Visoka tehnička škola Niš

Studijski program:

Savremene računarske tehnologije

Internet programiranje

(9)

Interfejsi i polimorfizam u Javi

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

Decembar, 2018.

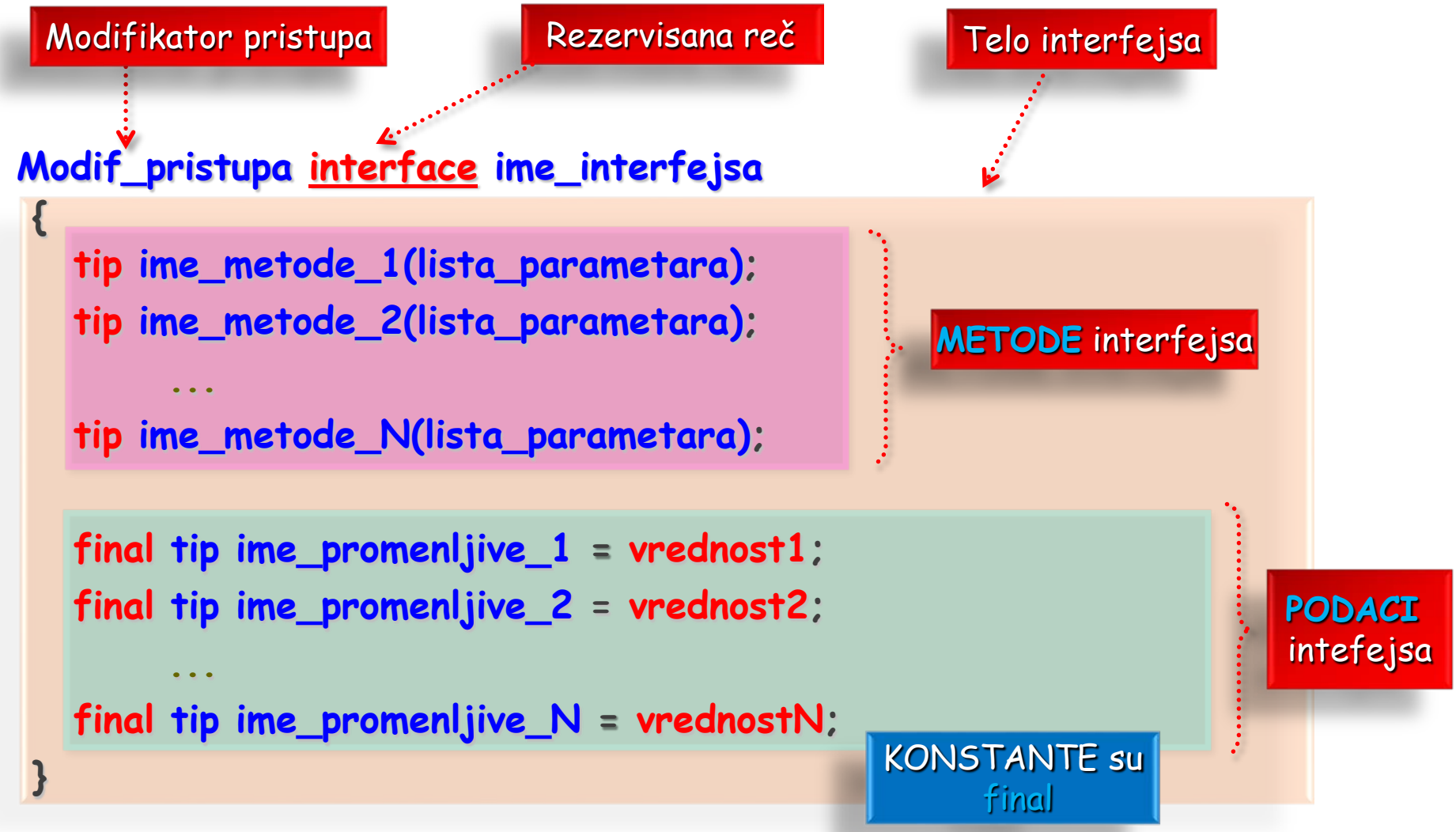
Interfejsi (1)

