1

Objektno orijentisano programiranje
vežbe

Staša Vujičić Stanković

Karakteristike programskog jezika Java

- **Mašinski nezavisna.**
  - Piši jednom, izvršavaj bilo gde!
  - Java program se nepromenjen može pokrenuti na bilo kojoj mašini i operativnom sistemu koji podržava JAVU.
- **Objektno-orijentisani programski jezik.**
  - OO programi su lakši za razumevanje, jer simuliraju spoljni svet.
    Njihovo proširivanje i održavanje je jednostavnije i oduzima manje vremena.

Osnovne vrste programa u Javi

- **Java aplikacije**

  Java aplikacije su samostalni programi.
- **Java apleti**

  Java apleti su dinamički, interaktivni programi koji su umetnuti u Web stranu.

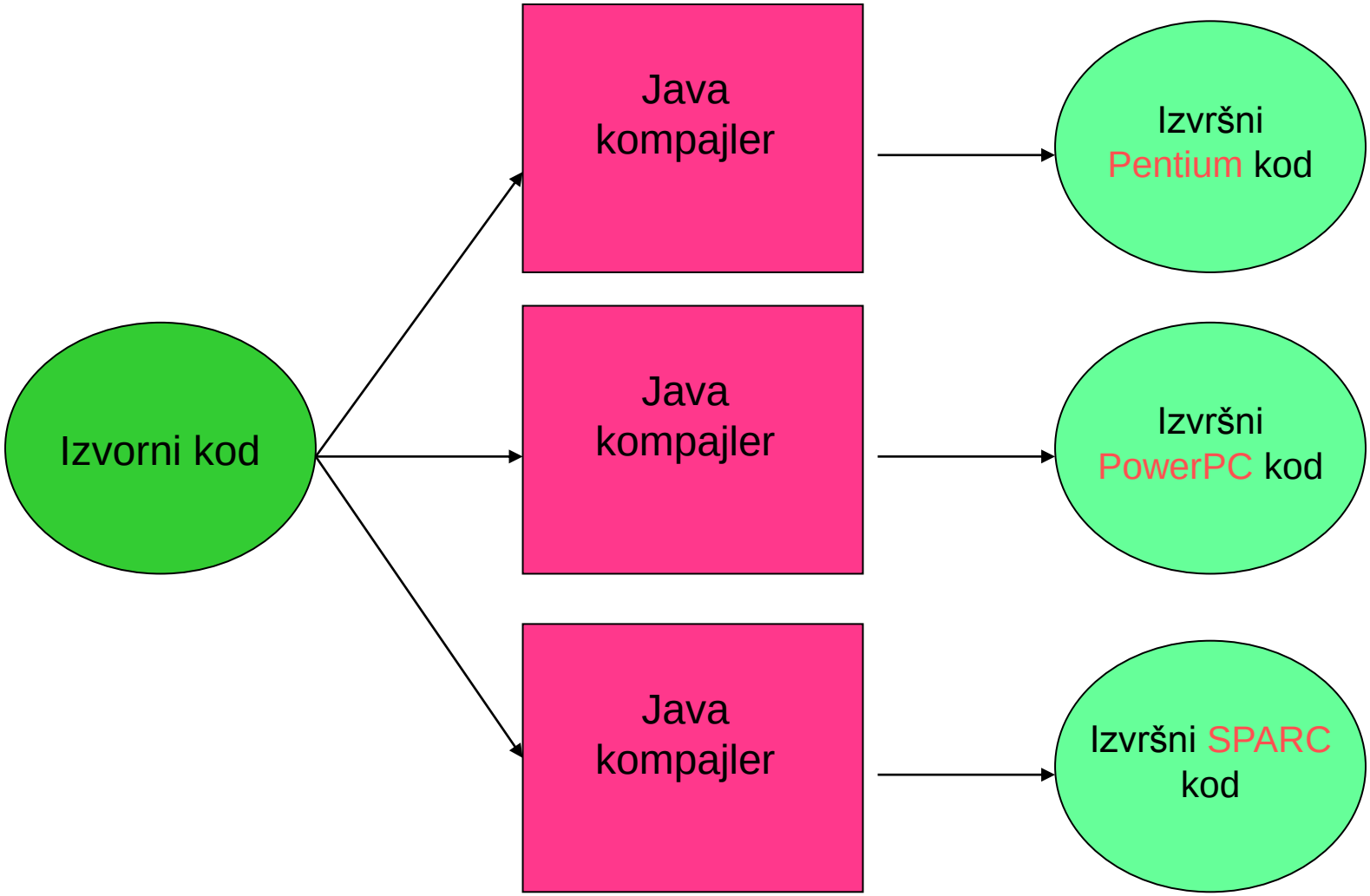
Pokretanje JAVA programa

- Java programi se mogu pokrenuti na širokom spektru operativnih sistema.
  Radiće jednako dobro i na PCu sa proizvoljnim Windowsom, Linuxom ili na Solarisu.
- Ovo je moguće zato što se JAVA program ne izvršava direktno na računaru.
- Pokreće se u standardizovanom okruženju koje se naziva **JAVA platforma** i implementira kao softver na širokom spektru računara i operativnih sistema.

