Visoka tehnička škola Niš

Studijski program:

Savremene računarske tehnologije

Internet programiranje

(3)

JAVA osnove

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

Oktobar, 2018.

Ciljevi programskog jezika Java

- Dizajneri **JAVE** su sebi ostavili nekoliko važnih ciljeva (programeskih snova) koji su inspirisani novim dešavanjima na polju računarske tehnike.
- U prvom redu **JAVA** je programski jezik:
  - nastao kao težnja programera za efikasnim programskim jezikom koji će prirodno raditi u **MREŽNOM OKRUŽENJU**;
  - koji treba da podržava **KLIJEN-SERVER TEHNOLOGIJU** i da razreši probleme distribuiranih aplikacija, odnosno korišćenje udaljenih metoda i podataka;
  - koji treba da ima podršku za nove primene i poseduje osobinu **PRENOSIVOSTI**, odnosno **SPOSOBNOST ADAPTACIJE** na nove **PLATFORME I OPERATIVNE SISTEME**;
  - u koji su ugrađena **JEZIČKA POBOLJŠANJA** i sve sve nove **NAPREDNE TEHNIKE** programiranja.
  - Koji 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**, Gosling je napušta napušta (trenutno je angažovan i u Guglu).

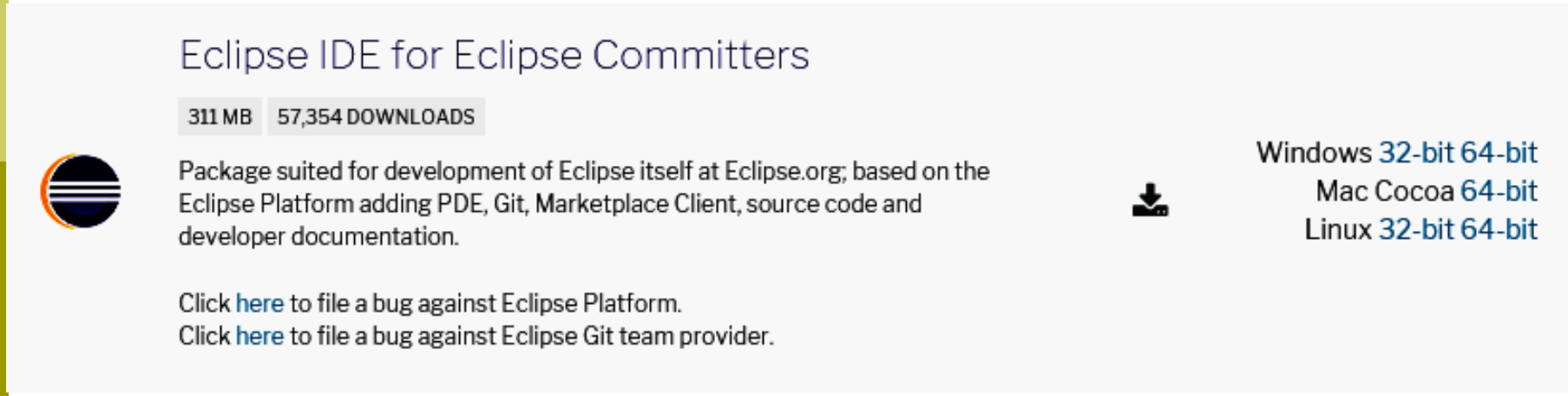
James Gosling

IDE Eclipse i JDK

- 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**.

IDE Eclipse i JDK

- **ECLIPSE** je besplatno, **generičko** integrisano **RAZVOJNO OKRUŽENJE**, koje se može koristiti i za **RAZVOJ** Java aplikacija.
- Da bi **Eclipse** prilagodili za rad u Javi, treba u njega **integristati JDK**, čime se formira **integrisano razvojno okruženje**.
- Ddeo prvog i čitav drugi ciklus lab. vežbi se izvodi u ovom okruženju.



Osnovne karakteristike Jave

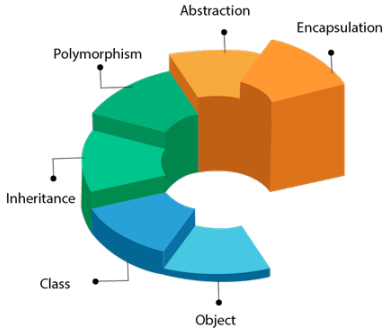
- **Javu** karakteriše **JEDNOSTAVANOST** i **SLIČNOST** sa C/C++ a naročito sa ***C#***.
- Neki od postignutih ciljeva Jave su da je:
  - **BEZBEDAN -** Java je izuzetno bezbedan programski jezik jer se izvršava u okruženju Javine viruelne mašine (**JVM** okruženju).
  - **PRENOSIV -** Java radi na **svim OS**.
  - **OBJEKTNO ORIJENTISAN (OO)** - Java je zasnovana na klasama.
  - **PODRŽAVA VIŠENITNO PROGRAMIRANJE -** Preuzima deo posla od operativnog sistema!
  - **NEZAVISAN OD PLATFORME - NAPIŠI JEDNOM** - **IZVRŠAVAJ BILO GDE** i **BILO KADA**, još jedan san programera!
  - **PODRŽAVA DISTRIBUIRANE APLIKACIJE -** Lako upravlja **TCP/IP** protkolima, i pristupa **udaljenim metodama**.
  - **PODRŽAVA DINAMIČNE APLIKACIJE -** Razrešavanje pristupa objektima u **trenutku izvršavanja** (o tome nešto više kasnije).

