVOSTI**, odnosno, sposobnost ADAPTACIJE na **NOVE PLATFORME** i **OPERATIVNE SISTEME**;
  - Java treba da ima ugrađena **JEZIČKA POBOLJŠANJA** i sve sve **NOVE NAPREDNE TEHNIKE PROGRAMIRANJA**;
  - Treba da ima ugrađenu podršku za **BEZBEDNO IZVRŠAVANJE KODA**.

Programski jezik Java

- **JAMES GOSLING**, **MIKE SHERIDAN** i **PATRICK NAUGHTON** su radila na projektu **JAVA technology** (1991. god.) koji je razvijen u kompaniji *Sun Microsystems Inc*.
  - **PATRICK NAUGHTON** je bio zadužen za se poslovni aspekt projekta.
  - **MIKE SHERIDAN** se bavio grafičkim okruženjem Aspen.
  - **JAMES GOSLING** se bavio programskim jezikom **JAVA** i virtuelnim okruženjem za njegovo izvršenje **JVM** - Javina Virtualna Mašina.
- Prva verzija JAVE je objavljena **1995**.
- Kada je kompanija **ORACLE** 2010. godine preuzela kompaniju *Sun Microsystems Inc.*, Gosling je napušta.

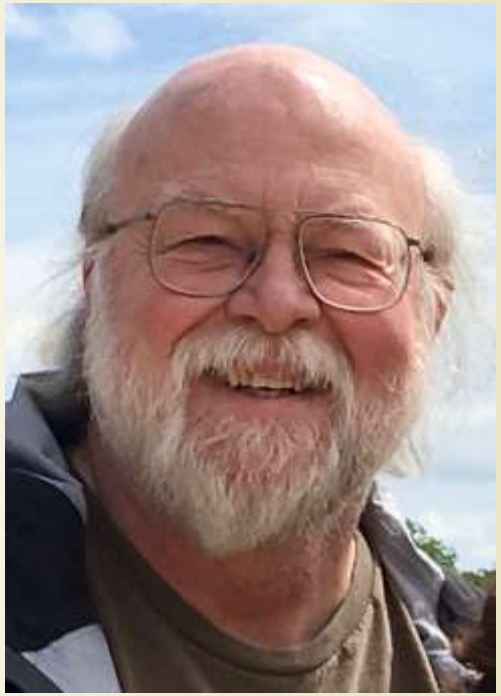
James Gosling

Patrick Naughton

Mike Sheridan

Nastanak i razvoj Jave

- **JAMES GOSLING** je trenutno je angažovan u **Guglu**.
- Kompanija **ORACLE** je nastavila razvoj na ovom programskom jeziku.
- **RAZVOJNI TIM** Jave izdaje nove verzije na svakih 6 meseci (u martu 2019. se očekuje nova verzija Jave koju će podržavati **OpenJDK** udruženje).
- Java se smatra **NAJBRŽE RASTUĆOM** programskom tehnologijom **SVIH VREMENA**.
- **JDK** (engl. *Java Development Kit*) je **SKUP RAZVOJNIH SOFTVERA** koji rade sa komandne linije, a **BESPLATNO** je raspoloživ zahvaljujući kompaniji **ORACLE**.
- Deo **LABORATORIJSKIH VEŽBI** biće realizovan upravo **JDK**-om.
- Preuzimanje **JDK**-a je moguć sa sajta: http://www.oracle.com.



Osnovne karakteristike Jave

- Javu karakteriše **JEDNOSTAVANOST** i sličnost sa C-om i C++-om, a naročito sa C#.
- Za Javu se može reći da je:
  - **OBJEKTNO ORIJENTISANA** (OO): Java je zasnovana na klasama.
  - **PRENOSIVA**: Java radi na svim OS.
  - **BEZBEDANA**: Java je izuzetno bezbedan programski jezik jer se izvršava u okruženju Javine viruelne mašine (**JVM** okruženju).
- Java podržava:
  - **VIŠENITNO PROGRAMIRANJE**: Preuzima deo posla od operativnog sistema!
  - **NEZAVISANOST OD PLATFORME**: NAPIŠI JEDNOM - **IZVRŠAVAJ BILO GDE i BILO KADA**, još jedan san programera!
  - **DISTRIBUIRANE APLIKACIJE**: Lako upravlja TCP/IP protkolima, i pristupa udaljenim metodama.
  - **PODRŽAVA DINAMIČNE APLIKACIJE**: Razrešavanje pristupa objektima u trenutku izvršavanja.

Java i OOP (1)

- Kod **PROGRAMIRANJA** se razlikuju **DVA KONCEPCIJSKA MODELA** za realizaciju aplikacije:
  - Realizacija oko **NAREDBI** – (engl. *procedure-oriented*)
  - Realizacija oko **PODATAKA** (engl. *object-oriented*).
- Setite se, kod **PROCEDURALNE PARADIGME** aplikacija se gradi **KORAK PO KORAK** formiranjem niza programskih instrukcija.
- Kod **PROCEDURALNE PARADIGME** programer se fokusira na **FUNKCIJE** kojima menja podatke.
- **OO PROGRAMSKA PARADIGMA** ima mnogo više **SLIČNOSTI** sa **REALNIM ŽIVOTOM**, odnosno, sa načinom na koji funkcioniše čovek.
- **SVAKI OO** program se sastoji od **MNOGO ENTITETA** koji se u žargonu OO jezika nazivaju **OBJEKTI**.
- Kod napisan u **OO JEZICIMA** se može **VIŠEKRATNO KORISTI** čime se programski kod **REDUKUJE**.
- **SVAKI JAVA PROGRAM JE OO**!

Java i OOP (2)

- Kod **OO PARADIGME** složene PROGRAMSKE STRUKTURE i PODACI su "**ZAPAKOVANI U KLASE**" kojima se pristupa na strogo definisan način.
- Java zadovoljava sve **OSNOVNE PARADIGME OO** programiranja, a to su:
  - Apstrakcija,
  - Kapsuliranje,
  - Nasleđivanje i
  - Polimorfizam.

Kapsuliranje u Javi

- Osnovna **JEDINICA KAPSULIRANJA** u JAVI je **KLASA** (engl. *class*).
- Zapamtite: **KLASE** definišu **NOVE TIPOVE PODATAKA**!
- **ČLANOVI KLASE** (engl. *class member*) mogu biti:
  - **PROGRAMSKI KOD** i
  - **PODACI**.
- **KLASOM** se definiše **STRUKTURA** i **ZAJEDNIČKO PONAŠANJE OBJEKTA** (nastalog – iniciranog od klase).
- Klasama se **KAPSULIRA SLOŽENOST** programske strukture i podataka.
- **OBJEKTI** su instancirani primerci klase – još se kaže da su objekti **INSTANCE KLASE**.
- KLASE uobičajeno imaju **JAVNI i PRIVATNI** deo.
  - Javni deo klase je dostupan **SPOLJNIM** korisnicima (engl. *public*).
  - Privatni deo klase je dostupan **SAMO ČLANOVIMA KLASE** (engl. *private*).

