VEŽBA 11: OOP – POLIMORFIZAM

Primer 1 – Čuvanje objekta tipa potklase u promenljivoj tipa natklase

```java
package com.asss.uup;

// natklasa
public class BMW {

    // polja
    private String tipVozila;
    private String model;
    private int snaga;
    private int zapremina;

    // konstruktor
    protected BMW(String model, int snaga, int zapremina) {
        this.model = model;
        this.snaga = snaga;
        this.zapremina = zapremina;
    }

    // pristupni i metodi mutatori
    public String getTipVozila() { return tipVozila; }

    public void setTipVozila(String tipVozila) { this.tipVozila = tipVozila; }

    public String getModel() { return model; }

    public void setModel(String model) { this.model = model; }

    public int getSnaga() { return snaga; }

    public void setSnaga(int snaga) { this.snaga = snaga; }

    public int getZapremina() { return zapremina; }

    public void setZapremina(int zapremina) { this.zapremina = zapremina; }

    // metod za prikaz
    @Override
    public String toString() {
        return "BMW (" + getTipVozila() + "): " + getModel() + " (" + getSnaga() + "/" + getZapremina() + ")";
    }
}
```

```java
package com.asss.uup;

// potklasa klase BMW
public class Automobil extends BMW {

    // konstruktor
    protected Automobil(String model, int snaga, int zapremina) {
        super(model, snaga, zapremina);
        this.setTipVozila("automobil");
    }

    // redefinisani metod toString()
    @Override
    public String toString() {
        return "AUTOMOBIL\n- model: " + getModel() + "\n- snaga: " + getSnaga() + "\n- zapremina: " + getZapremina();
    }
}
```

```java
package com.asss.uup;

// potklasa klase BMW
public class Motor extends BMW {

    // konstruktor
    protected Motor(String model, int snaga, int zapremina) {
        super(model, snaga, zapremina);
        this.setTipVozila("motor");
    }

    // redefinisani metod toString()
    @Override
    public String toString() {
        return "MOTOR\n- model: " + getModel() + "\n- snaga: " + getSnaga() + "\n- zapremina: " + getZapremina();
    }
}
```

```java
package com.asss.uup;

public class Main {

    public static void main(String[] args) {

//        Automobil automobil = new Automobil("M5 Competition", 625, 4395);
//        Motor motor = new Motor("M1000RR", 212, 999);

        // POLIMORFIZAM - cuvanje objekta tipa potklase u promenljivoj tipa natklase
        BMW automobil = new Automobil( model: "M5 Competition",  snaga: 625,  zapremina: 4395);
        BMW motor = new Motor( model: "M1000RR",  snaga: 212,  zapremina: 999);

        System.out.println(automobil);

        System.out.println();

        System.out.println(motor);

    }
}
```

Primer 2 – Polimorfno ponašanje metoda

```java
package com.asss.uup;

/*
    APSTRAKTNA (nat)KLASA
    klasa koja ne moze imati svoje objekte
    (nije moguce instancirati je)
*/
public abstract class Oblik {

    /*
        apstraktni metod
        metod koji nema definiciju i
        klasa koja ga sadrzi takodje
        mora biti apstraktna
    */
    public abstract void crtaj();

    public abstract void povrsina();

}
```

```java
package com.asss.uup;

// konkretna (pot)klasa
public class Krug extends Oblik {

    private int r;

    public Krug(int r) { this.r = r; }

    /*
        konkretna implementacija
        ((re)definisan nasledjeni apstraktni metod)
    */
    @Override
    public void crtaj() { System.out.println("Crtam krug!"); }

    @Override
    public void povrsina() { System.out.println("Povrsina: " + r * r * Math.PI); }

}
```

```java
package com.asss.uup;

// konkretna (pot)klasa
public class Pravougaonik extends Oblik {

    private int a, b;

    public Pravougaonik(int a, int b) {