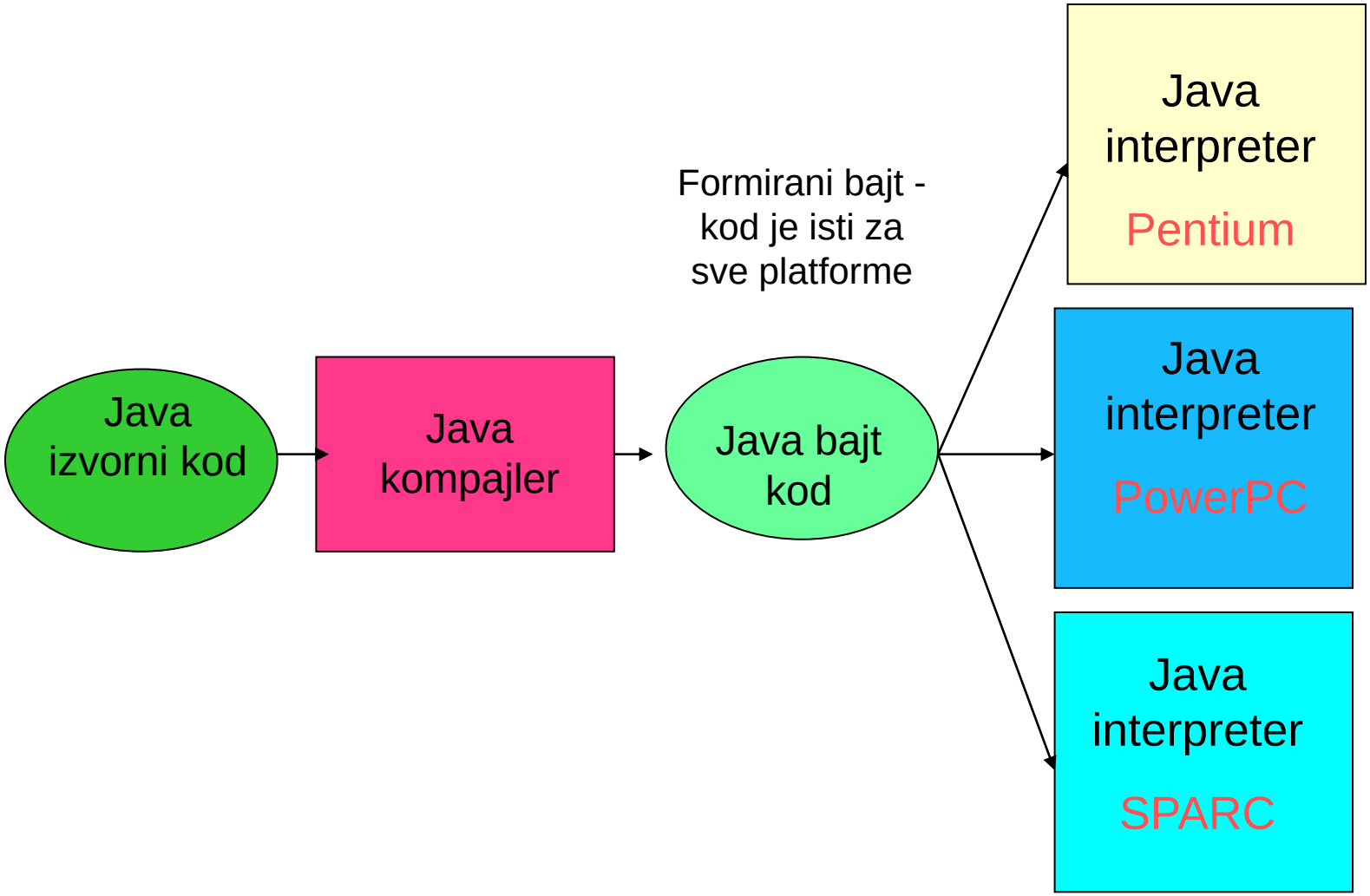
JAVA PLATFORMA

- Sastoji se od dva elementa:
- **JAVA virtuelna mašina - JVM**
  - Softverska implementacija hipotetičkog računara.
- **JAVA application programming interface - API**
  - Skup komponenti koje obezbeđuju pisanje interaktivnih aplikacija u Javi –
    rad sa grafičkim korisničkim interfejsom,
    već kreiran skup klasa raspoloživih programeru za korišćenje.