Java i OOP (1)

- Realizaciju softverske aplikacije se u programerskom smislu može realizovati na **DVA KONCEPCIJSKA** modela:
  1. Realizacija oko **NAREDBI/FUNKCIJA** POP (engl. Procedure-Oriented Programing)
  2. Realizacija oko **PODATAKA** OOP (engl. Object-Oriented Programing).
- Kod **PROCEDURALNOG** programiranja aplikacija se gradi **KORAK PO KORAK** formiranjem niza programskih instrukcija.
- Programer je fokusiran na funkcije kojima pristupa i menja podatke.
- **OO PROGRAMIRANJE** mnogo više ima **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)

- Može se reći da su **ČETRI** osnovna principa OO programiranja:
  - **Apstrakcija**,
  - **Kapsuliranje**,
  - **Nasleđivanje** i
  - **Polimorfizam**.



- **OO programiranje** je inspirisano pojmom **APSTRAKCIJE** koja se bazira na procesu **ZANEMARIVANJA DETALJA** kako bi se uočila **funkcija**.
  - **Primer automobila**: Da li automobil zamišljate kao složeni mehanizam od 10000 delova?! **Ne!**
  - Automobil se prihvata kao **dobro definisan objekat** koji se ponaša na **jedinstven** način.
  - Apstrakcija automobila omogućava da se prevezete na odredište **NE OPTEREĆIJUĆI** se **složenošću mehanizma** samog automobila.
- Složene **PROGRAMSKE STRUKTURE** i **PODACI** su "**ZAPAKOVANI U KLASE**" kojima se pristupa na **strogo definisan način**.

Kapsuliranje u Javi (1)

- Pojam **KAPSULIRANJA** se odnosi na mehanizam koji povezuje **PODATKE** i **NAREDBE** sa kojima one rade.
- Programski kod koji je **KAPSULIRAN** pravi **ZAŠTITNU ČAURU oko podataka i naredbi NEDOZVOLJAVAJUĆI** proizvoljan – nekontroliani pristup.
- Pristupanje **kapsuliranim podacima** i **naredbama** je **STROGO DEFINISANO**.
  - Primer iz automobila: Menjač brzina se može šaltati **samo** preko ručice menjača, i to **nema uticaja** na migavce automobila!
- 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**.
- **Funkcije** (zapamtite: funkcije su **PROCEDURE** u proceduralnom programiranju), u **OO programiranju** se nazivaju **METODE** (one rade sa podacima).

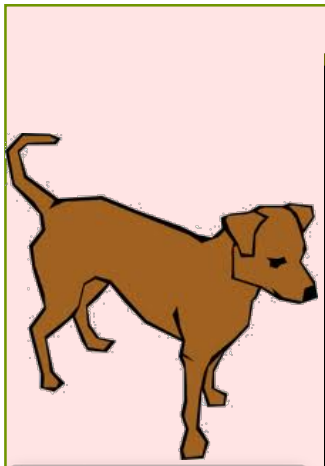
Kapsuliranje u Javi (2)

- **PODACI** se nazivaju i **članovima klase – i predstavljaju PROMENLJIVE KLASE**.
- **KLASOM** se definiše **STRUKTURA** i **ZAJEDNIČKO PONAŠANJE OBJEKTA** (nastalog – iniciranog od klase).
- **OBJEKTI** su **inicirani** (kaže se i **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**.
- Na ovaj način se **KAPSULIRA SLOŽENOST** programske strukture i podataka.
- Istovremeno se na ovaj način obezbeđuje **ZAŠTITA** od **zlonamernog** pristupa **podacima** ili **metodama**.

Nasleđivanje u Javi (1)

- **NASLEĐIVANJE** (engl. inheritance – nasleđe, baština) je **PROCES** u OO programiranju kojim jedan objekat **NASLEĐUJE SVE OSOBINE** drugog.
- Ovaj koncept **NASLEĐIVANJA** podrazumeva **HIJERAHIJSKU KLASIFIKACIJU** objekata.
- Bez postojanja **hijerarhije**, svakom objektu bi morali **IZNOVA** zadavati **SVE njegove karakteristike!**
- Sa konceptom **NASLEĐIVANJA** dovoljno je samo **definisati SPECIFIČNE KARAKTERISTIKE** tog objekta koje će ga činiti **jedistvenim u klasi**, ali sa **SVIM** prethodno definisanim (prihvatljivim) karakteristikama.
- **OPŠTA SVOJSTVA** objekti **nasleđuju** od **svojih "roditelja"**.
- Pogledajmo na sledećem slajdu **ilustrovan koncept nasleđivanja** primenjen u **OO programiranju**.

Nasleđivanje u Javi (2)



Generički pas

PAS
Boja
Boja očiju
Visina
Dužina
Težina
Leži
Protresi se
Dođi
Sedi

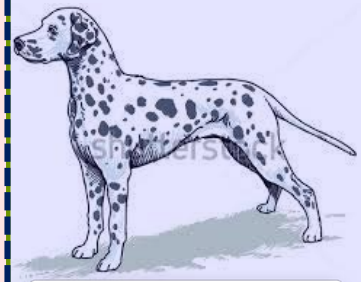
Generičke – OPŠTE OSOBINE (svojstva)

Metode (komande)

KLASA

Operator NEW

KONKRETNE osobine (svojstva)