- Na predavanjima iz ovog predmeta već je više puta rečeno da **METODE** definišu **NAČIN PRISTUPA** podacima u klasama.
- Upotrebom rezervisane reči **interface** može se potpuno **ODVOJITI NAČIN PRISTUPA PODACIMA** (kaže se interfejs) od **same realizacije klase**!
- Rezervisanom rečju **interface** se zapravo zadaje **SKUP METODA** koje će **KASNIJE**, neka **klasa** (**jedna** ili **više** njih), realizovati - implementirati.
- Dozvoljeno je da se **INTERFEJS DEKLARIŠE**, a da se tom prilikom (u tom trenutku) **NE RAZMATRA** kako će on biti zaista implementiran.
- Prilikom **REALIZACIJE INTERFEJSA**, klasa **mora** da napravi **POTPUN SKUP METODA** koje su njime definisane.
- **INTEFEJSOM** se dakle definiše **ŠTA** metoda treba da radi, ali **NE I KAKO** to treba da se izvede!
- Međutim, svaka **KLASA slobodno odlučuje KAKO** će metode iz interfejsa biti realizovane.

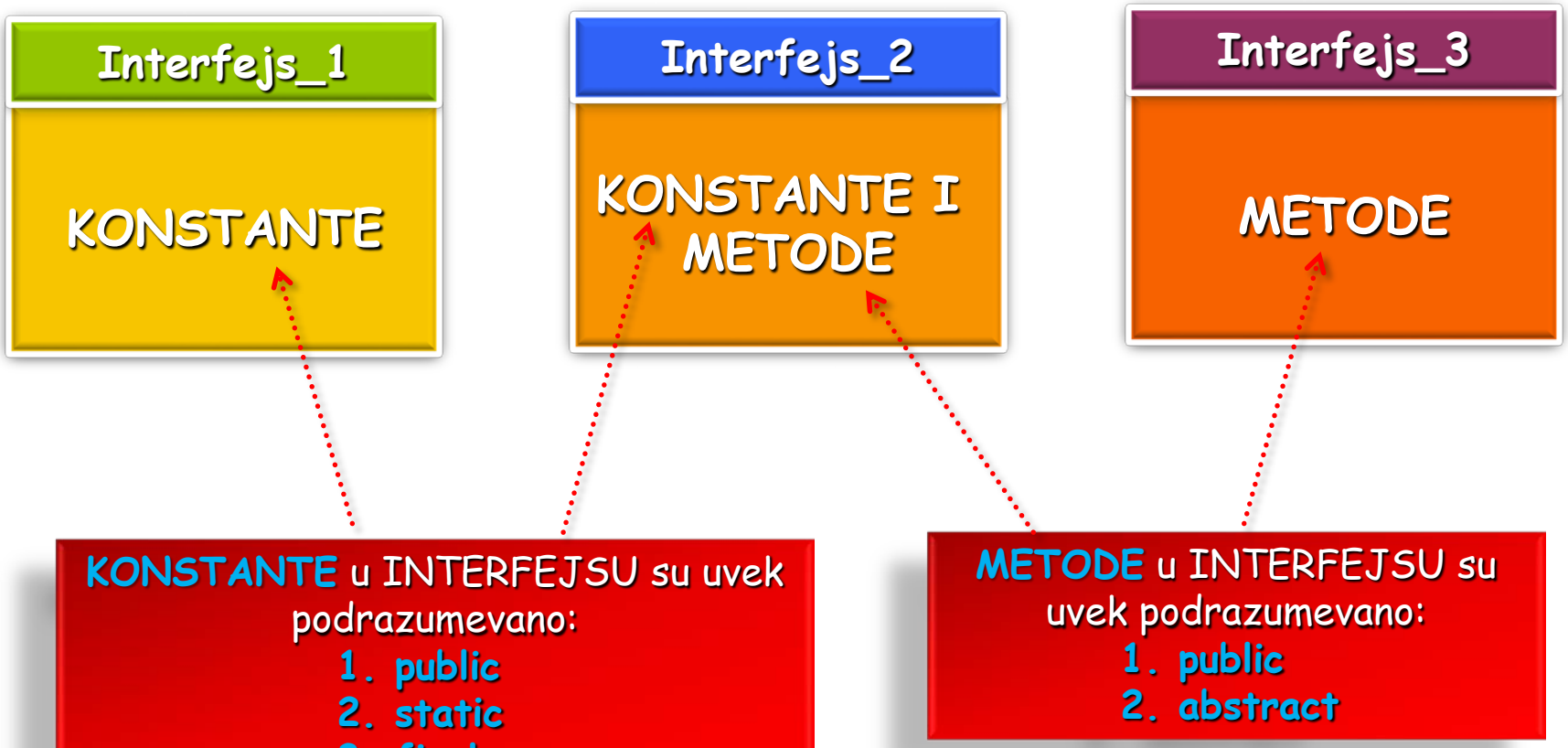
Interfejsi (2)