Tradicionalni način obrade izvornog programa



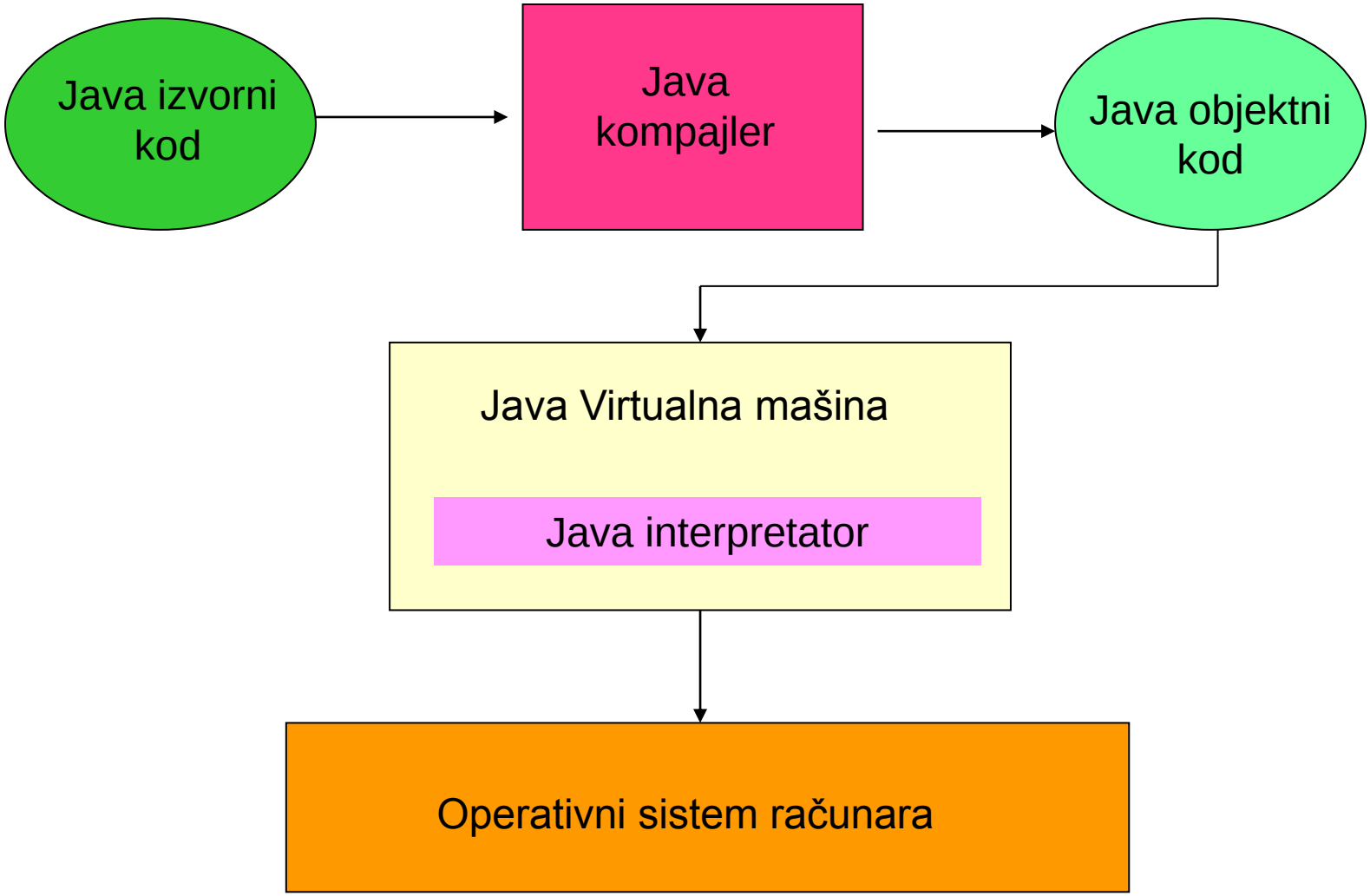
Način obrade izvornih kodova napisanih u Javi



IZVRŠAVANJE JAVA PROGRAMA

- Java kompajler konvertuje Java izvorni kod u binarni program koji se sastoji iz bajtkoda.
- **Bajtkod (Bytecode)** su mašinske instrukcije za JVM.
- Kada se izvršava Java program, program koji se zove **Java interpreter**, tumači bajtkod i izvršava akcije koje su specificirane u bajtkodu, unutar JVM.