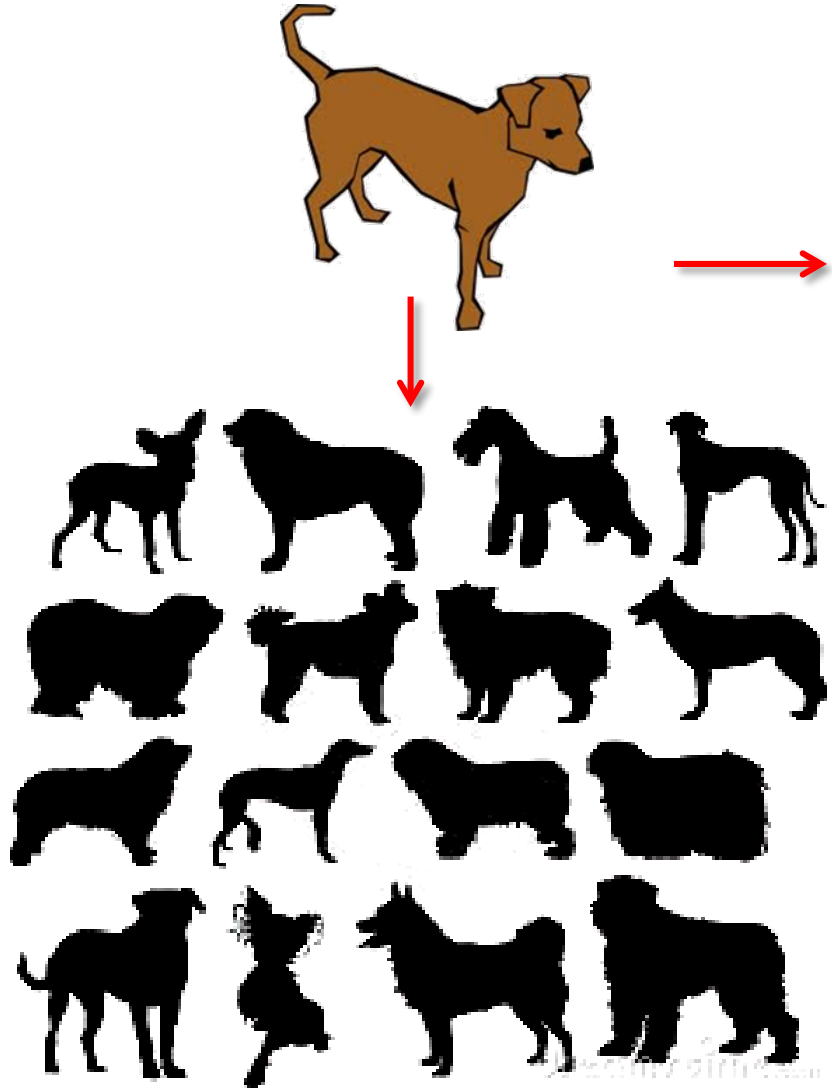


Konkretan pas

PAS
Siva
Bela
30 cm
100 cm
15kg
Leži
Protresi se
Dođi
Sedi

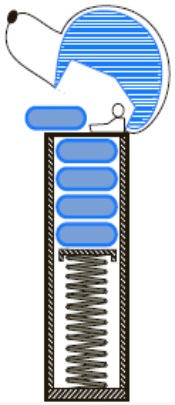
OBJEKT(Instanca)

Instance u Javi



Polimorfizam u Javi (1)

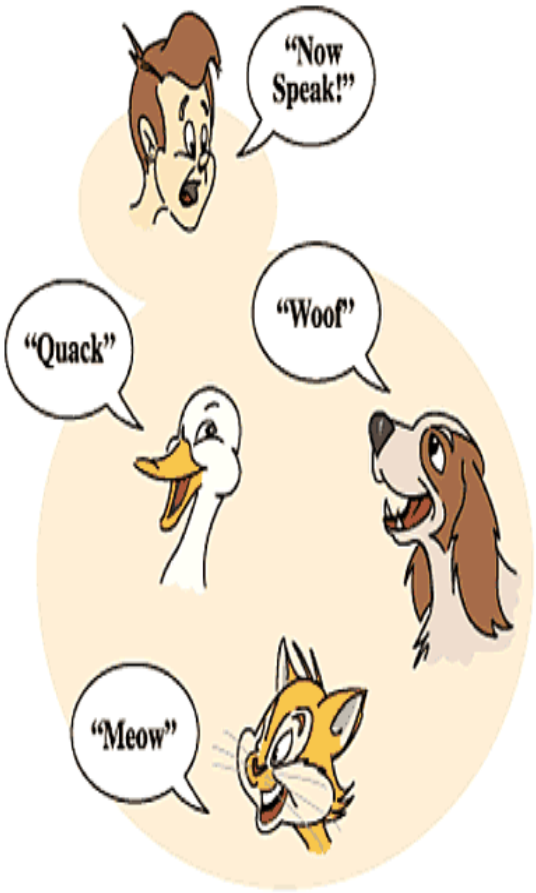
- **POLIMORFIZAM** se zasniva na **JEDINSTVENOM NAČINU PRISTUPA** za **opštu klasu akcija** (**kontekst poziva akcije** određuje njenu specifičnost).
- Primer_1: Korišćenje **čula mirisa kod psa**.
  - Jednim **ISTI ČULOM** pas razlikuje i **mačku** i **hranu**!
- Primer_2: Korišćenje **STEK STRUKTURE** za različite tipove podataka.
  - Algoritam kojim se kreira i upravlja stek strukturom je **UVEK ISTI** i ne zavisi od toga da li se smeštaju iteger, char ili object tipovi podataka.
  - U **OO** programiranju je dakle moguće napisati **OPŠTI PROGRAM ZA RAD SA STEKOM**, a metode će imati ista imena nezavisno od tipa podataka.
  - Kod **proceduralnih i funkcionalnih jezika** moraju se napisati **tri verzije metoda** - po jedna za svaki tip podataka.