- Imajte na umu, dekleracija interfejsa **NE IMPLICIRA** nikakvu realizaciju.
- Standardno, da bi metoda iz jedne klase mogla da pozove metodu iz druge, obe klase **MORAJU POSTOJATI** u trenutku prevođenja zbog provere potpisa metoda – **INTERFEJSI** prevazilaze ovo ograničenje.
- **INTERFEJSI** podržavaju **DINAMIČKO RAZREŠAVANJE METODA** u trenutku izvršavanja!
- Čak i klase koje **NISU** u hijerarhijskom smislu nasleđivanja srodne, **MOGU REALIZOVATI ISTI INTEFEJS**.
- Veoma je značajno da se **INTERFEJSI MOGU PROŠIRIVATI** baš kao i klase!
- Na osnovu već izloženog, očigledno je da je **INTERFEJS** veoma **sličan APSTRAKTNOJ KLASI**.
- **RAZLIKA** je u tome što **klasa može** da realizuje **VIŠE od jednog interfejsa** !
- Sa druge strane, **klasa** može da nasledi **samo jednu NATKLASU**.

Dekleracija interfejsa



Podrazumevani pristupi

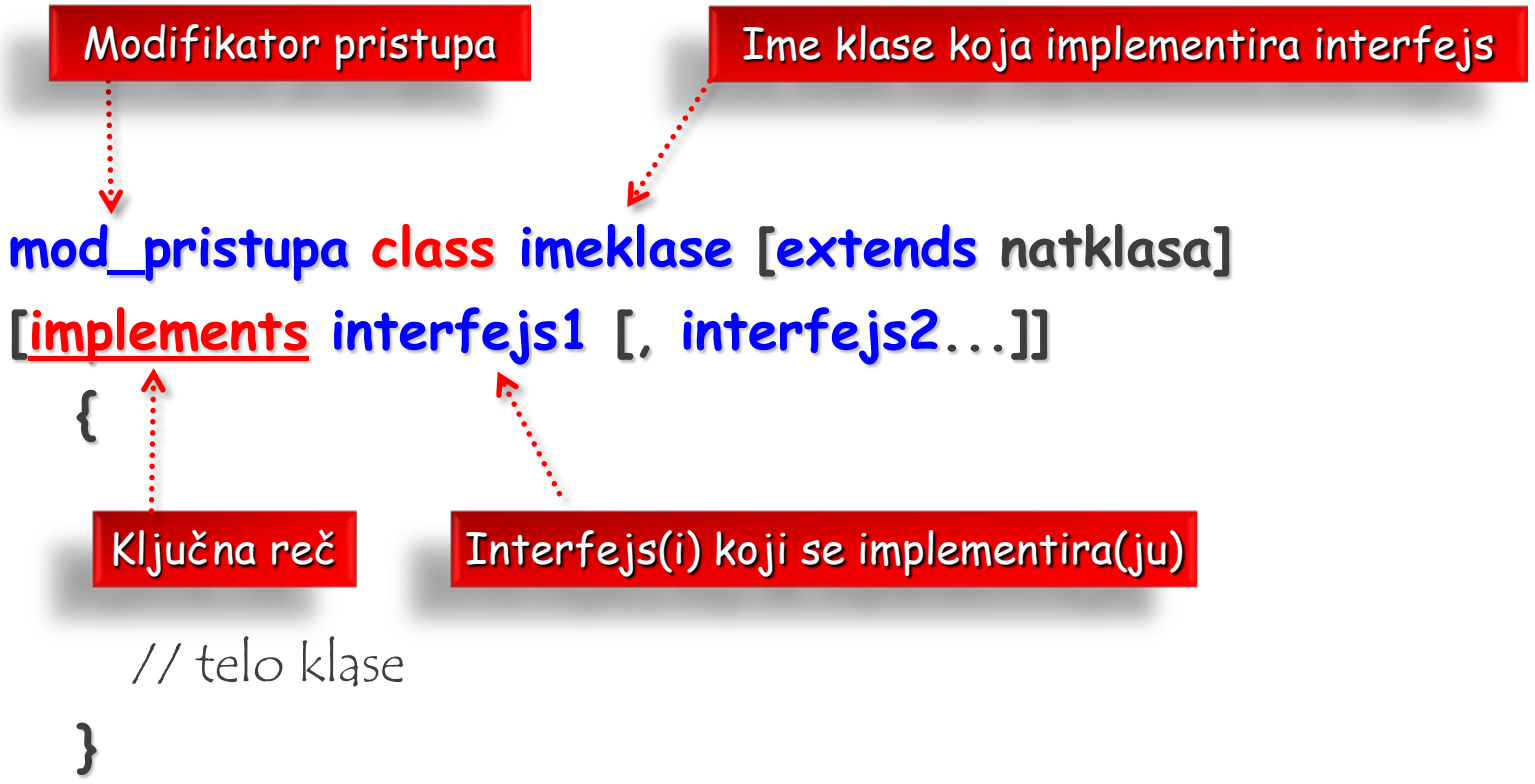


Implementacija interfejsa (1)

- Kada je interfejs **DEFINISAN**, **JEDNA** ili **VIŠE** klasa mogu da ga **REALIZUJU** (još se kaže i **IMPLEMENTIRAJU**).
- Ako klasa **realizuje neki interfejs**, u njenu **dekleraciju TREBA UKLJUČITI** odredbu **implements**.
- Ključna reč **implements** obaveštava kompajler o **odluci - želji** klase da **IMPLEMENTIRA neki od intefejsa**.
- Kada se kaže da Vam je neki **INTEFEJS POZNAT**, to zapravo znači da poznajete **SVE METODE** koje su podržane interfejsom.
- Osnovna **NAMENA INTERFEJSA** je da omogući **KREIRANJE KLASA** koje će posedovati unapred poznate metode!
- Kada poznajete **SVE METODE U KLASI** može se reći da poznajete tu klasu iako **NE ZNATE** detalje kako su one realizovane.

Implementacija interfejsa (2)

