VEŽBA 12: OOP – INTERFEJSI

Primer 1 – Pojam i primena interfejsa

```java
package com.asss.uup;

/*
    Interfejs
    - pravi se kao i klasa s tim da se deklarise kljucnom recju "interface"
    - ne može imati svoje instance i sva polja su mu *public static final*,
      a svi metodi *public abstract* tako da nema potrebe navoditi modifikatore
    - najčešće se koristi za deklaraciju zajedničkih ponašanja
    - klase ga implementiraju pomoću ključne reči "implements"
    - prilikom imenovanja najčešće se koristi pridev, za razliku od klasa gde
      se najčešće koristi imenica (kao ovde)
    - interfejsi u potpunosti omogućavaju primenu polimorfizma
*/
public interface Kompjuter {

    // public static final
    String os = "Windows 10";

    // public abstract
    void ukljuci();

    void iskljuci();

}
```

```java
package com.asss.uup;

// implementacija interfejsa u klasu
public class Desktop implements Kompjuter {

    // implementacija metoda nasledjenih iz interfejsa
    @Override
    public void ukljuci() { System.out.println("Ukljuci kompjuter sa OS: " + os); }

    @Override
    public void iskljuci() { System.out.println("Iskljuci kompjuter sa OS: " + os); }

}
```

```java
package com.asss.uup;

public class Main {

    public static void main(String[] args) {

//        Desktop desktop = new Desktop();

        // primena polimorfizma za interfejs
        Kompjuter desktop = new Desktop();

        desktop.ukljuci();
        desktop.iskljuci();

    }
}
```