Primer Stek-a

Polimorfizam u Javi (2)

- **Kod Jave**, kompajler (prevodilac, JIT) **ODREĐUJE METODU** koja će biti **pozvana** u zavisnosti od **KONTEKSTA** samog poziva!
- Ovo ima za posledicu **smanjenje složenosti** programskog **koda**.
- Tokom vremena, **može** se menjati **UNUTRAŠNJA KONSTRUKCIJA KLASE** ne remeteći kod koji se oslanja na **JAVNI INTERFEJS** klasa.
- Koncept Jave je **slično kao i C#** zasnovan na **SPECIJALIZOVANOM IZVRŠNOM OKRUŽENJU** – **Javinoj Virtualnoj Mašini - JVM**.
- Pogledajmo na sledećem slajdu **koncept** Jave kojim se prevazilazi problem **HETEROGENOSTI terminalne opreme**, što je jedan od problema vezan za Internet programiranje.



Javina virtuelna mašina JVM

Izvorni kod

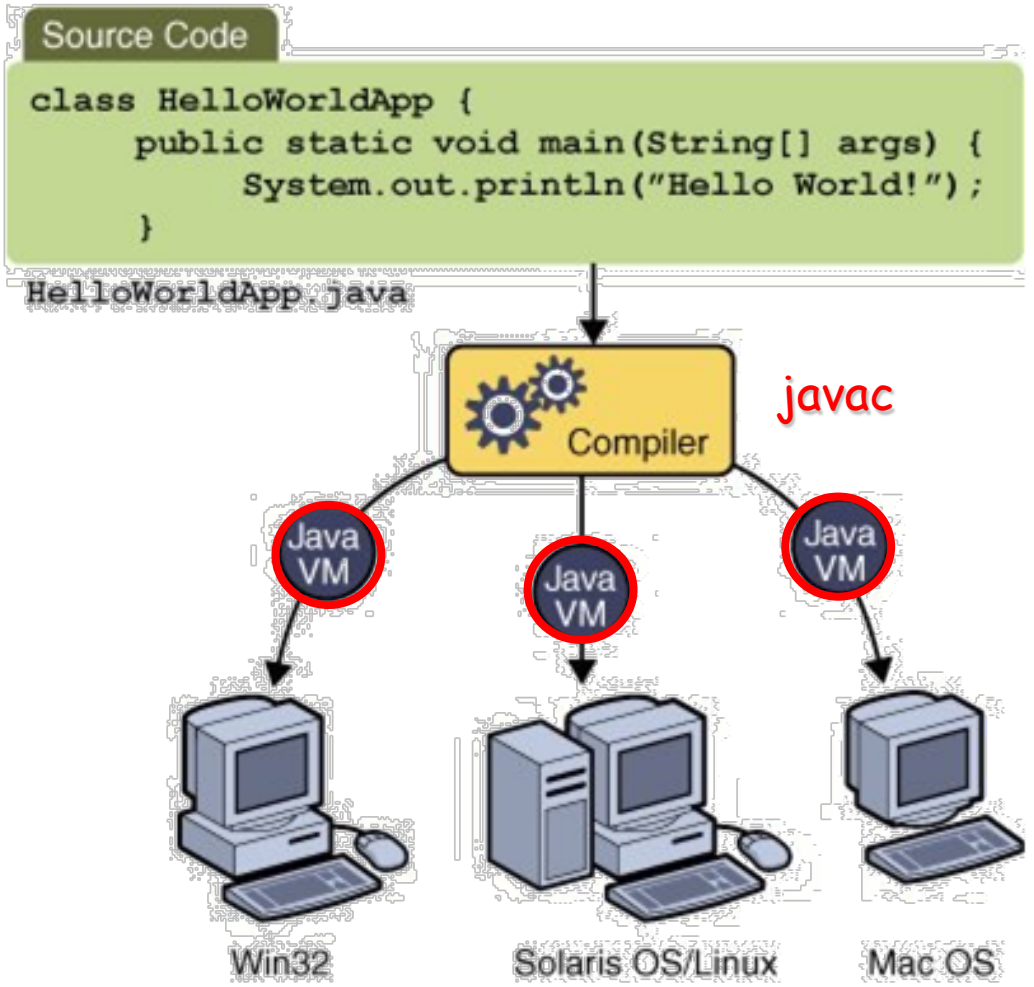
*.java

Bajtkod

*.class

Izvršni kod

*.exe

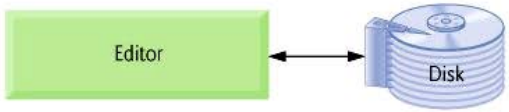


Java i Bytecode

- **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) i 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!