- Generalno, klase se mogu kreirati **IMPLEMENTACIJOM INTERFEJSA** na sledeći način:



Primer interfejsa (1)

```
// Dekleracije interfejsa Callback
interface Callback
  {
     void callback(int param);
   }
```

Definicija interfejsa **Callback** koji ima samo jednu metodu **callback()**

Realizacija (implementacija) interfejsa **Callback** u klasi **Client**

```
class Client implements Callback
 {
    // Implementiranje Callback interfejsa

    public void callback(int p)
      {
        System.out.println("callback pozvana sa " + p);
       }
 }
```

Realizacija metode definisane u interfejsu **Callback**

Primer interfejsa (2)

- Naravno, klase koje realizuju neki interfejs **mogu** definisati i **SVOJE SOPSTVENE ČLANOVE**.

Implementacija metode iz interfejsa

Implementacija interfejsa Callback

```
class Client implements Callback {
    public void callback(int p) {
     System.out.println("callback pozvana sa " + p);
  }

    void nonInterfaceMeth() {
     System.out.println("Klase koje implementiraju interfaces " +
                        "mogu takođe definisati druge članove.");
  }
}
```

Metoda nonInterfaceMeth() nije definisana interfejsom, može biti DODANA u procesu implementacije

Konstante u interfejsu

- Pomoću **interfejsa** mogu se uvesti **ZAJEDNIČKE KONSTANTE** u **VIŠE** klasa!
- Neophodno je **DEKLARISATI INTRFEJS sa promenljivama** koje su **INICIJALIZOVANE**.
- Prilikom realizacije interfejsa **sve navedene promenljive** postaju **konstante**:

```
interface deljeneKonstante {
    int NO = 0;
    int YES = 1;
    int MAYBE = 2;
    int LATER = 3;
    int SOON = 4;
    int NEVER = 5;
}
```

U intefejsu konstante **MORAJU** biti inicijalizovane (final)!