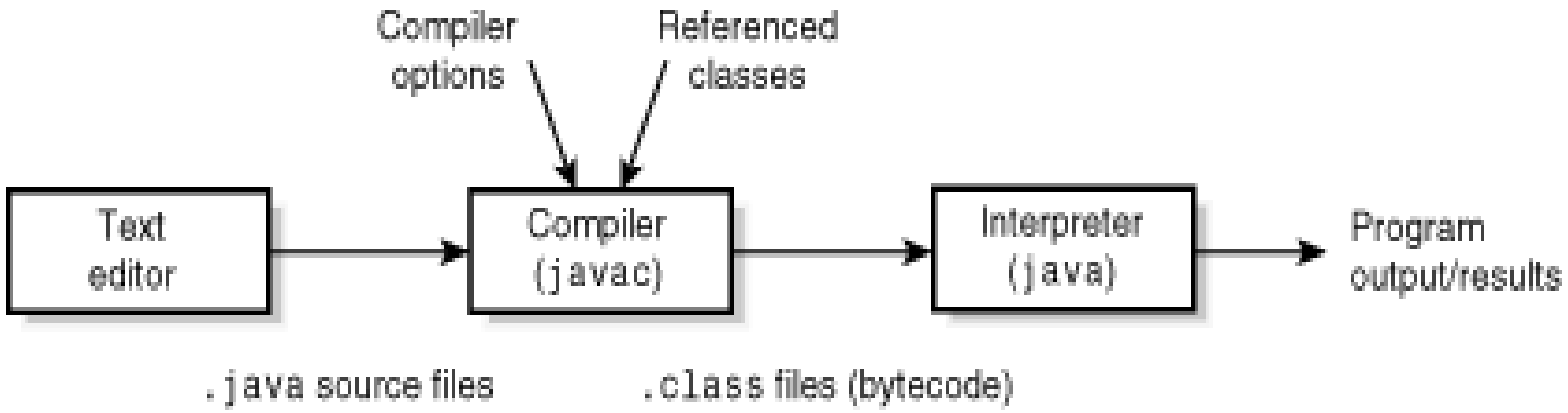
Izvršavanje JAVA programa



IZVRŠAVANJE JAVA PROGRAMA

- Java izvorni kod, pišemo u text editoru, i snimamo sa ekstenzijom **.java** (ZdravoSvete.java).
- Kompajliramo ga iz komandne linije komandom: **javac ZdravoSvete.java**
- Kompajler generiše bytecode program koji je ekvivalentan izvornom kodu, nosi isto ime i ekstenziju **.class**
- Da bi se izvršio bajtkod program Java interpreterom piše se komanda: **java ZdravoSvete**

IZVRŠAVANJE JAVA PROGRAMA



Prvi Java program Zdravo svete!!!

```
public class ZdravoSvete
  {
        public static void main(String[] args)
                {
                System.out.println("Zdravo svete");
                }
  }
```

Prvi Java program Zdravo svete!!!

- Snimimo zadatak kao: **ZdravoSvete.java**
- Kompajliramo pozivom komande:
  javac ZdravoSvete.java
- Izvršavamo pozivom komande:
  java ZdravoSvete

Mogući problemi

- Ukoliko kompilacija ne prođe,
  česti uzroci mogu biti:
  - PATH nije uključen ili nije ispravno uključen
  - case-sensitive: imena klase i fajla se moraju potpuno poklopiti i u njima ne smeju postojati beline
  - . , ; su bitni znaci i moraju biti na svojim mestima
  - ( ) { } i [ ] uvek dolaze u paru i ne prepliću se!

Objektno orijentisano programiranje u Javi

- Kada se koristi programski jezik koji nije OO, rešenja se izražavaju u terminima brojeva i karaktera.
- Kod OO programskih jezika na brojeve i karaktere se gleda kao na primitivne tipove, a problemi se rešavaju u terminima entiteta, tj. objekata koji se javljaju u kontekstu problema.

Objektno orijentisano programiranje u Javi

- Problem: Organizacija voznog parka u firmi.
- Šta su objekti?
- Neki od objekata u Java programu:
  prevozno sredstvo (generalno posmatrano),
  ali i autobus, automobil...

Objektno orijentisano programiranje u Javi

- **Objekat** – integralna celina podataka i procedura za rad sa njima.
- **OOP** – programska paradigma zasnovana na skupu objekata koji dejstvuju međusobno. Glavne obrade se zasnivaju na manipulisanju objektima.

Objektno orijentisano programiranje u Javi

- **Klasa** – Skup objekata sa zajedničkim svojstvima. Npr. prevozno sredstvo je klasa.
- **Potklasa** – klasa koja nasleđuje sve osobine roditeljske klase (**nadklasa**) i ima dodatnu specijalizaciju.
  Npr. automobil ima sve karakteristike klase prevozno sredstvo, ali ima i svoje dodatne karakteristike – npr. 4 točka...

Objektno orijentisano programiranje u Javi

- **Instanca** – konkretan objekat iz klase.
- **Instancne promenljive ili atributi klase** – parametri koji definišu objekat neke klase. Mogu biti osnovni tipovi podataka ili objekti neke druge klase.
- Npr. automobil:
  - boja: bela, crvena, crna,...
  - proizvođač: Honda, Opel, BMW, ...
  - model: limuzina, sportski, karavan...

Objektno orijentisano programiranje u Javi

- Primer:

```
class Krug {                    ← klasa Krug
    int x, y;
    float r;
    }
Krug k = new Krug(10,20,5);   ← instanca klase Krug
```

- x, y i r su instancne promenljive koje u instanci k imaju redom vrednosti 10, 20 i 5.