Nezavisanost od platforme - JVM

- Tokom postojanje, unutrašnja konstrukcija klase se može **MENJATI**, a da se pritom ne remeti programski kod koji se oslanja na **JAVNI INTERFEJS** klasa.
- Java je **KOMPAJLERSKI JEZIK** – dakle posedje kompajler, ali za razliku od ostalih jezika, **NE PREVODI** izvorni kod u izvšni - **MAŠINSKI KOD**!
- Razlog ovome leži u ideji da programski jezik Java treba bude **NEZAVISAN OD PLATFORME** (bilo hardverske bilo softverske).
- To je razlog zašto Java poseduje **DODATNI PREVODILAC** (kompajler) koji se aktivira tek U **TRENUTKI IZVRŠAVANJA** - **JIT** (engl. *Just In Time*) kompajler.
- **JIT** kompajlerom i drugim **RESURSIMA** upravlja **SPECIJALIZOVANO IZVRŠNO OKRUŽENJE JAVE** Javina Virtualna Mašinia - **JVM** (engl. *Java Virtual Machine*).
- Koncept OO programskih jezika kojim se prevazilazi problem **HETEROGENOSTI** terminalne opreme je prikazan na sledećem slajdu.

Javina virtuelna mašina JVM

Izvorni kod

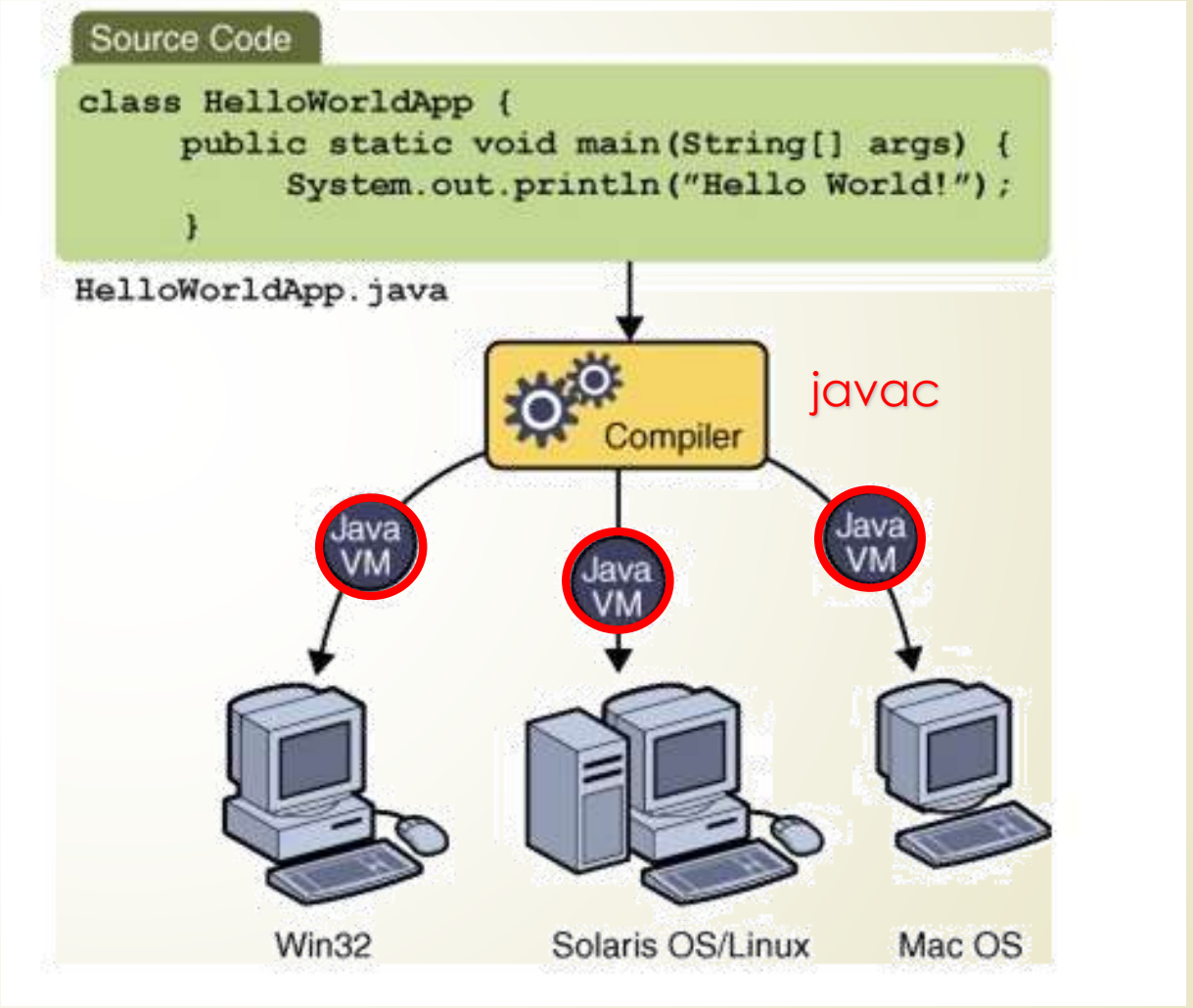
*.java

Bajtkod

*.class

Izvršni kod

*.exe



Java i Bytecode

- Java programi se **PRE IZVRŠAVANJA** prevode u **MEĐUKOD** koji se naziva **BAJT-KOD** (engl. *bytecode*).
- **BAJT-KOD** je **ISTI** bez obzira na kompjutersku platformu na kojoj se izvršava (i hardversku i softversku).
- **BAJT-KOD** se **PREVODI** u **IZVRŠNI KOD PROCESORA** **JIT KOMPAJLEROM**, tek tada se može **IZVRŠITI** u okruženju programa domaćina: **JVM** (Javine Virtualne Mašine).
- **BAJT-KOD** se može izvršiti **NA SVAKOM RAČUNARU** koji poseduje **JVM**.
- Dakle na svakoj računarskoj platformi **TREBA IMATI** instaliranu JVM za:
  - **Određni** procesor i
  - **Određeni** operativni sistem.
- **JVM** je posebno napisan za **SVAKU** računarsku platformu!