- Već je napomenuto, interfejs se **može NASLEDITI**!
- U Javi se za **NASLEĐIVANJE INTERFEJSA** koristi se rezervisana reč **extends** (dakle, isto kao i prilikom nasleđivanja klase).

Nasleđivanje interfejsa

```
interface A {
  void metod_1();
  void metod_2();
}
  interface B extends A {
  void metod_3();
}
  class MojaKlasa implements B {
    public void metod_1() {
    System.out.println("Implementiran metod_1().");
  }
   public void metod_2() {
    System.out.println("Implementiran metod_2().");
  }
   public void metod_3() {
    System.out.println("Implementira metod_3().");
   }
}
```

Dekleracija interfejsa A

Interfejs B je **nasledio** interfejs A i dodao (abstract) metodu metod3().

Ova klasa mora implementirati sve metode iz A i B.

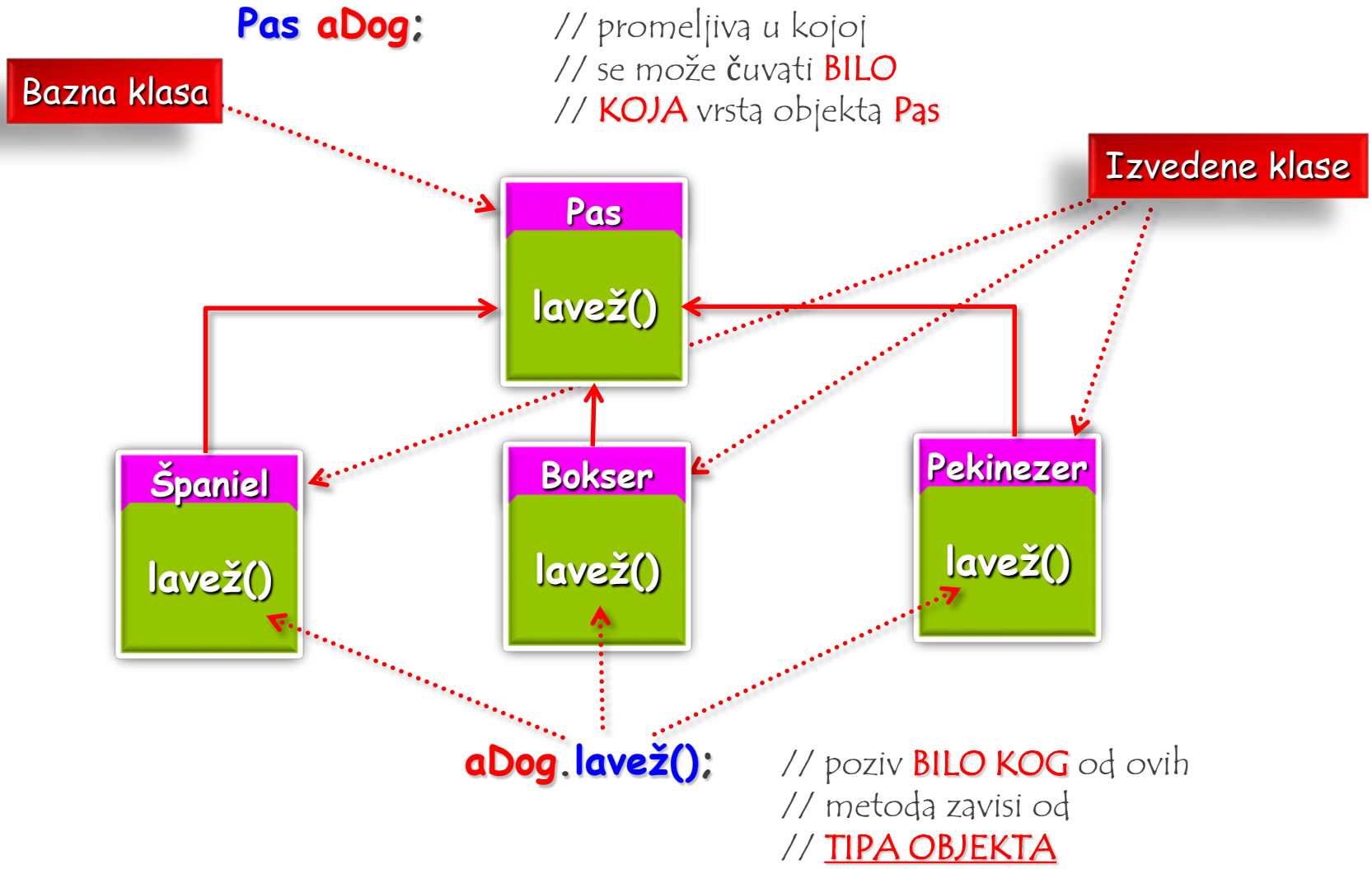
Imlementacija svih metoda: metod_1 do metod_3 iz intefejsa A i B