Objektno orijentisano programiranje u Javi

- Klasa pored kolekcije podataka specifikuje i šta sve može da se radi sa objektima te klase. Definiše listu mogućih operacija nad objektima klase. To se postiže blokom programskog koda koji se naziva **metoda**.
- Na primer:
  - Startuj motor
  - Zaustavi auto
  - Ubrzaj...

Objektno orijentisano programiranje u Javi

- **Metod** – funkcija koja je sastavni deo objekta, tj. postupak kojim se realizuje poruka upućena objektu.
- **Poruka** – skup informacija koji se šalje objektu. Sastoji se od adrese objekta primaoca poruke i saopštenja u kome mu se kazuje šta treba da uradi.

Enkapsulacija u Javi

- U OO jezicima bitan je koncept **enkapsulacije** što podrazumeva skrivanje podataka i metoda unutar objekta.
- Enkapsulacija je bitna za bezbednost i integritet objekata.
- Ovo se postiže ključnom rečju **private**.

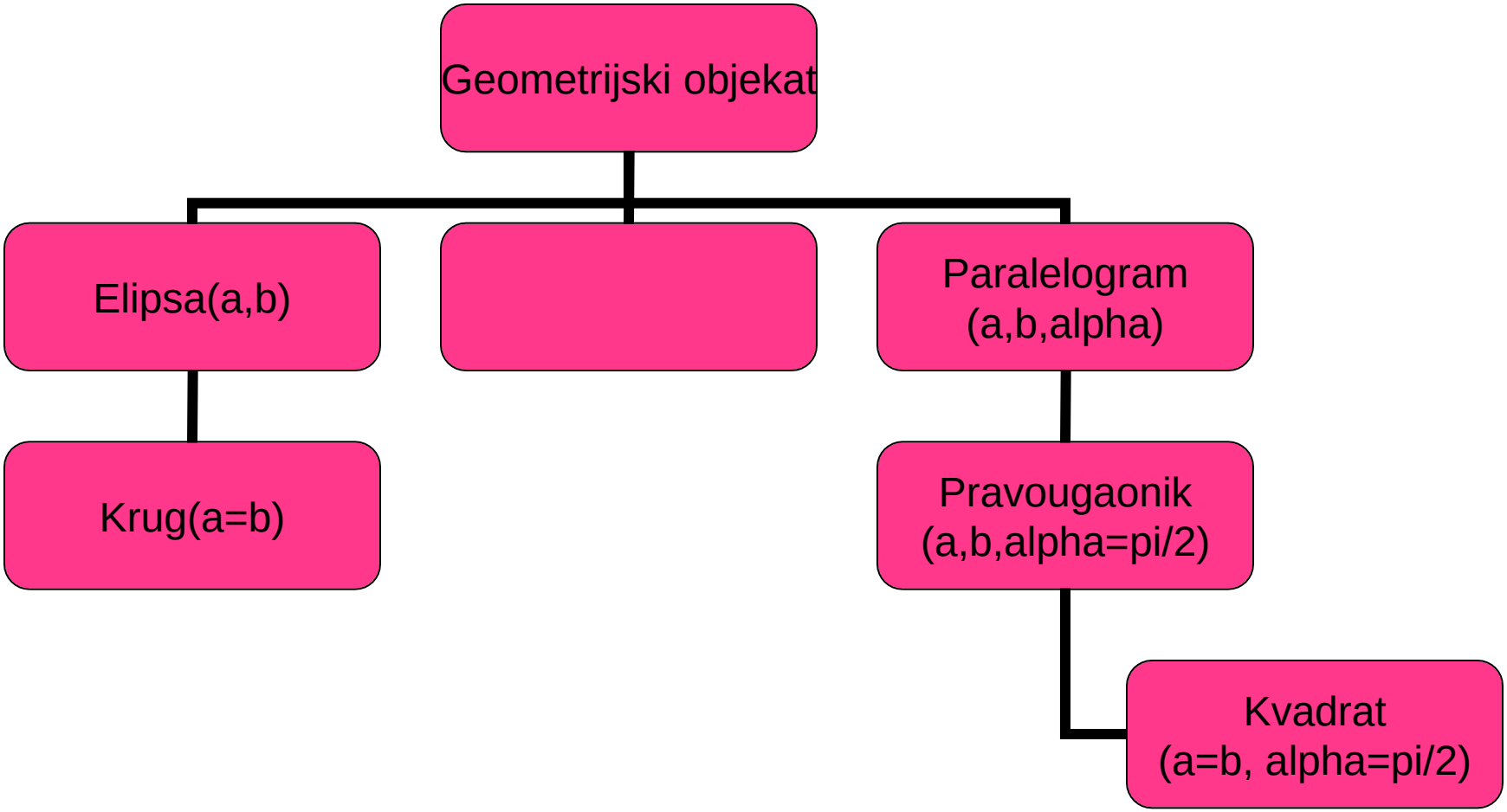
Enkapsulacija u Javi

- Ključna reč **private** ispred instancne promenljive obezbeđuje da samo kôd unutar metoda klase može da pristupa i menja direktno njenu vrednost.
- Metod takođe može biti privatni.