Primer 2 – Upotreba interfejsa u hijerarhiji klasa (apstraktnih i konkretnih)

```java
package com.asss.uup;

import com.asss.uup.interfejsi.Komponentnost;

/*
    apstraktna natklasa koja je najvisa u pisanom delu hijerarhije
    (iznad nje se nalazi samo klasa Object)
    i koja implementira interfejs Komponentnost
    (posto je klasa apstraktna ne mora da implementira definiciju
     u nasledjeni metod "prikaziKomponente()" iz interfejsa)
*/
public abstract class PametniUredjaj implements Komponentnost {

    private String procesor;
    private String graficka;
    private int ram;

    protected PametniUredjaj(String procesor, String graficka, int ram) {
        this.procesor = procesor;
        this.graficka = graficka;
        this.ram = ram;
    }
}
```

```java
package com.asss.uup.komp;

import com.asss.uup.PametniUredjaj;

/*
    apstraktna klasa koja prosiruje apstraktnu (nat)klasu PametniUredjaj
    i koja pored clanova definisanih u klasi PametniUredjaj nasledjuje i
    implementirane clanove interfejsa Komponentnost
    (imajuci u vidu da je i ovde rec o apstraktnoj klasi, nije potrebno
     pisati definiciju nasledjenim metodima interfejsa Komponentnost)
*/
public abstract class Kompjuter extends PametniUredjaj {

    private int brojUsbPortova;
    private boolean utpPort;

    protected Kompjuter(String procesor, String graficka, int ram, int brojUsbPortova, boolean utpPort) {
        super(procesor, graficka, ram);
        this.brojUsbPortova = brojUsbPortova;
        this.utpPort = utpPort;
    }
}
```

```java
package com.asss.uup.komp;

import com.asss.uup.interfejsi.Visedelan;

/*
    konkretna klasa koja predstavlja jednu od konkrentih potklasa apstraktne (nat)klase Kompjuter
    i u njoj se mora napisati definicija svih nasledjenih apstraktnih metoda kako bi klasa mogla
    da ostane konkretna, u suprotnom se mora oznaciti kao apstraktna
    (implementacija interfejsa Visedelan je prvobitni oblik klase koji u ovom slucaju nije
     neophodan, jer je na kraju primera dodata implementacija interfejsa Komponentnost u klasu
     PametniUredjaj koja predstavlja indirektnu natklasu ove klase)
*/
public class Desktop extends Kompjuter implements Visedelan {

    public Desktop(String procesor, String graficka, int ram, int brojUsbPortova, boolean utpPort) {
        super(procesor, graficka, ram, brojUsbPortova, utpPort);
    }

    @Override
    public void prikaziKomponente() {
        System.out.println("Kuciste sa svojim komponentama, monitor, mis, tastatura, ozvucenje.");
    }
}
```

```java
package com.asss.uup.komp;

import ...

/*
    konkretna klasa koja predstavlja jednu od konkrentih potklasa apstraktne (nat)klase Kompjuter
    i u njoj se mora napisati definicija svih nasledjenih apstraktnih metoda kako bi klasa mogla
    da ostane konkretna, u suprotnom se mora oznaciti kao apstraktna
    (implementacija interfejsa Jednodelan je prvobitni oblik klase koji u ovom slucaju nije
     neophodan, jer je na kraju primera dodata implementacija interfejsa Komponentnost u klasu
     PametniUredjaj koja predstavlja indirektnu natklasu ove klase)
    i koja jos implementira i interfejs Prenosiv ciji metod "nacinNosenja()" i redefinise
*/
public class Laptop extends Kompjuter implements Prenosiv, Jednodelan {

    public Laptop(String procesor, String graficka, int ram, int brojUsbPortova, boolean utpPort) {
        super(procesor, graficka, ram, brojUsbPortova, utpPort);
    }

    @Override
    public void nacinNosenja() { System.out.println("Laptop se moze nositi u torbi ili rancu."); }

    @Override
    public void prikaziKomponente() {
        System.out.println("Kuciste sa integrisanim komponentama i panelom sa tastaturom i touchpad-om," +
                           " na koje je zakacen monitor.");
    }
}
```

```java
package com.asss.uup.tel;

import ...

/*
    apstraktna klasa koja prosiruje apstraktnu (nat)klasu PametniUredjaj, koja pored clanova
    definisanih u klasi PametniUredjaj nasledjuje i implementirane clanove interfejsa Komponentnost
    (implementacija interfejsa Jednodelan je prvobitni oblik klase koji u ovom slucaju nije neophodan,
     jer je na kraju primera dodata implementacijainterfejsa Komponentnost u klasu PametniUredjaj koja
     predstavlja direktnu natklasu ove klase) i koja jos implementira i interfejs Prenosiv
    (iako ne postoji potreba da se u klasi redefinise nijedan nasledjeni apstraktni metod, isti se
     mogu redefinisati i kao takvi bice nasledjeni u sve potklase, konkretne ili apstraktne)
*/
public abstract class MobilniTelefon extends PametniUredjaj implements Prenosiv, Jednodelan {

    private int brojKamera;
    private boolean senzorBlizine;

    protected MobilniTelefon(String procesor, String graficka, int ram, int brojKamera, boolean senzorBlizine) {
        super(procesor, graficka, ram);
        this.brojKamera = brojKamera;
        this.senzorBlizine = senzorBlizine;
    }

    // definisanje apstraktnog metoda nasledjenog iz interfejsa Jednodelan (po prvobitnoj varijanti)
    @Override
    public void prikaziKomponente() {
        System.out.println("Kuciste sa integrisanim komponentama preko kojih je postavljen touch screen.");
    }

}
```

```java
package com.asss.uup.tel;

/*
    konkretna klasa koja predstavlja jednu od konkrentih potklasa apstraktne (nat)klase MobilniTelefon
    i u njoj se mora napisati definicija svih nasledjenih apstraktnih metoda kako bi klasa mogla da
    ostane konkretna, u suprotnom se mora oznaciti kao apstraktna
    (imajuci u vidu da su kroz hijerarhiju nasledjeni i metod "prikaziKomponente()" i metod
     "nacinNosenja", kao i to da je prvi vec definisan u natklasi, potrebno je samo jos definisati i
     drugi - "nacinNosenja")
*/
public class Telefon extends MobilniTelefon {

    public Telefon(String procesor, String graficka, int ram, int brojKamera, boolean senzorBlizine) {
        super(procesor, graficka, ram, brojKamera, senzorBlizine);
    }

    @Override
    public void nacinNosenja() { System.out.println("Telefon se moze nositi u dzepu."); }

}
```

```java
package com.asss.uup.tel;

/*
    konkretna klasa koja predstavlja jednu od konkrentih potklasa apstraktne (nat)klase MobilniTelefon
    i u njoj se mora napisati definicija svih nasledjenih apstraktnih metoda kako bi klasa mogla da
    ostane konkretna, u suprotnom se mora oznaciti kao apstraktna
    (imajuci u vidu da su kroz hijerarhiju nasledjeni i metod "prikaziKomponente()" i metod
     "nacinNosenja", kao i to da je prvi vec definisan u natklasi, potrebno je samo jos definisati i
     drugi - "nacinNosenja")
*/
public class Tablet extends MobilniTelefon {

    public Tablet(String procesor, String graficka, int ram, int brojKamera, boolean senzorBlizine) {
        super(procesor, graficka, ram, brojKamera, senzorBlizine);
    }

    @Override
    public void nacinNosenja() { System.out.println("Tablet se moze nositi u futroli."); }
}
```

```java
package com.asss.uup.interfejsi;

/*
    samostalni interfejs
    (ne prosiruje ni jedan drugi)
*/

public interface Prenosiv {

    void nacinNosenja();

}
```