Operating Systems

Windows

Platform	CPU Architecture
Windows Server 2019	x64 (64-bit)
Windows Server 2016	x64 (64-bit)
Windows Server 2012 R2	x64 (64-bit)
Windows Server 2012	x64 (64-bit)
Windows 10	x64 (64-bit)
Windows 8.x	x64 (64-bit)
Windows 7	x64 (64-bit)

Linux

Platform	CPU Architecture
Oracle Linux	x64 (64-bit)
Oracle Linux	x64 (64-bit)
Oracle Linux	x64 (64-bit)
Red Hat Enterprise Linux	x64 (64-bit)
Red Hat Enterprise Linux	x64 (64-bit)
Red Hat Enterprise Linux	x64 (64-bit)
Suse Linux Enterprise Server	x64 (64-bit)
Suse Linux Enterprise Server	x64 (64-bit)
Ubuntu Linux	x64 (64-bit)
Ubuntu Linux	x64 (64-bit)

- Oracle Linux covers both kernels: Red Hat Compatible and Unbreakable.

macOS

Platform	CPU Architecture
macOS	x64

RAZVOJNE FAZE

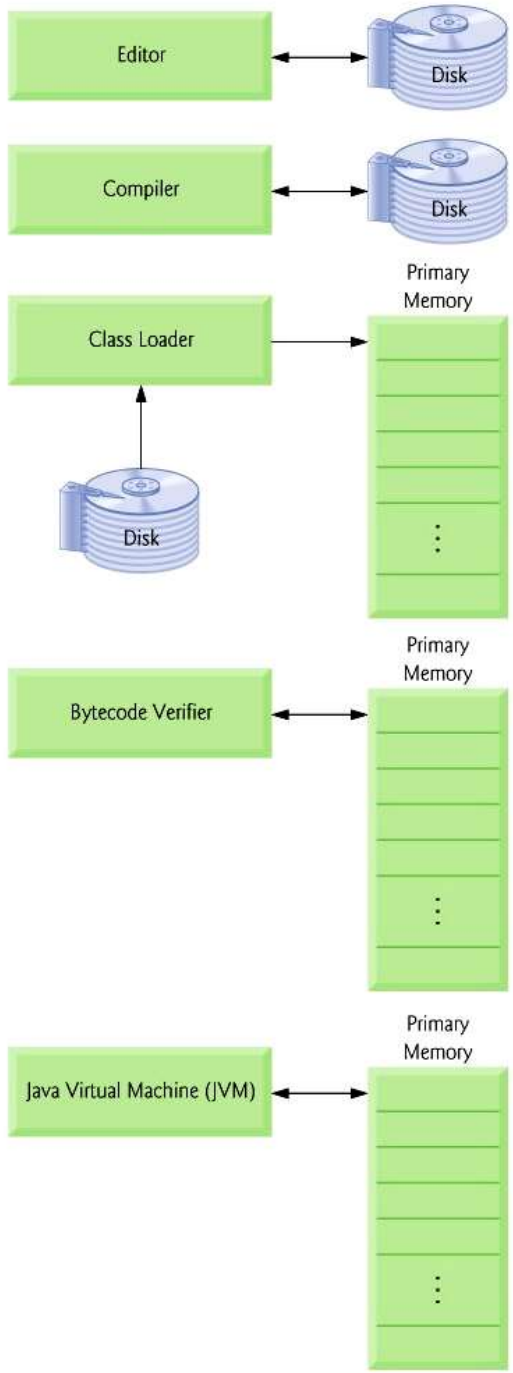
Faza 1: **Editovanje**

Faza 2: **Kompajliranje**
javac

Faza 3: **Punjenje**

Faza 4: **Verifikacija**

Faza 5: **Izvršenje**
java



Izvorni kod se kreira i edituje u editoru teksta i smešta na disk sa ektezijom *.java.

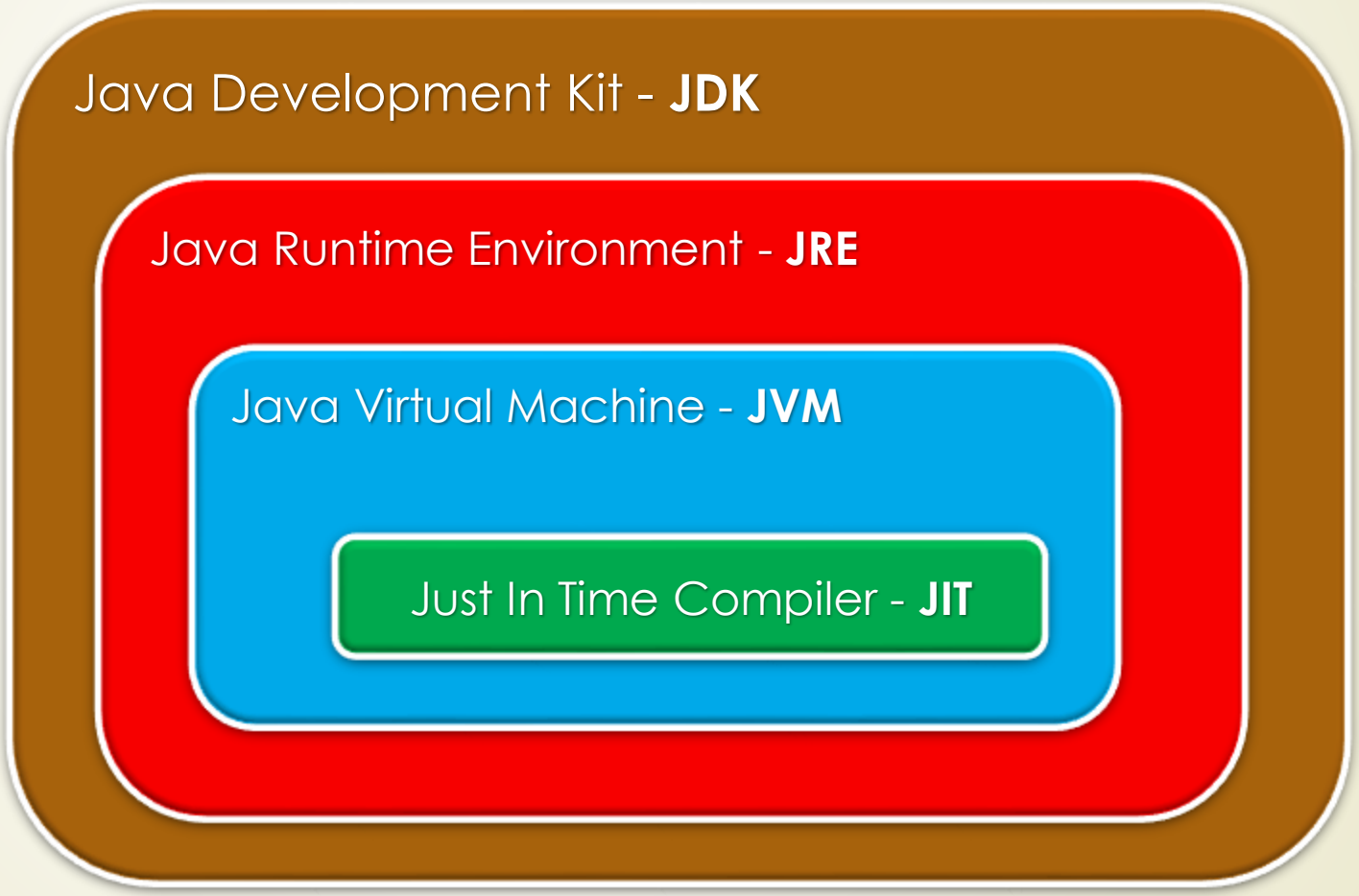
Kompajler formira bytecode i smešta ga na disk u formi .class.