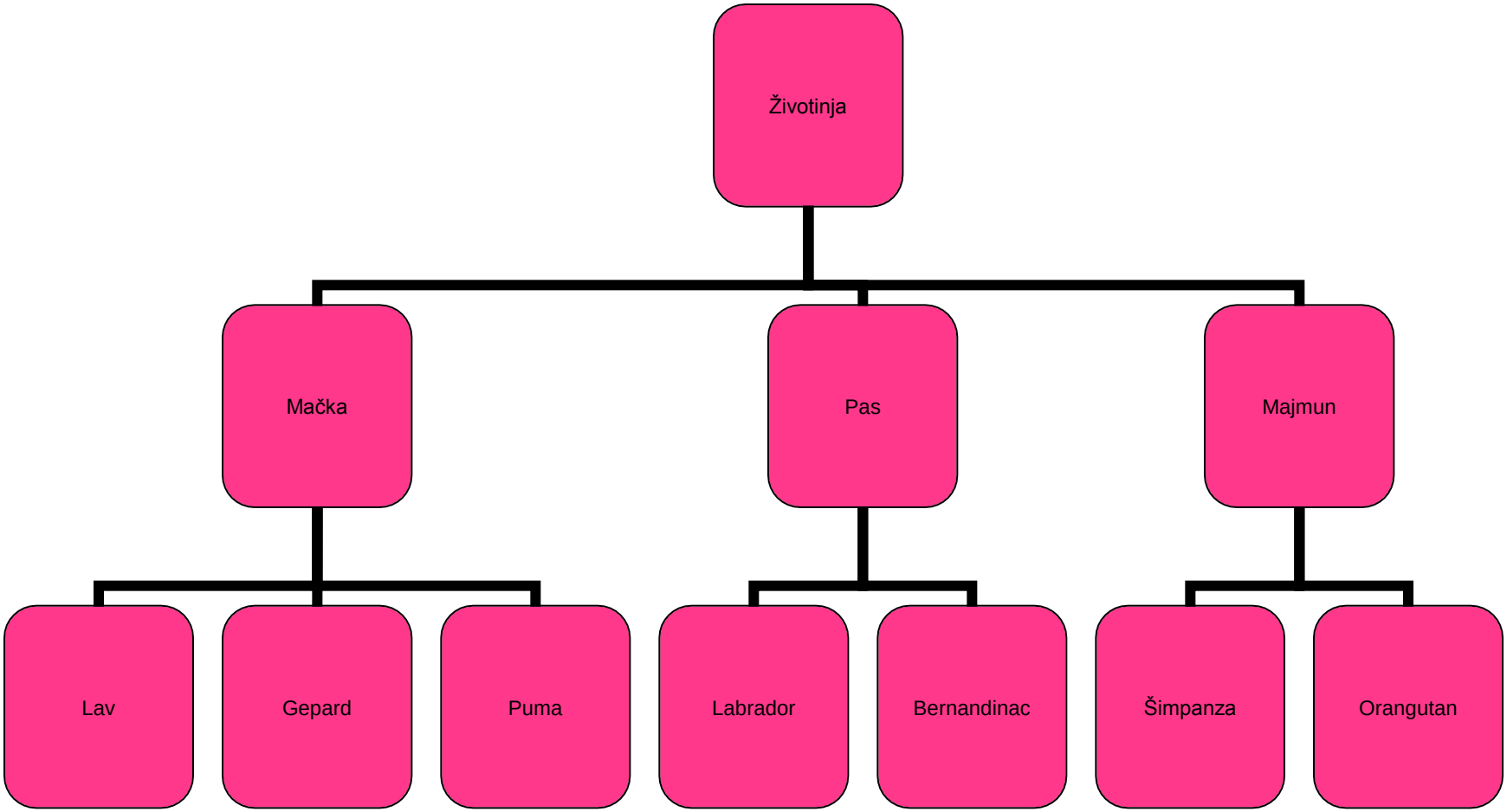
PRIMERI NASLEĐIVANJA

- Prilikom projektovanja programa uočavaju se veze između pojedinih klasa i način povezivanja tih klasa sa drugim klasama. Ovde je bitna uloga nasleđivanja.

PRIMERI NASLEĐIVANJA



PRIMERI NASLEĐIVANJA



Struktura Java programa

- Java program se uvek sastoji iz jedne ili više klasa.
- U Javi imamo klasu **Object** koja je natklasa svih klasa .

Struktura Java programa

- Uobičajeno je da se svaka klasa stavlja u poseban fajl koji ima isto ime kao i klasa sadržana u njemu.
- Java izvorni fajl mora da ima ekstenziju **.java**.
- Svaka Java aplikacija sadrži klasu koja definiše metod *main()*.
  Ime ove klase je ime koje dajemo kao argument Java interpreteru.