RAZVOJNE FAZE

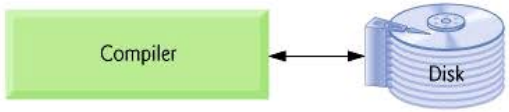
Faza 1: EDITOVANJE



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

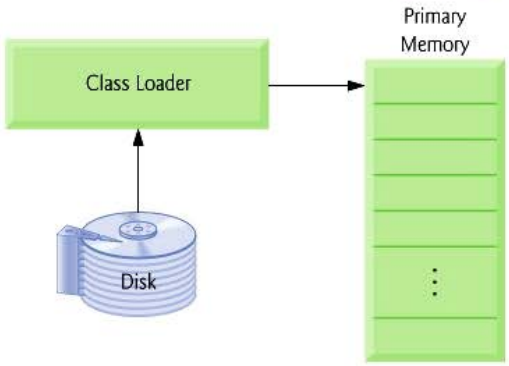
Faza 2: KOMPAJLIRANJE

javac



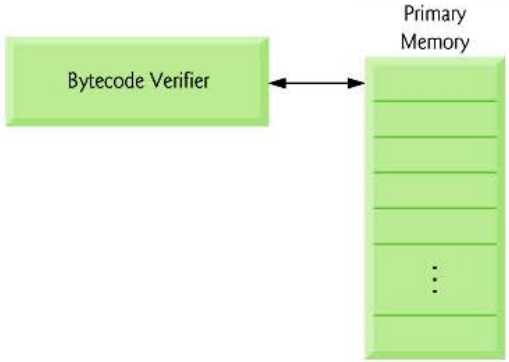
Kompajler formira bytecode i smešta ga na disk u formi .class.