ClassLoader čita .class fajl koji sadrži bajt-kod sa diska i smešta ga u memoriju.

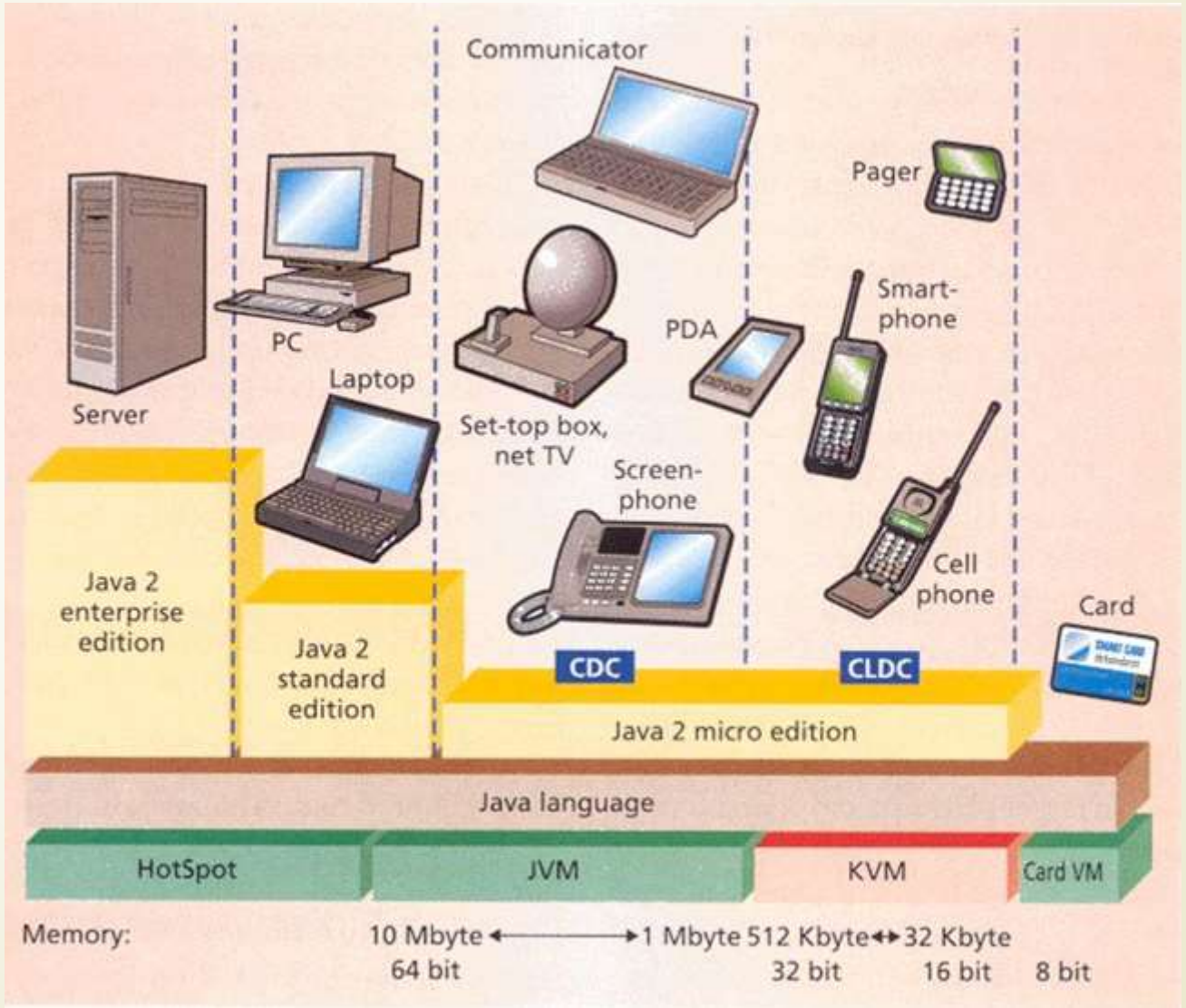
Potvrđuje se da je bytecode važeći i ne poseduje bezbedonosna ograničenja.