Biblioteka klasa u Javi

- **Biblioteka u Javi** - kolekcija već postojećih klasa.
- Klase su grupisane u odgovarajuće skupove koji se nazivaju **paketi**.
- Korisnik sam može da pravi svoje pakete koji će sadržati odgovarajuće klase.

Standardni paketi

- Java raspolaže velikim brojem standardnih paketa. Najčešće se koriste:
- **java.lang** – osnovne karakteristike jezika, rad sa nizovima i stringovima.
  Klase iz ovog paketa su uvek dostupne korisničkom programu, tj. automatski se uključuju u program (npr. klase Integer, String, Math ...)
- **java.io** – klase za ulazno/izlazne operacije
- **java.util** – klase Vector(uopšteni niz), Stack, Scanner ...
- **java.awt** – paket za rad sa grafičkim komponentama(GUI)
- **java.swing** – klase koje obezbeđuju fleksibilne komponente za građenje GUI-ja.

Standardni paketi

- Da bismo koristili klase iz paketa (sem paketa java.lang), koristimo **import** deklaraciju sa imenom paketa ili klase:

  import java.util.Vector;

  import java.util.*; - ovim se uključuju sve klase iz paketa
- Može se koristiti klasa iz nekog paketa i bez import deklaracije, ali se mora navesti puno ime klase:

  java.util.Vector a;

Gde je biblioteka?

- Standardne klase su spakovane u jedan komprimovani fajl **rt.jar** koji se nalazi u **jre/lib** direktorijumu koji se kreira prilikom instalacije JDK-a.

Forma metoda main() - primer Zdravo.java

- Metod main() mora imati unapred određenu formu da bi bio prepoznat od strane Java interpretera kao metod odakle kreće izvršavanje programa.

```
public class Zdravo
{
        public static void main(String[] args)
        {
                System.out.println("Zdravo!");
        }
}
```

Forma metoda main() - primer Zdravo.java

- **public static void main(String[] args)**
  - **public -** metod je globalno dostupan
  - **static** – dostupan i kada ne postoje objekti klase
  - **void** – ne vraća vrednost
  - **args** – lista argumenata komandne linije koji su objekti klase String
    (broj argumenata se može odrediti pozivom args.length)

FORMA METODA MAIN() - PRIMER ZDRAVO.JAVA

- **System.out.println("Zdravo!");**
  - **System** – standardna klasa koja sadrži objekte za rad sa standardnim U/I uređajima. Sadržana je u paketu java.lang.
  - Objekat **out** predstavlja standardni izlazni tok i član je klase System.
  - **println()** metod pripada objektu out i ispisuje string zadat između dvostrukih navodnika.
- Poziv klasnog metoda –
  ime_objekta.ime_metoda

Programi, podaci, promenljive i izračunavanja

- Java koristi UNICODE, koji je 16-bitni kod, razvijen za podršku karakterima skoro svih jezika. ASCII odgovara skupu od prvih 128 karaktera UNICODE skupa.
- Java je CASE-SENSITIVE, tj. u Javi se pravi razlika između malih i velikih slova.

Programi, podaci, promenljive i izračunavanja

- Eksplicitne vrednosti podataka koje se javljaju u programu nazivamo **literalima**,
  npr. 15 je celobrojni literal tipa int.
- **Identifikator** može da bude proizvoljne dužine, a može početi slovom, donjom crtom ili dolarom. Ne smeju se koristiti ključne reči.
- Jedine stvari koje nisu objekti u Javi su promenljive koje odgovaraju jednom od osnovnih tipova podataka koji se nazivaju **primitivnim tipovima podataka**.

Primitivni tipovi podataka

- Celobrojni tipovi podataka:
  - **byte** – 8 bitova
  - **short** – 16 bitova
  - **int** – 32 bita
  - **long** – 64 bita

    (svi su signed, tj. čuvaju i pozitivne i negativne brojeve)
  - Literal tipa long ima sufiks L ili l.
  - Za heksadekadnu osnovu koristi se prefiks 0x ili 0X.

Primitivni tipovi podataka

- Tipovi podataka u pokretnom zarezu:
  - **float** – 32 bita
  - **double** – 64 bita
  - Literali u pokretnom zarezu su podrazumevano tipa double.
  - Kada hoćemo da naglasimo da je vrednost tipa float dodamo sufiks f ili F.

Podaci, promenljive, izračunavanja

- Ključna reč **final** označava da se vrednost promenljive ne može menjati.
- Konvencija je da se konstante pišu velikim slovima.
- **Kastovanje.**
  - Primeri:

```
int a=3;
int b=5;
double c=1.5+b/a;
double d=1.5+(double)b/a;
```

- Ne postoji automatska konverzija boolean → int i obrnuto
  (ne može se pisati npr. if(a), već mora if(a!=0))!

Podaci, promenljive, izračunavanja

- Promenljiva se može uvesti bilo gde u bloku. Oblast važenja je od tog mesta do kraja bloka.
- If, else, switch, for, while, do, break, continue – isto kao u C-u.
- Goto ne postoji.
- Kreiranje objekta neke klase vrši se operatorom **new**.