Faza 3: PUNJENJE



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

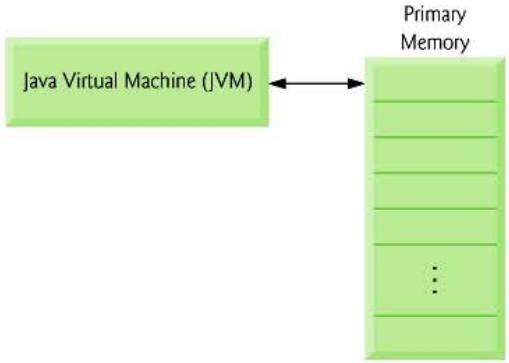
Faza 4: VERIFIKACIJA



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

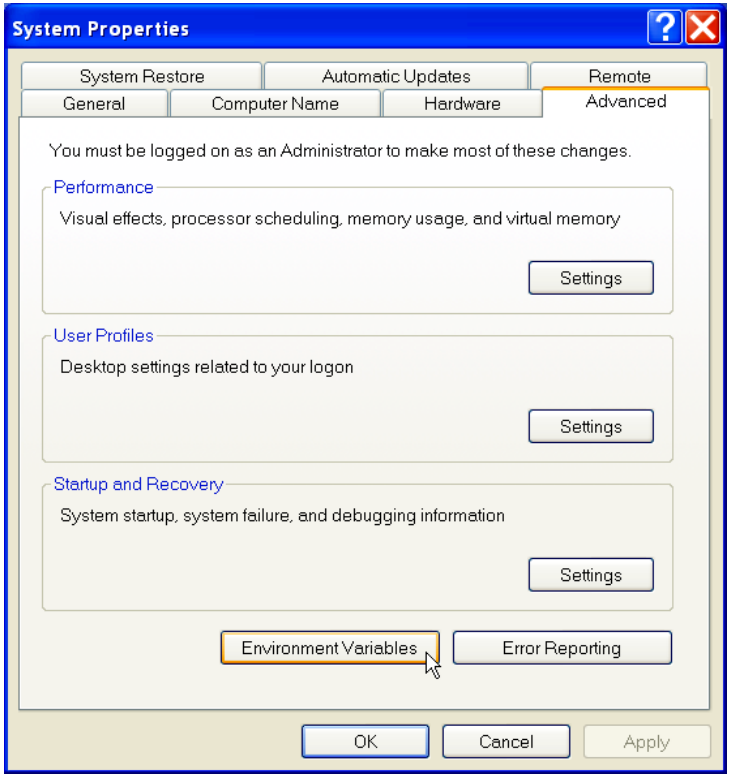
Faza 5: IZVRŠENJE

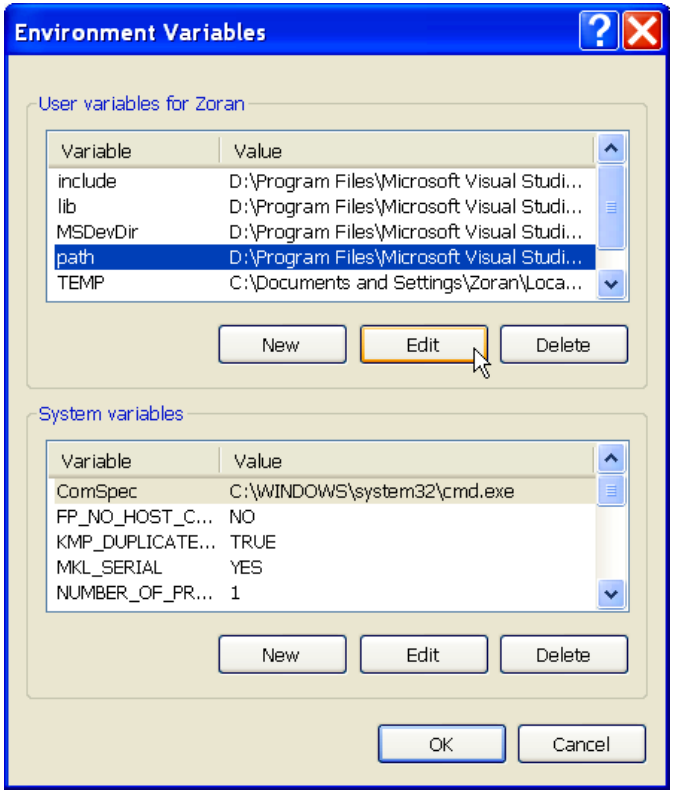
JIT/java



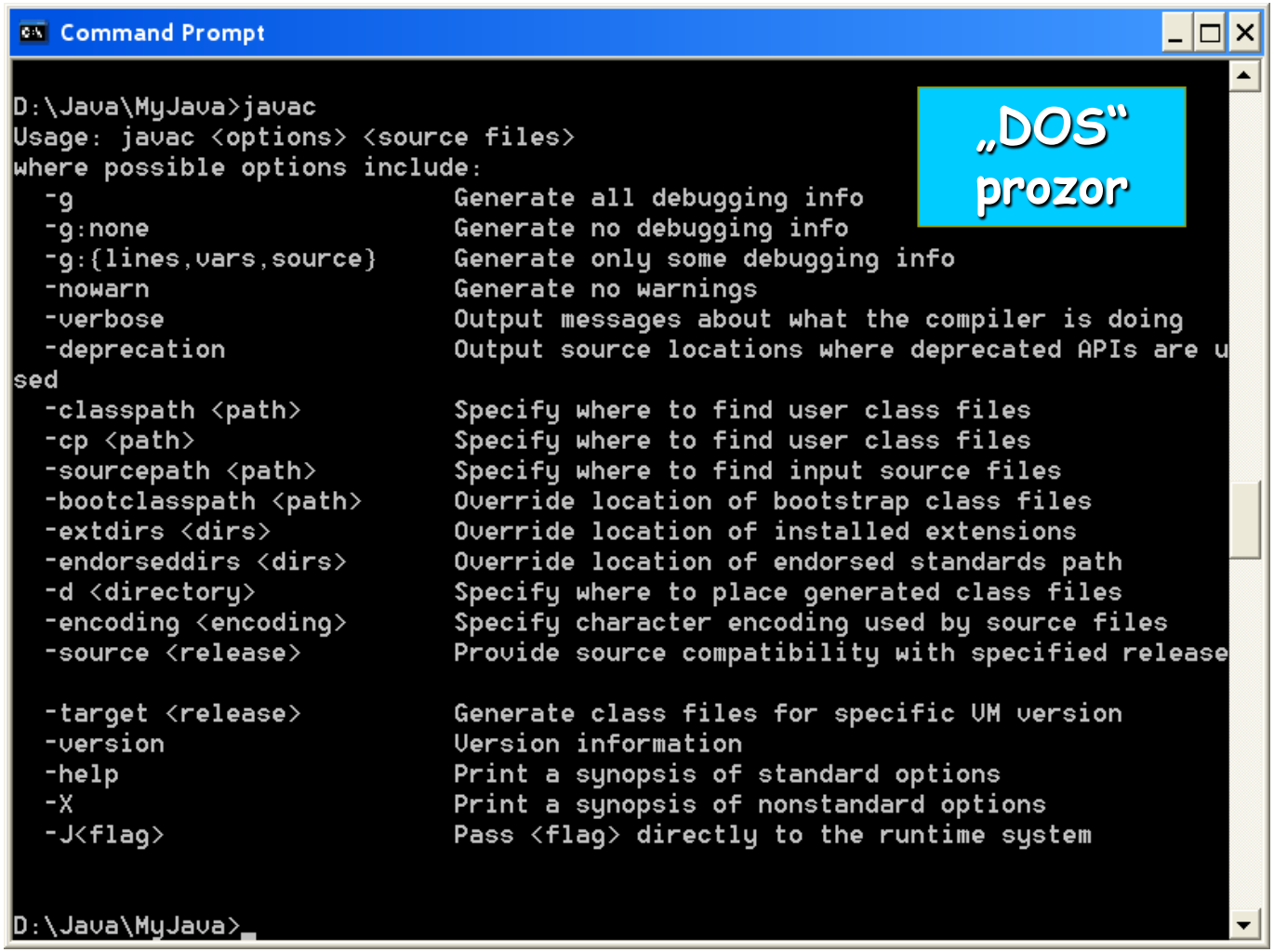
JVM učitani bytecode prevodi u jezik koji procesor razume i izvršava