JVM učitani bytecode prevodi u jezik koji procesor razume i izvršava

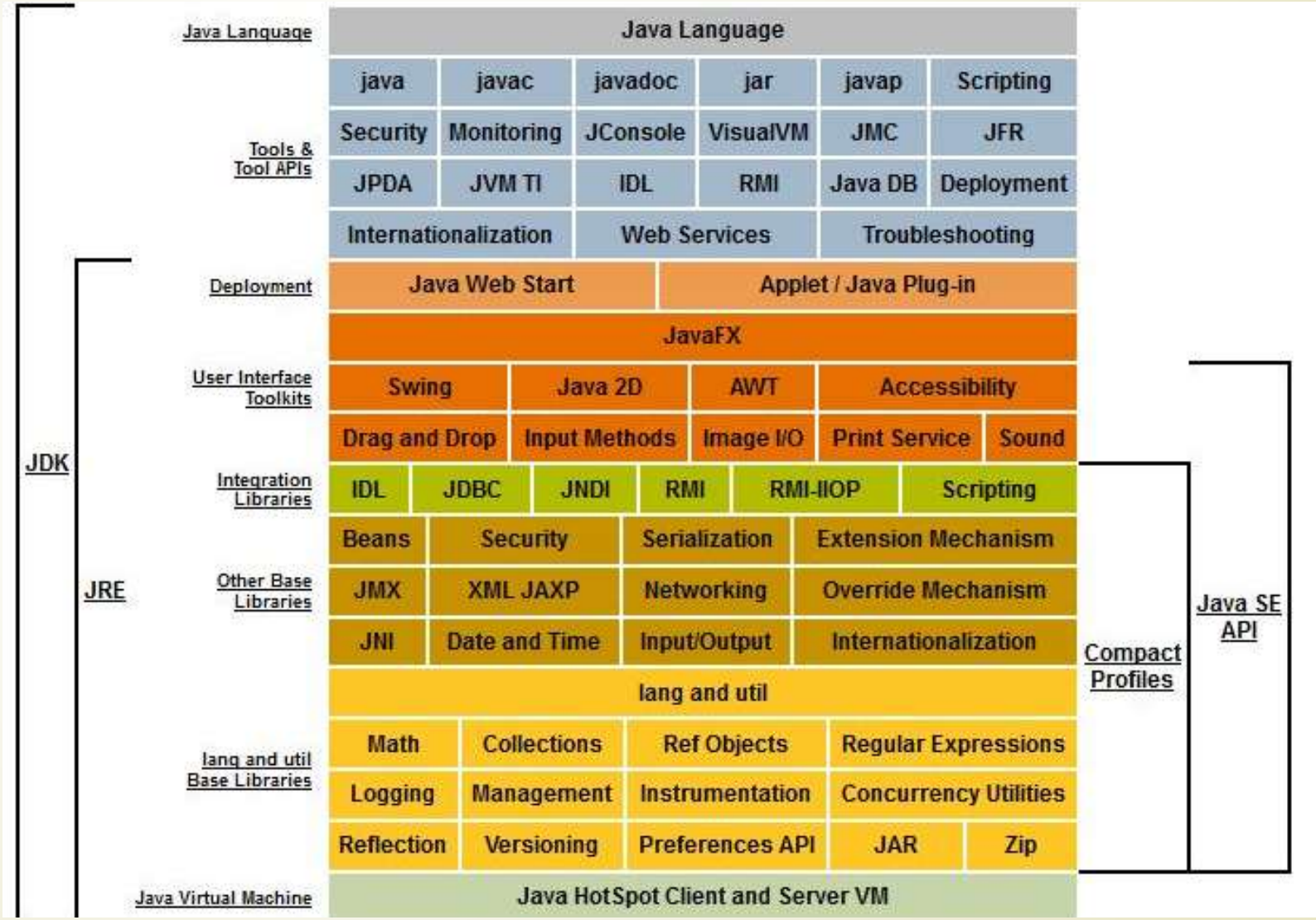
JDK = JRE +JVM + JIT



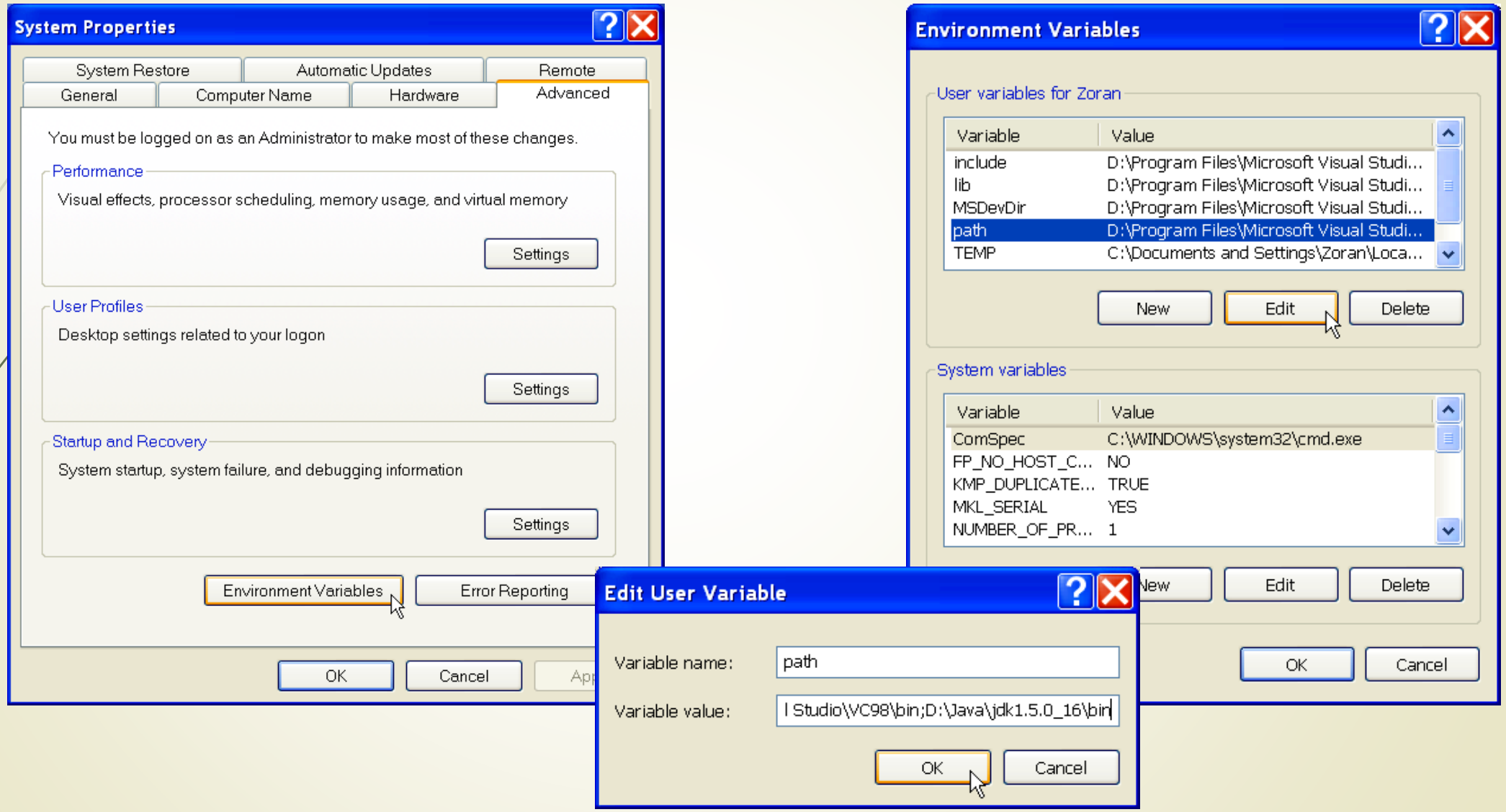
Softverske platforme Jave: JEE, JSE, JME