Primena interfejsa

Polimorfizam (1)

- Iz **C#** već znamo da pojam **POLIMORFIZMA** podrazumeva mogućnost da se jedna promenljiva **BILO KOG TIPA** može koristi za referenciranje objekata **RAZLIČITIH TIPOVA**.
- Takođe, polimorfizam podrazumeva i **POZIVANJE ONOG METODA** koji je karakterističan za **TIP OBJEKTA** koja ta promenljiva referencira.
- Zahvaljujući polimorfizmu, **ISTI POZIV METODA** može da se ponaša **DRUGAČIJE**, u **zavisnosti od TIPA OBJEKTA** na koji se primenjuje.
- Već znamo, polimorfizam funkcioniše **SAMO** sa objektima **IZVEDENE KLASE**.
- **REFERENCA NA OBJEKT IZVEDENE KLASE** može da se čuva u **promenljivoj tipa IZVEDENE KLASE**, ali i u promenljivoj tipa **BILO KOJE** direktne ili indirektne bazne klase.
- Polimorfizam se **ISKLJUČIVO** primenjuje na **METEODE**, a **NIKAKO** na **podatke članove**.

Polimorfizam (2)



Interfejsi i polimorfizam (1)

- Kod kuće verovatno imate televizor (TV), DVD plejer (VCR), HiFi ili slične uređaje koji poseduju **DALJINSKI UPRAVLJAČ** (**DU**).
- Na svim daljinskim upravljačima verovatno postoji **ZAJEDNIČKI PODSKUP DUGMADI** koje imaju **ISTE** (ili slične) funkcije.
- **Primer ovih funkcija** mogu biti:
  - **Uključivanje**,
  - **Isključivanje**,
  - **Gašenje tona**,
  - **Pojačavanje** i **Utišavanje**,
  - **Promena kanala naviše**,
  - **Promena kanala naliže**,
  - **mute funkcija**
  - **i slično**.

Interfejsi i polimorfizam (2)

- Da li se može napraviti **UNIVERZALNI DALJINSKI UPRAVLJAČ** koji se može **ADAPTIRATI** uređaju koji ga koristi?
- Sam za sebe **DU ne služi ničemu**, jer se njime definiše skup **standardnih funkcija**, ali se funkcija svakog dugmeta **mora programirati** za svaki uređaj **PONAOSOB**.
- **SKUP UREĐAJA** sa **DU** se može predstaviti **KLASAMA** (TV, VCR, HiFi, ...).
- **SVAKA KLASA** koristi **ISTI INTERFEJS DU** ali na sebi svojstven - **drugi način**.