        this.a = a;
        this.b = b;
    }

    /*
        konkretna implementacija
        ((re)definisan nasledjeni apstraktni metod)
    */
    @Override
    public void crtaj() { System.out.println("Crtam pravougaonik!"); }

    @Override
    public void povrsina() { System.out.println("Povrsina: " + a * b); }

}
```

```java
package com.asss.uup;

// konkretna (pot)klasa
public class Trougao extends Oblik {

    private int c, h_c;

    public Trougao(int c, int h_c) {
        this.c = c;
        this.h_c = h_c;
    }

    /*
        konkretna implementacija
        ((re)definisan nasledjeni apstraktni metod)
    */
    @Override
    public void crtaj() { System.out.println("Crtam trougao!"); }

    @Override
    public void povrsina() { System.out.println("Povrsina: " + c * h_c / 2); }

}
```

```java
package com.asss.uup;

public class Main {

    public static void main(String[] args) {

        Krug krug = new Krug( r: 7);
        Pravougaonik pravougaonik = new Pravougaonik( a: 9,  b: 12);
        Trougao trougao = new Trougao( c: 27,  h_c: 28);

        krug.crtaj();
        pravougaonik.crtaj();
        trougao.crtaj();

        System.out.println();

        krug.povrsina();
        pravougaonik.povrsina();
        trougao.povrsina();

        System.out.println("\n--------------------------\n");

        nacrtajOblik(krug);
        nacrtajOblik(pravougaonik);
        nacrtajOblik(trougao);

        System.out.println();

        izracunajPovrsinuOblika(krug);
        izracunajPovrsinuOblika(pravougaonik);
        izracunajPovrsinuOblika(trougao);

    }

    /*
        polimorfno ponasanje metoda
        (metodu se predaje objekat tipa Oblik (u kojem se cuva
         objekat neke od njegovih konkrentih implementacija) ili
         objekta tipa neke od njegovih potklasa)
    */
    private static void nacrtajOblik(Oblik oblik) { oblik.crtaj(); }

    private static void izracunajPovrsinuOblika(Oblik oblik) { oblik.povrsina(); }

}
```

Primer 3 – Direkno i indirektno nasledjivanje i polimorfizam i polimorfno ponašanje metoda u funkciji istog

```java
package com.asss.uup;

/*
    apstraktna natklasa iz koje su izvedene dve apstraktne poklase,
    koje dalje prosiruju jos po dve konkretne (pot)klase
*/
public abstract class Objekat {

    /*
        polje koja ce naslediti sve potklase klase Objekat
        (private polje se nasledjuje, ali mu se ne moze
         pristupiti direktno pristupiti)
    */
    private int brojProstorija;

    protected Objekat(int brojProstorija) {
        this.brojProstorija = brojProstorija;
    }

    /*
        apstraktni metod koji ce se kao i ostali clanovi
        nasledjivati niz hijerarhiju klasa, s tim da sve
        klase koje ga ne (re)definisu treba oznaciti kao
        apstraktne - kljucnom recju "abstract"
    */
    protected abstract void prikaziStrukturuProstorija();

    public int getBrojProstorija() {
        return brojProstorija;
    }

    public void setBrojProstorija(int brojProstorija) {
        this.brojProstorija = brojProstorija;
    }

    @Override
    public String toString() {
        return "Objekat {brojProstorija = " + brojProstorija + '}';
    }
}
```

```java
package com.asss.uup;

/*
    imajuci u vidu da nasledjeni apstraktni metod nije (re)definisan
    klasa mora da bude oznacena kao apstraktna
*/
public abstract class StambeniObjekat extends Objekat {

    /*
        novo polje
        ovom polju se moze pristupiti samo u domenu ove klase,
        tj. prisutno je u njoj i njenim potklasama
        (deo hijerarhije klasa u kojoj je ova klasa na vrhu)
    */
    private String podrucje;

    public StambeniObjekat(int brojProstorija, String podrucje) {