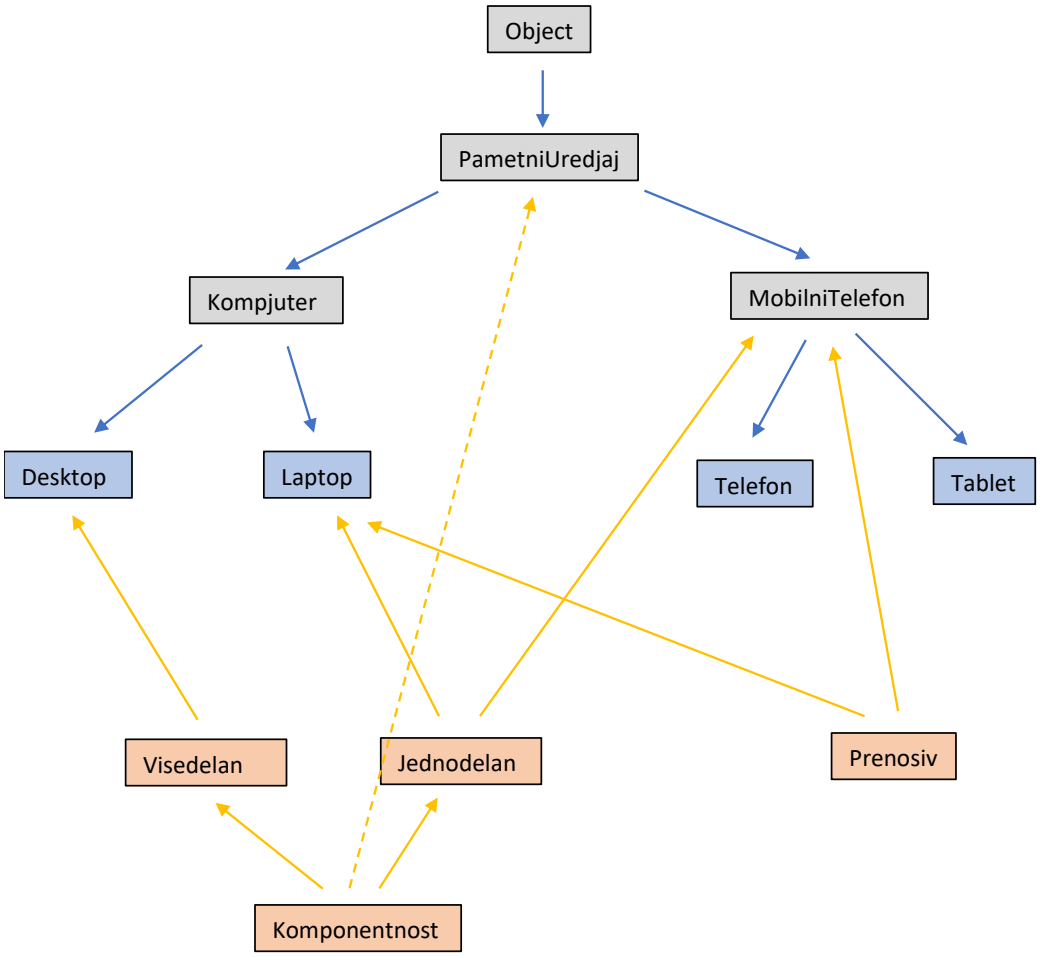
```java
package com.asss.uup.interfejsi;

// interfejs koji prosiruju drugi interfejsi
public interface Komponentnost {

    void prikaziKomponente();

}
```