Interfejsi i polimorfizam DU

```
public interface RemoteControl
{
    boolean powerOnOff();                    // Vraća novo stanje, on = true
    int volumeUp(int increment);             // Vraća novi nivo glasnoće
    int volumeDown(int decrement);           // Vraća novi nivo glasnoće
    void mute();                             // Obustavi zvučni izlaz
    int setChannel(int channel);             // Biranje i vraćanje br. kanala
    int channelUp();                         //  Vraćanje novog kanala
    int channelDown();                       // Vraćanje novog kanala
}
```

Definisanje INTERFEJSA RemoteControl

Definisanje metoda interfejsa RemoteControl

Interfejsi i polimorfizam TV (1)

```
import static java.lang.Math.max;
import static java.lang.Math.min;


public class TV implements RemoteControl {
  private String make = null;
  private int screensize = 0;
  private boolean power = false;
  private int MIN_VOLUME = 0;
  private int MAX_VOLUME = 100;

public TV(String make, int screensize)
  {
   this.make = make;
   this.screensize = screensize;
  }
```

Klasa TV Implementira interfejsa RemoteControl

Inicijalizovane promenljive (konstante) definisane u klasi TV

Konstrktor klase TV sa parametrima IME UREĐAJA i veličine ekrana

Interfejsi i polimorfizam TV (2)

Realizacija metode powerOnOff()

```
public boolean powerOnOff() {
        power = ! power;
        System.out.println(make + " "+ screensize + " inch TV power
                                                " + (power ? " on. " : " off. "));
    return power;
}
```

Realizacija metode mute()

Umesto prave funkcionalnosti ispisuje se zadatak na konzoli

```
public void mute() {
   if( ! power) {                    // Ako je isključen
   return;                           // ne radi ništa
  }
   volume = MIN_VOLUME;
     System.out.println(make + " " + screensize + " inch TV volume level:
                                                " + volume);
}
```

Interfejsi i polimorfizam TV (3)

```
public int volumeDown(int decrement) {
    if ( ! power) {          // Ako je uređaj isključen
    return 0;                // ne radi ništa
}                            // U suprotnom:
    volume -= decrement;
    volume = max(volume, MIN_VOLUME);
    System.out.println(make + " " + screensize + " inch TV volume level: "
                                                        + volume);
   return volume;
  }
 . . .
}
```

Realizacija metoda volumeDown()

Ne manja glasnoća od najmanje definisane

Interfejsi i polimorfizam VCR (1)

```
import static java.lang.Math.max;
import static java.lang.Math.min;


public class VCR implements RemoteControl {
  public VCR(String make) {
  this.make = make;
}


public boolean powerOnOff() {
  power = ! power;
  System.out.println(make + " VCR power " + (power ? "on." : "off."));
 return power;
}
```

Klasa VCR implementira interfejsa RemoteControl

Iste metode su implementirane za VCR uređaj!!!

Interfejsi i polimorfizam VCR (1)

```
public int volumeUp(int increment) {
    if(!power) {
 return 0;
}
 volume += increment;
 volume = min(volume, MAX_VOLUME);
 System.out.println(make + " VCR volume level: " + volume);
 return volume;
 }
 . . .
}
```

Iste metode su implementirane za VCR uređaj

Na sličan način se mogu kreirati klase koje realozuju Hi-Fi ili neki drugi uređaj

Polimorfizam na delu

```
import static java.lang.Math.random;

public class TryRemoteControl {
public static void main(String args[]) {
RemoteControl remote = null;
for(int i = 0 ; i<5 ; i++) {
    if(random() < 0.5)
        remote = new TV(random() < 0.5 ? "Sony" : "Hitachi",
                                          random() < 0.5 ? 32 : 28);
    else
        remote = new VCR(random()<0.5 ? "Panasonic" : "JVC");
  remote.powerOnOff();          // Prekidač je uključen
  remote.channelUp();           // Postavi sledeći kanal
  remote.volumeUp(10);          // Pojačaj glasnost na 10
} } }
```

Metoda iz random paketa

Drugi parametar u konstruktoru

Polimorfizam na delu – moguć izlaz

```
Sony 28 inch TV power on.
Sony 28 inch TV tuned to channel: 1
Sony 28 inch TV volume level: 10
Panasonic VCR power on.
Panasonic VCR tuned to channel: 1
Panasonic VCR volume level: 10
Sony 32 inch TV power on.
Sony 32 inch TV tuned to channel: 1
Sony 32 inch TV volume level: 10
JVC VCR power on.
JVC VCR tuned to channel: 1
JVC VCR volume level: 10
Sony 28 inch TV power on.
Sony 28 inch TV tuned to channel: 1
Sony 28 inch TV volume level: 10
```