        super(brojProstorija);
        this.podrucje = podrucje;
    }

    /*
        novi metod
        ovom metodu se moze pristupiti samo u domenu ove klase,
        tj. prisutan je u njoj i njenim potklasama
        (deo hijerarhije klasa u kojoj je ova klasa na vrhu)
    */
    public String getPodrucje() { return "Podrucje u kom se objekat nalazi: " + podrucje; }

    /*
        redefinisanje nasledjenog metoda
        (na samom pocetku vrednosti koja se vraca je pozvan
         istoimeni metod natklase pomocu kljucne reci "super")
    */
    @Override
    public String toString() {
        return super.toString() + "\nStambeniObjekat {podrucje = " + podrucje + '}';
    }
}
```

```java
package com.asss.uup;

public class Kuca extends StambeniObjekat {

    // novo polje
    private int povrsinaDvorista;

    public Kuca(int brojProstorija, String podrucje, int povrsinaDvorista) {
        super(brojProstorija, podrucje);
        this.povrsinaDvorista = povrsinaDvorista;
    }

    // konkretna implementacija metoda nasledjenog iz indirektne natklase Objekat
    @Override
    protected void prikaziStrukturuProstorija() {
        System.out.println("Dnevna soba, Spavaca soba, Kuhinja, Trpezarija, Podrum, Potkrovlje, Kupatilo, Ostava");
    }

    /*
        redefinisanje nasledjenog metoda
        (na samom pocetku vrednosti koja se vraca je pozvan istoimeni metod natklase pomocu kljucne reci "super")
    */
    @Override
    public String toString() { return super.toString() + "\nKuca {povrsinaDvorista = " + povrsinaDvorista + '}'; }
}
```

```java
package com.asss.uup;

public class Zgrada extends StambeniObjekat {

    // novo polje
    private int brojSpratova;

    public Zgrada(int brojProstorija, String podrucje, int brojSpratova) {
        super(brojProstorija, podrucje);
        this.brojSpratova = brojSpratova;
    }

    // konkretna implementacija metoda nasledjenog iz indirektne natklase Objekat
    @Override
    protected void prikaziStrukturuProstorija() {
        System.out.println("Dnevna soba kuhinjom i trpezarijom, Spavaca soba, Kupatilo");
    }

    /*
       redefinisanje nasledjenog metoda
       (na samom pocetku vrednosti koja se vraca je pozvan istoimeni metod natklase pomocu kljucne reci "super")
    */
    @Override
    public String toString() { return super.toString() + "\nZgrada {brojSpratova = " + brojSpratova + '}'; }
}
```

```java
package com.asss.uup;

/*
    imajuci u vidu da nasledjeni apstraktni metod nije (re)definisan
    klasa mora da bude oznacena kao apstraktna
*/
public abstract class UgostiteljskiObjekat extends Objekat {

    /*
        novo polje
        ovom polju se moze pristupiti samo u domenu ove klase,
        tj. prisutno je u njoj i njenim potklasama
        (deo hijerarhije klasa u kojoj je ova klasa na vrhu)
    */
    private String naziv;

    public UgostiteljskiObjekat(int brojProstorija, String naziv) {
        super(brojProstorija);
        this.naziv = naziv;
    }

    /*
        novi metod
        ovom metodu se moze pristupiti samo u domenu ove klase,
        tj. prisutan je u njoj i njenim potklasama
        (deo hijerarhije klasa u kojoj je ova klasa na vrhu)
    */
    public String getNaziv() { return "Ugostiteljski objekat: " + naziv; }

    /*
        redefinisanje nasledjenog metoda
        (na samom pocetku vrednosti koja se vraca je pozvan
         istoimeni metod natklase pomocu kljucne reci "super")
    */
    @Override
    public String toString() { return super.toString() + "\nUgostiteljskiObjekat {naziv = " + naziv + '}'; }
}
```

```java
package com.asss.uup;

public class Hotel extends UgostiteljskiObjekat {

    // novo polje
    private int brojZvezdica;