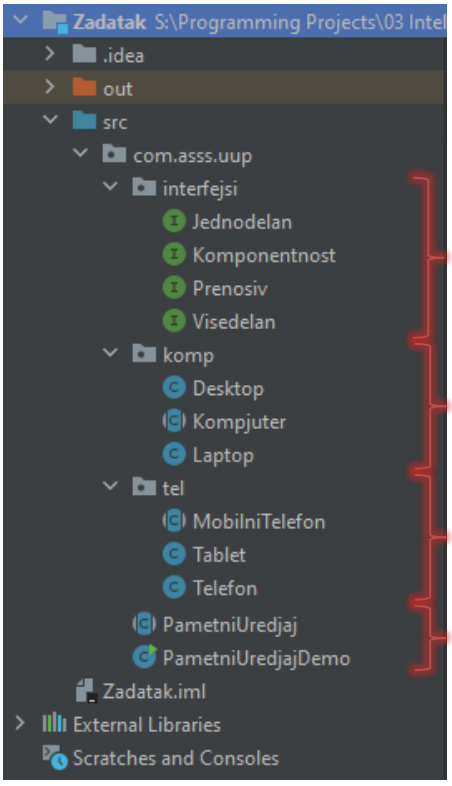
```java
package com.asss.uup.interfejsi;

/*
    interfejs moze naslediti/prosiriti drugi interfejs
    i time nasledjuje sve njegove clanove
*/
public interface Jednodelan extends Komponentnost {
}
```

```java
package com.asss.uup.interfejsi;

/*
    interfejs moze naslediti/prosiriti drugi interfejs
    i time nasledjuje sve njegove clanove
*/
public interface Visedelan extends Komponentnost {
}
```

```java
package com.asss.uup;

import ...

public class PametniUredjajDemo {

    public static void main(String[] args) {

        /*
            Svaki od napravljenih objekata se moze cuvati u promenljivoj tipa natklase u cijoj se hijerarhiji nalazi,
            kao i tipa interfejsa koji implementira direktno ili indirektno
        */

        /*
            tipovi natklasa:                            tipovi interfejsa:
            Object, PametniUredjaj, Kompjuter           Komponentnost, Visedelan
        */
        Desktop desktop = new Desktop( procesor: "Intel",  graficka: "NVidia",  ram: 64,  brojUsbPortova: 9,  utpPort: true);
        /*
            tipovi natklasa:                            tipovi interfejsa:
            Object, PametniUredjaj, Kompjuter           Komponentnost, Jednodelan, Prenosiv
        */
        Laptop laptop = new Laptop( procesor: "Intel",  graficka: "NVidia",  ram: 32,  brojUsbPortova: 3,  utpPort: true);

        /*
            tipovi natklasa:                            tipovi interfejsa:
            Object, PametniUredjaj, MobilniTelefon      Komponentnost, Jednodelan, Prenosiv
        */
        Telefon telefon = new Telefon( procesor: "Snapdragon",  graficka: "Adreno",  ram: 6,  brojKamera: 5,  senzorBlizine: true);
        /*
            tipovi natklasa:                            tipovi interfejsa:
            Object, PametniUredjaj, MobilniTelefon      Komponentnost, Jednodelan, Prenosiv
        */
        Tablet tablet = new Tablet( procesor: "Snapdragon",  graficka: "Adreno",  ram: 8,  brojKamera: 2,  senzorBlizine: true);

        // promenom tipa promenljive u kojoj se objekti cuvaju dolazice do pojave gresaka u pozivima metoda
        desktop.prikaziKomponente();
        System.out.println();

        laptop.prikaziKomponente();
        laptop.nacinNosenja();
        System.out.println();

        telefon.prikaziKomponente();
        telefon.nacinNosenja();
        System.out.println();

        tablet.prikaziKomponente();
        tablet.nacinNosenja();
        System.out.println();
    }
}
```

Dijagram primera





Paket sa interfejsima

Paket sa klasama hijerarhije Kompjuter

Paket sa klasama hijerarhije MobilniTelefon

Apstraktna natklasa i demo klasa

Primer 3 – Tipovi interfejsa

```
/*
    "Obican" interfejs
    je interfejs koji ima
    proizvoljan broj polja
    i vise od jednog metoda
*/
interface ObicanInterfejs {

    // polja i metodi

}
```

```
/*
    Marker interfejs
    je interfejs koji nema
    clanove i sluzi kompajleru
    kao smernica na koji nacin
    da posmatra objekte klase
    koja ga implementira
*/
interface MarkerInterfejs {

}
```

```
/*
    Funkcionalni interfejs
    je onaj interfejs koji u sebi
    ima deklaraciju proizvoljnog
    broja polja i samo jedan metod,
    a karakteristicni su po tome
    sto omogucavaju pisanje lambda
    izraza koji pripadaju obliku
    programiranja koji se zove
    funkcionalno programiranje
*/
interface FunkcionalniInterfejs {

    // polja i samo jedan metod

}
```