Java: setovanje J2SE 5

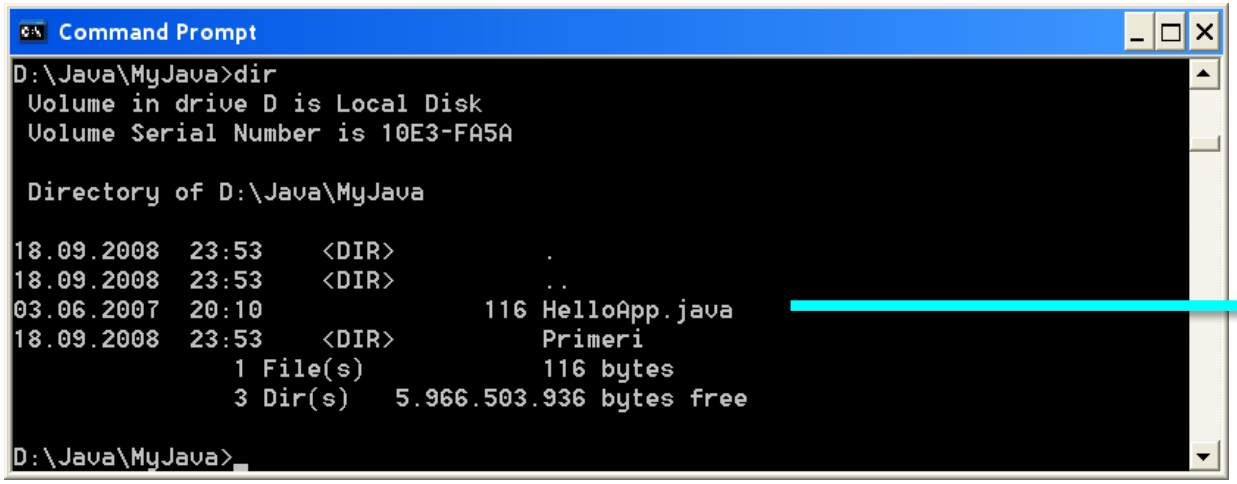




Konzolni kompajler: javac

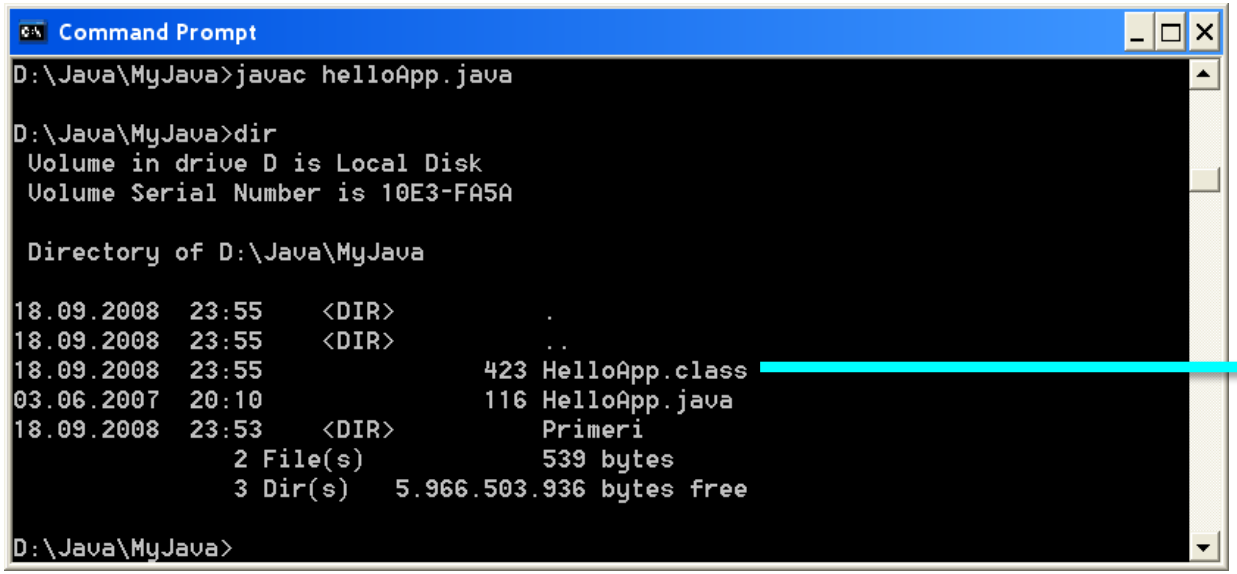


Java: .java i .class datoteke



Izvorni kod,

.java

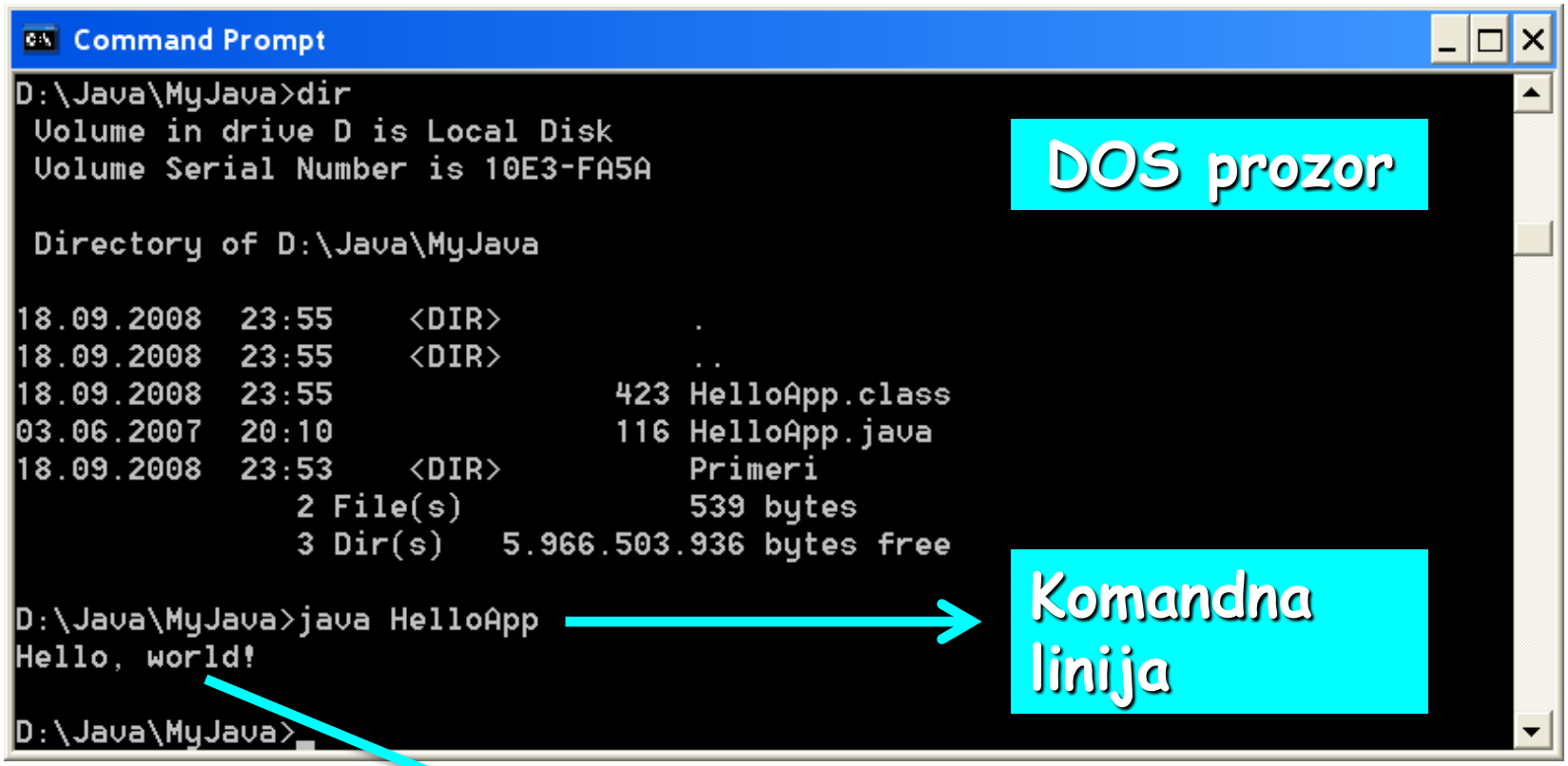


Bajt kod,

.class

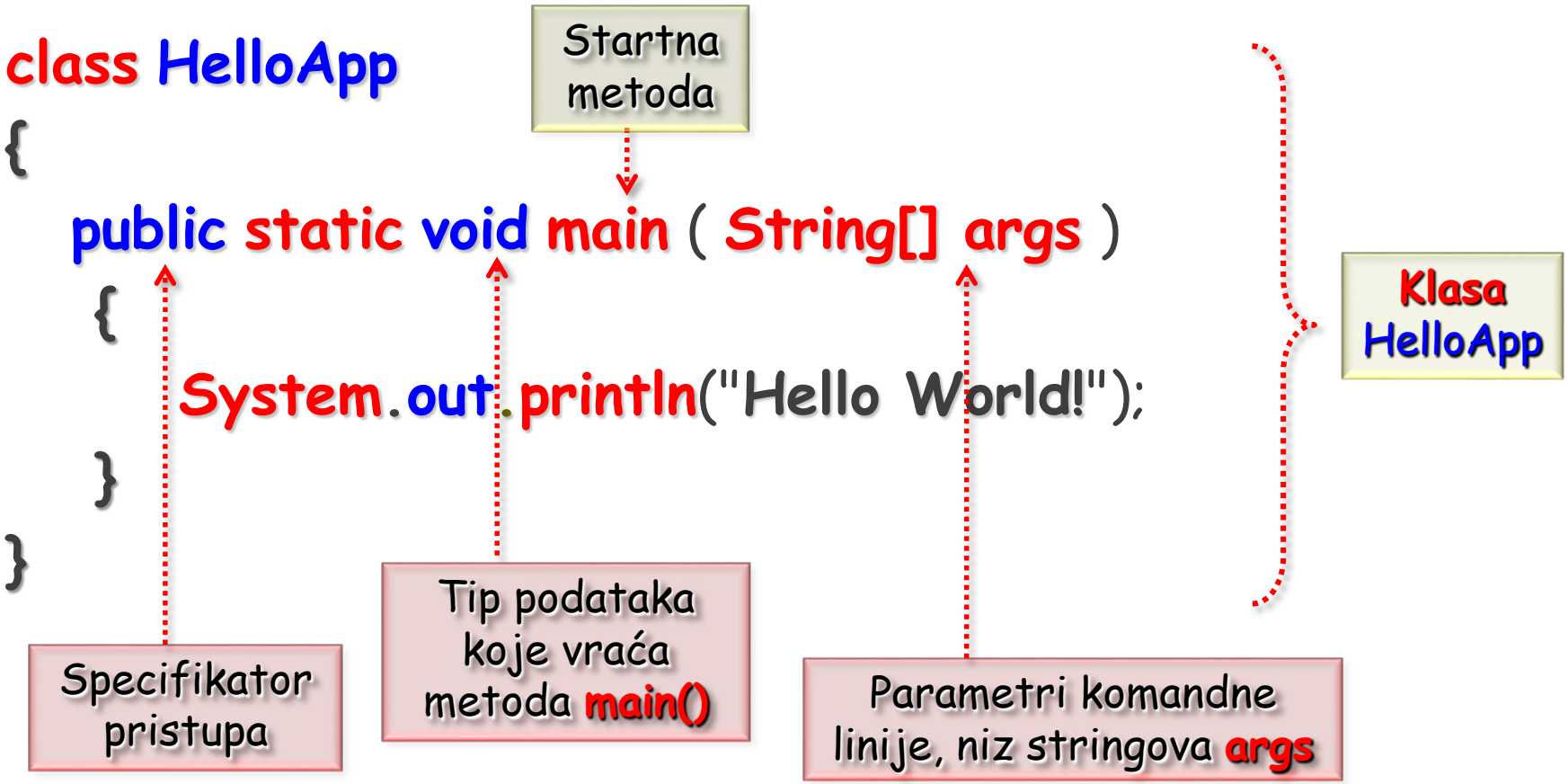
Java: izvršenje programa

- Izvršavanje sa komandne linije: **java HelloApp**



Java-prvi program

- U Javi je sve u **KLASAMA**, pa i sama aplikacija **HelloApp**!



Java: 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 ispisuje tekst: **Hello world**

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

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

Java: Prostor imena

- 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 (paketa).

- Umesto:

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

piše se:

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