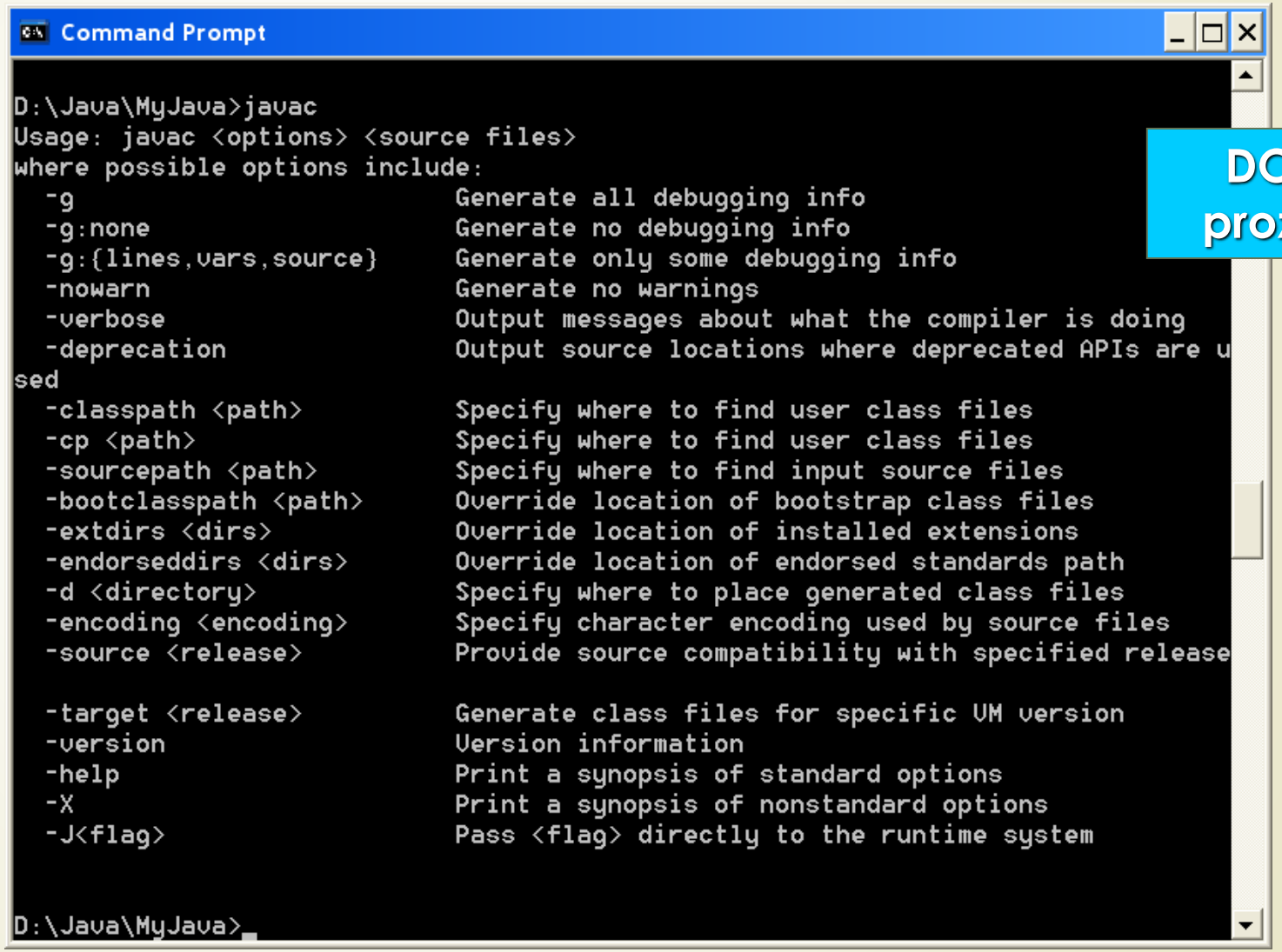
Javin conceptualni dijagram



Java JDK: setovanje J2SE 5

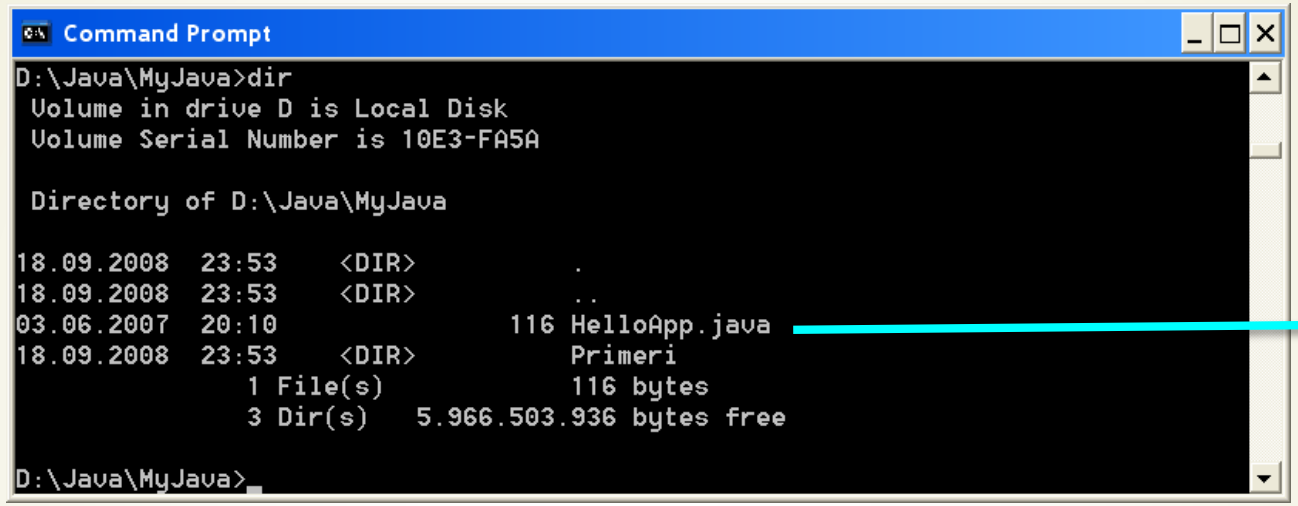


Konzolni kompajler: javac

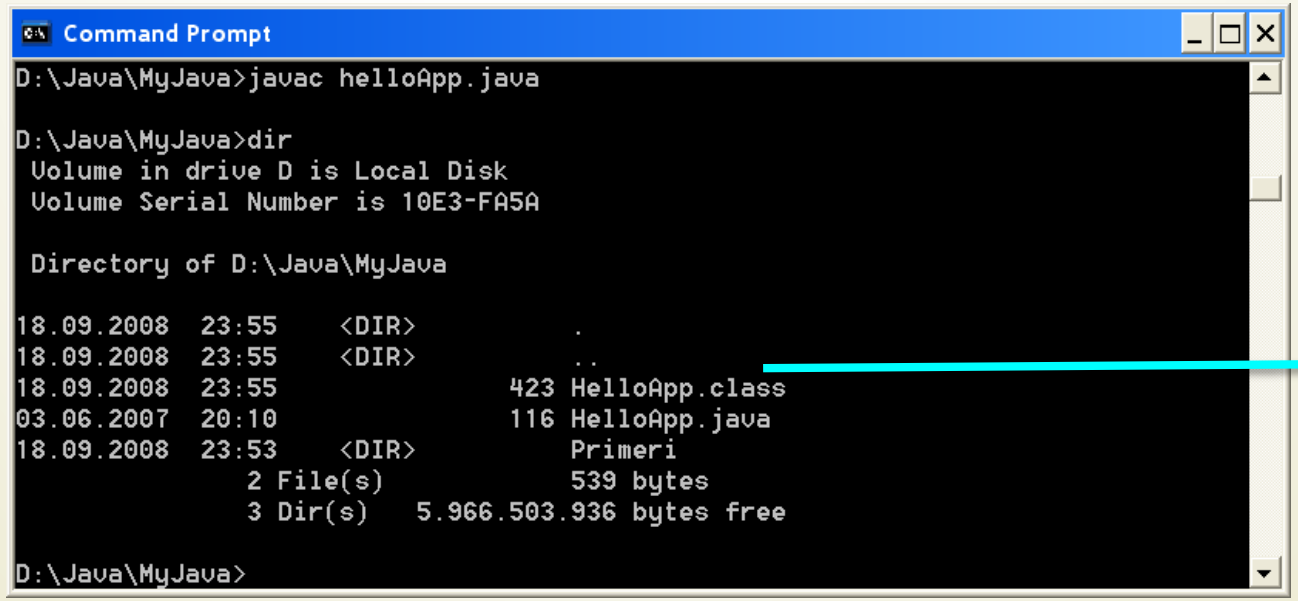


DOS prozor

Konzolni kompajler: javac



Izvorni kod, **.java**

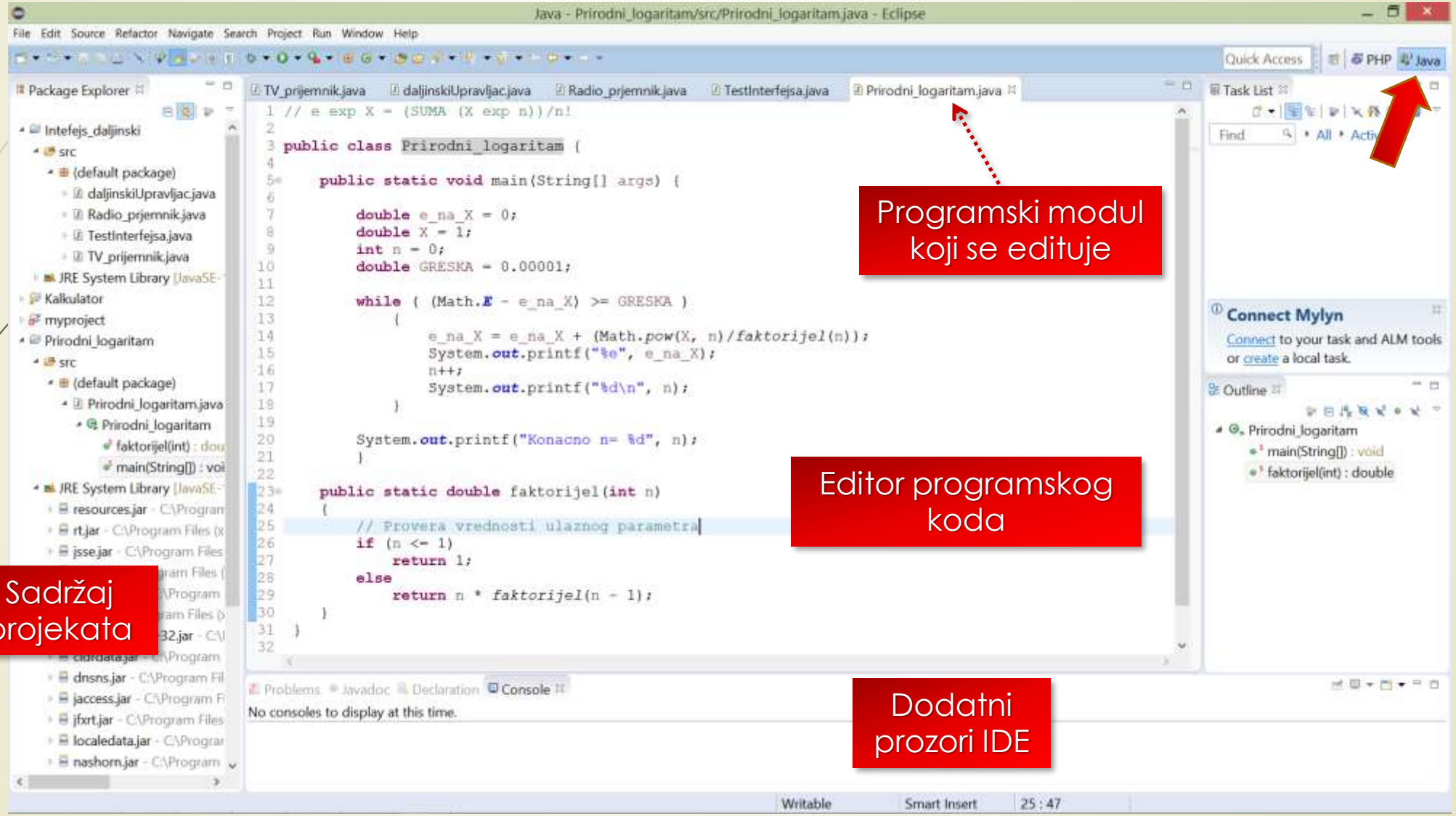


Bajt kod, **.class**

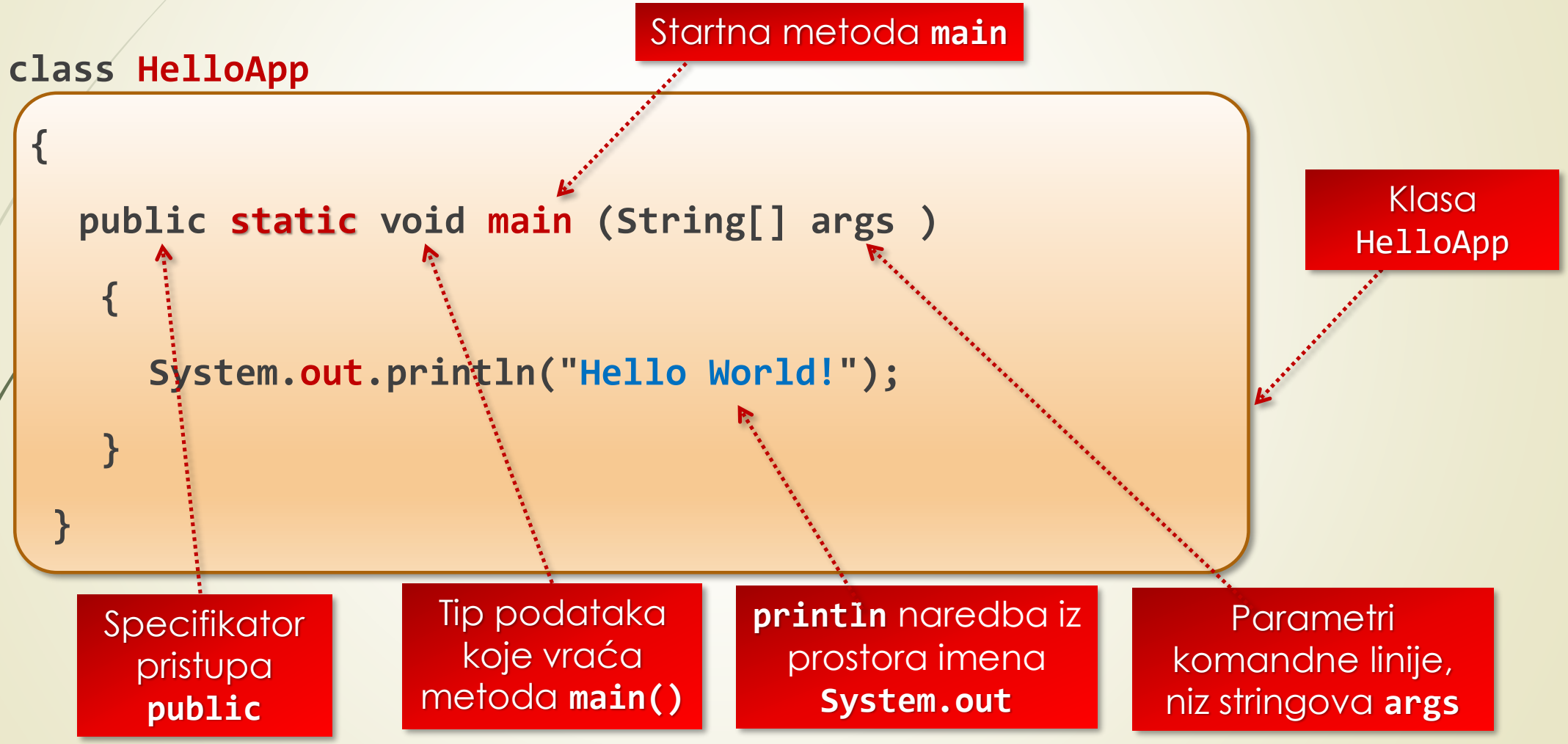
Java i Bytecode

- Izvršavanje sa komandne linije: java HelloApp

IDE Eclipse i Java



Java-prvi program, Hello World!



Paketi - prostor imena u Javi

- U Javi se definišu **PROSTORI IMENA** u kojima se smeštaju metode i podaci koji se koriste u odgovarajućoj klasi.
- Ovo znači da metode mogu imati ista imena ako pripadaju **RAZLIČITIM** imenskim prostorima.
- Da bi se definisalo koja se metoda tačno poziva mora se navesti **HIJERARHIJSKI PRISTUP** preko prostora imena