Rad sa stringovima

- Postoje dve klase
  String (za konstantne stringove) i
  StringBuffer (za promenljive stringove).
- Može se koristiti i deklaracija pomoću char[],
  ali ne postoji automatska konverzija između niza
  i stringa!
  - Primer:

```
char ime[] = {‘a’, ‘l’, ‘e’, ‘k’, ‘s’, ‘a’, ‘n’, ‘d’, ‘a’, ‘r’};
// ispravno
char ime[] = “aleksandar”;  // ne sme
String ime = “aleksandar”; // ispravno
```

Rad sa stringovima

- Metode:
  - **length()** – dužina stringa
  - **charAt(int)** – pristup elementu na datoj poziciji
  - **Operator +** se koristi za spajanje(konkatenaciju) stringova
    - String s = "Rezultat je:" + rez;
      // rez je npr. tipa int
      // Postoji automatska konverzija int → String.

    ...
- Generalna oznaka za prazan objekat je **null**.
  - String a = null;

Stringovi, argumenti komandne linije

- public static void main (String[] args)

  ↑ niz stringova

  args.length dužina niza
- args[0], args[1], … to su ti stringovi
- C:\> java Zdravo a b c

  nisu argumenti | 0 1 2

  ↑ ↑ ↑ argumenti

Klasa Scanner

- **Scanner** je klasa koja implementira jednostavni skener koji parsira primitivne tipove i stringove koristeći regularne izraze.
- Ulaz se deli na tokene, a kao delimiter se podrazumevano koristi karakter space (blanko).

KLASA SCANNER

- Rezultujući tokeni se mogu konvertovati u vrednosti različitih tipova koristeći metode:
  - **next()** Učitava sledeći kompletan token.
    nextDouble() Učitava sledeći token kao double.
  - **nextInt()** Učitava sledeći token kao int.
  - **nextLine()** Učitava sledeću liniju.
- Za svaki od ovih metoda postoje odgovarajući metodi kao što je:
  - **hasNextInt()** Vraća true ako se sledeći token sa ulaza može pročitati kao int vrednost pozivom metoda nextInt().
- Primer:

```
Scanner sc = new Scanner(System.in);
int i = sc.nextInt();
```

KLASA SCANNER

- Jedan od metoda za promenu delimitera je: **useDelimiter(String pattern).**

```
import java.util.Scanner;

public class ParseString
{
    public static void main(String[] args)
    {
        Scanner scanner =
        new Scanner("1, 2, 3, 4, 5, 6, 7, 8, 9, 10").useDelimiter(", ");

        while (scanner.hasNextInt()) {
            int num = scanner.nextInt();

            if (num % 2 == 0)  System.out.println(num);
        }
    }
}
```

ZADATAK - ISPISARGKOMLINIJE.JAVA

```
public class IspisArgKomLinije {
/*  Ispisujemo argument komandne linije.   */
   public static void main(String[] args) {

          // Ispisujemo pozdravnu poruku.
          System.out.print("Zdravo ");

          if (args.length>0)
          {   System.out.println(args[0]); }

          if (args.length <= 0)
          {   System.out.println("ko god da si!"); }
   }
}
```

ZADACI:

1. Napisati program koji izračunava obim i površinu kruga datog poluprečnika r=12.45, i ispisuje dobijene vrednosti na standardni ulaz.

2. Napisati program koji sabira prirodne brojeve od 1 do n.

3. Napisati program koji učitava dva broja sa standardnog ulaza i ispisuje njihov zbir na standardni izlaz.

Zadatak – Krug.java

```
public class Krug
{  /* Racunamo obim i povrsinu kruga datog
   poluprecnika r.*/
   public static void main (String args[])
   {
         double r = 12.45;
         double O = 2 * r * Math.PI;
         double P = r * r * Math.PI;

         // Pisi na standardni izlaz.
         System.out.println("Obim: " + O);
         System.out.println("Povrsina: " + P);
   }
}
```

ZADATAK – SABERI.JAVA

```
class Saberi
{
   /*      Sabiramo brojeve od 1 do n.          */
   public static void main (String args[])
   {
         int n = 65;
         int s = ((n * (n+1))/2);

         // Pisi na standardni izlaz.
         System.out.println("Suma prvih " + n + " brojeva
   je: " + s);
   }
}
```

ZADATAK – SKENER.JAVA

```
import java.util.Scanner;

public class Skener
{
    /*  Ucitavamo dva broja sa standardnog ulaza i ispisujemo
    njihov zbir.*/
    public static void main(String [] args)
    {
            Scanner skener = new Scanner(System.in);
            System.out.println("Unesite dva cela broja");
            int a = skener.nextInt();
            int b = skener.nextInt();
            int zbir = a + b;
            System.out.println("Zbir brojeva " + a + " i " + b + " je
    " + zbir);
    }
}
```