    public Hotel(int brojProstorija, String naziv, int brojZvezdica) {
        super(brojProstorija, naziv);
        this.brojZvezdica = brojZvezdica;
    }

    // konkretna implementacija metoda nasledjenog iz indirektne natklase Objekat
    @Override
    protected void prikaziStrukturuProstorija() {
        System.out.println("Hol, Recepcija, Apartmani, Restoran za rucavanje");
    }

    /*
       redefinisanje nasledjenog metoda
       (na samom pocetku vrednosti koja se vraca je pozvan istoimeni metod natklase pomocu kljucne reci "super")
    */
    @Override
    public String toString() { return super.toString() + "\nHotel {brojZvezdica = " + brojZvezdica + '}'; }
}
```

```java
package com.asss.uup;

public class Restoran extends UgostiteljskiObjekat {

    // novo polje
    private int brojSala;

    public Restoran(int brojProstorija, String naziv, int brojSala) {
        super(brojProstorija, naziv);
        this.brojSala = brojSala;
    }

    // konkretna implementacija metoda nasledjenog iz indirektne natklase Objekat
    @Override
    protected void prikaziStrukturuProstorija() {
        System.out.println("Lobi, Sale, Kuhinja");
    }

    /*
       redefinisanje nasledjenog metoda
       (na samom pocetku vrednosti koja se vraca je pozvan istoimeni metod natklase pomocu kljucne reci "super")
    */
    @Override
    public String toString() { return super.toString() + "\nRestoran {brojSala = " + brojSala + '}'; }
}
```

```java
package com.asss.uup;

public class Main {

    public static void main(String[] args) {

//        Kuca kuca = new Kuca(7, "selo", 15);
//        Zgrada zgrada = new Zgrada(3, "grad", 5);
//
//        Hotel hotel = new Hotel(2, "Sumadija", 3);
//        Restoran restoran = new Restoran(2, "Domacin", 3);

        StambeniObjekat kuca = new Kuca( brojProstorija: 7,  podrucje: "selo",  povrsinaDvorista: 15);
        StambeniObjekat zgrada = new Zgrada( brojProstorija: 3,  podrucje: "grad",  brojSpratova: 5);

        UgostiteljskiObjekat hotel = new Hotel( brojProstorija: 2,  naziv: "Sumadija",  brojZvezdica: 3);
        UgostiteljskiObjekat restoran = new Restoran( brojProstorija: 2,  naziv: "Domacin",  brojSala: 3);

//        Objekat kuca = new Kuca(7, "selo", 15);
//        Objekat zgrada = new Zgrada(3, "grad", 5);
//
//        Objekat hotel = new Hotel(2, "Sumadija", 3);
//        Objekat restoran = new Restoran(2, "Domacin", 3);

//        Object kuca = new Kuca(7, "selo", 15);
//        Object zgrada = new Zgrada(3, "grad", 5);
//
//        Object hotel = new Hotel(2, "Sumadija", 3);
//        Object restoran = new Restoran(2, "Domacin", 3);
```

```
        /*
            U zavisnosti od tipa promenljive u kojoj se cuvaju objekti
            konkretnih potklasa metodi ce se izvrsavati ili nece.
        */

        // pozivi metoda krajnjih potklasa (konkretnih klasa)
        kuca.prikaziStrukturuProstorija();
        System.out.println(kuca + "\n");

        zgrada.prikaziStrukturuProstorija();
        System.out.println(zgrada + "\n");

        hotel.prikaziStrukturuProstorija();
        System.out.println(hotel + "\n");

        restoran.prikaziStrukturuProstorija();
        System.out.println(restoran + "\n");

        System.out.println("\n-----------------------------------------\n");

        // pozivi metoda apstraktnih potklasa
        kuca.getPodrucje();
        zgrada.getPodrucje();

        hotel.getNaziv();
        restoran.getNaziv();

        System.out.println("\n-----------------------------------------\n");

        // poziv metoda definisanih u apstraktnoj natklasi
        kuca.getBrojProstorija();
        zgrada.getBrojProstorija();

        hotel.getBrojProstorija();
        restoran.getBrojProstorija();

    }
}
```