Testiranje tipa objekta, instanceof (1)

- U nastavku je opisana primena **INTERFEJSA** koji omogućava **SORTIRANJE OBJEKATA** na **OBJEKTOM-SPECIFIČAN** način.
- Mnoge vrste objekata se mogu se **SORTIRATI**, tako se na primer imena studenata mogu sortirati po **abecedi** ili **azbuci**, dok se špil karata može sortirati po **boji** karata ili po **brojevima**.
- Jasno je da mehanizam, odnosno, **ALGORITAM SORTIRANJA** treba da bude sasvim **DRUGAČIJI** i zavisi od tipa objekata.
- Ako formiramo interfejs **Comparable**, on bi trebalo da omogući **UPOREĐIVANJE BILO KOJA DVA OBJEKATA** nastala implementacijom **interfejsa** te se kao takvi mogu komparirati a time i sortirati.
- U Javi, slično kao i C#-u (**is**, **as**), postoji ključna reč **instanceof** kojom se **TESTIRA TIP OBJEKTA**.
- Vraća se boolean promenljiva - **true** ako je objekt **ZADATOG** tipa ili se može **KONVERTOVATI** u zadati tip.

Testiranje tipa objekta, instanceof (2)

```
public class MainClass {
    public static void main(String[] args) {
        // Define some object:
        GameObject someObject = new Player();
        // Test if the first object is a GameObject.
        if(someObject instanceof GameObject)
            System.out.println("Object is a GameObject!");
        else
            System.out.println("Not a GameObject...");

       // Test if it is a Player.
        if(someObject instanceof Player)
            System.out.println("Object is a Player!");
        else
            System.out.println("Not a Player...");
        // Test if it is an NPC.
        if(someObject instanceof NPC)
            System.out.println("Object is a NPC!");
        else
            System.out.println("Not an NPC...");
    }
}
```

Kolekecije i interfejsi (1)*

```
public class Point implements Comparable {
  public double x, y;
  public Point(double x, double y) {
  this.x = x;
  this.y = y;
}

public void print() {
  System.out.println("X: " + x + " Y: " + y + " Mag: " + Math.sqrt(x*x+y*y));
}
```

Kreiranje klase Point implementacijom interfejsa Comparable

Konstruktor

print() je metoda klase Point

Kolekecije i interfejsi (2)*

```
public int compareTo(Object o) {
 if(! (o instanceof Point))
     return 0;
Point otherPoint = (Point) o;

Double thisAbsMag = Math.sqrt(x * x + y * y);
Double otherPointAbsMag = Math.sqrt(otherPoint.x * otherPoint.x +
                                    otherPoint.y * otherPoint.y);
return thisAbsMag.compareTo(otherPointAbsMag);

  //if(thisAbsMag > otherPointAbsMag) return 1;
  //if(thisAbsMag < otherPointAbsMag) return -1;
   //return 0;
  }
}
```

Testiranje: da li je objekt o tipa Point

Kastovanje objekta o - Cast operator

Ref. Tačka thisAbsMag

Poziv sa Duble.compareTo()

Računanje absolutne vrednosti otherPoint tačke od koordinatnog početka.

Kolekecije i interfejsi (3)*

```
import java.util.ArrayList;
import java.util.Collections;

public class MainClass {
public static void main(String[] args) {
    int numberOfPoints = 5;
    ArrayList<Point> points = new ArrayList<Point>();

  for(int i = 0; i < numberOfPoints; i++)
        points.add(new Point(Math.random() * 100, Math.random() * 100));
        System.out.println("Unsorted: ");
      for(int i = 0; i < numberOfPoints; i++)
          points.get(i).print();
```

Kolekecije i interfejsi (4)*

```
Collections.sort(points);
        System.out.println("Sorted: ");
       for(int i = 0; i < numberOfPoints; i++)
            points.get(i).print();

         points.sort(Collections.reverseOrder());
         System.out.println("Sorted in Reverse: ");

        for(int i = 0; i < numberOfPoints; i++)
          points.get(i).print();
   }
}
```