- U Javi se prostor imena naziva **PAKET**.
- Umesto:

```
println("Hello World!"),
```

piše se:

```
System.out.println("Hello World!")
```

Analiza programa

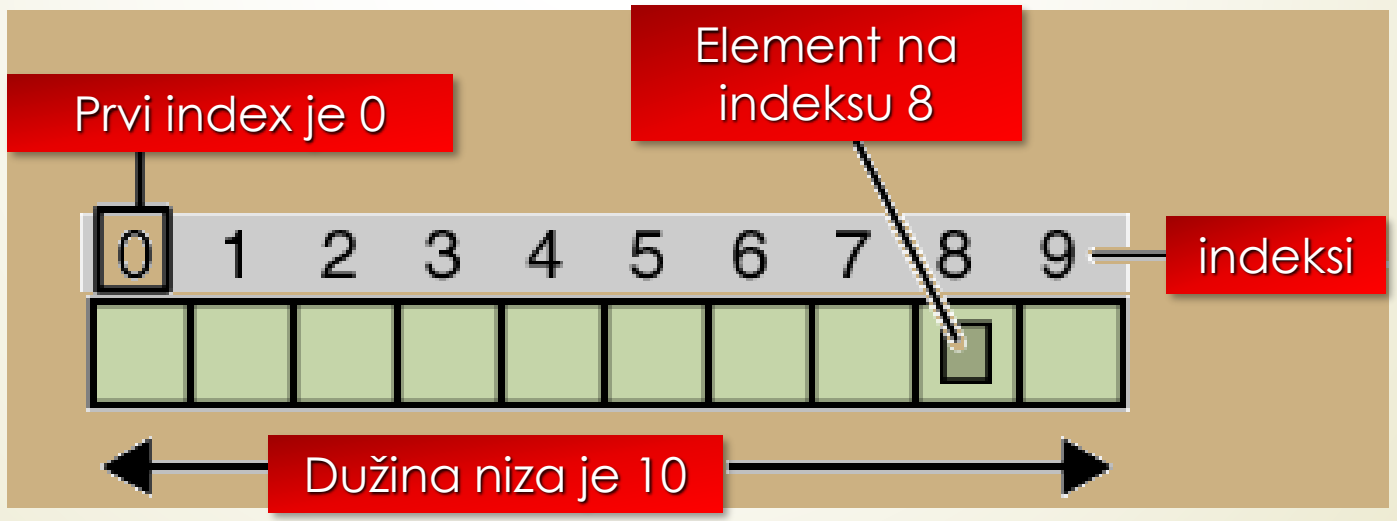
- **Class** Helloapp
- Program počinje pozivom metode **main()**
- Specifikator pristupa: **public**
- Metoda **main()** ne vraća vrednost **void**
- Parametri komandne linije: **String[] args**
- Na ekranu se ispisuje tekst: **Hello world**

```
System.out.println("Hello World!");
```

println naredba iz prostora imena **System.out**

Java: nizovi

- **NIZ** (engl. *array*) predstavlja **GRUPU PROMENLJIVIH ISTOG TIPA** koje se pojavljuju pod **ZAJEDNIČKIM NAZIVOM**.
- Zapamtite: Nizovi su REFERENCNI TIPOVI podataka!
- Koje su posledice (ili benefiti) ove činjenice?



Java: xD nizovi

- Mogući su nizovi **SVIH TIPOVA**!
- Pristup elementima niza moguć je preko njihovog **REDNOG BROJA – INDEX-a**.
- Moguće je definisati sledeće tipove nizova:
  - **JEDNODIMENZIONE** (1D) i
  - **VIŠEDIMENZIONE** nizove (xD):
- Primer dekleracije 1D, 2D i 3D nizova:

```
month_days[] = new int[12];          // 1D niz
int twoD[][] = new int[4][5];        // 2D niz
int threeD[][][] = new int[3][4][5];     // 3D niz
```

Operator **new** je neophodno primeniti prilikom formiranja **REFERENCNIH** tipova!

Java: 2D nizovi

```
class Matrix {
  public static void main (String args[]) {
    double m[][] = {
      { 0*0, 1*0, 2*0, 3*0 },
      { 0*1, 1*1, 2*1, 3*1 },
      { 0*2, 1*2, 2*2, 3*2 },
      { 0*3, 1*3, 2*3, 3*3 }
    };
    int i, j;
    for(i=0; i<4; i++) {
      for(j=0; j<4; j++)
        System.out.print(m[i][j] + " ");
      System.out.println();
   }
  }
}
```

Dekleracija i inicijalizacija 2D niza

Štampanje elemenata 2D niza !