SLAGANJE

Slaganje

- **Slaganje** takođe omogućava kreiranje nove klase od postojećih, ali ne nasleđivanjem, već njihovim “slaganjem” – “složeni” objekat sadrži “prostije” objekte, relacija klasa je **ima** (engl. *has-a*) – npr. **vozilo ima motor**
- Slaganje je jednostavnije i fleksibilnije od nasleđivanja
- Vrste slaganja: **asocijacija**, **agregacija** i **kompozicija**

Slaganje

Vrste slaganja:

- **Asocijacija** (engl. association) – semantički slaba veza – primer osoba-vozilo, doktor-pacijent
- **Agregacija** (engl. aggregation) – specijalni vid unidirekcione asocijacije – tipično odnos celina/komponenta, primer vozilo-motor
- **Kompozicija** (engl. composition) – specijalni vid “jake” agregacije – primer kuća-sobe

	Asocijacija	**Agregacija**	**Kompozicija**
Vlasnik	Nema vlasnika	Jedan vlasnik	Jedan vlasnik
Životni vek	Poseban vek	Poseban vek	Vek vlasnika

Slaganje – primer 2.5 Vozilo i Motor

- Klasa Vozilo i klasa Motor

```
public class Motor {
    private boolean radi;
    private int tip;
    private int snaga;
    private int kubikaza;

    public void ukljuci(){
        ...
    }
    ...
```

```
public class Vozilo {
    protected String marka;
    protected String tip;
    protected Motor pogAgregat;
    protected String registracija;
    protected int godiste;
    protected Osoba vlasnik;

    ...
```

- Agregacija Vozilo-Motor, asocijacija Vozilo-Osoba
- Za vežbu: proširićemo primer 2.4 uvodeći klasu Motor primenom agregacije

Slaganje – primer 2.5 Vozilo i Motor

```
public class Motor {
        boolean radi;           //pokrenut ili ne
        private String tip;     //dizel ili benzin
        private int snaga;      // snaga u kW
        private int kubikaza; // kubikaza u ccm

        Motor() {}

        Motor(boolean radi, String tip, int snaga, int kubikaza){
                this.radi = radi;
                this.tip = tip;
                this.snaga = snaga;
                this.kubikaza = kubikaza;
        }

        ...
```

Slaganje – primer 2.5 Vozilo i Motor

```
...

public void postaviRadi(boolean radi) {
        this.radi = radi;
}

public boolean pribaviRadi() {
        return this.radi;
}
public void ukljuci() {
        if (this.pribaviRadi()==false)
                this.postaviRadi(true);
}
public void iskljuci() {
        if (this.pribaviRadi()==true)
                this.postaviRadi(false);
}

...
```

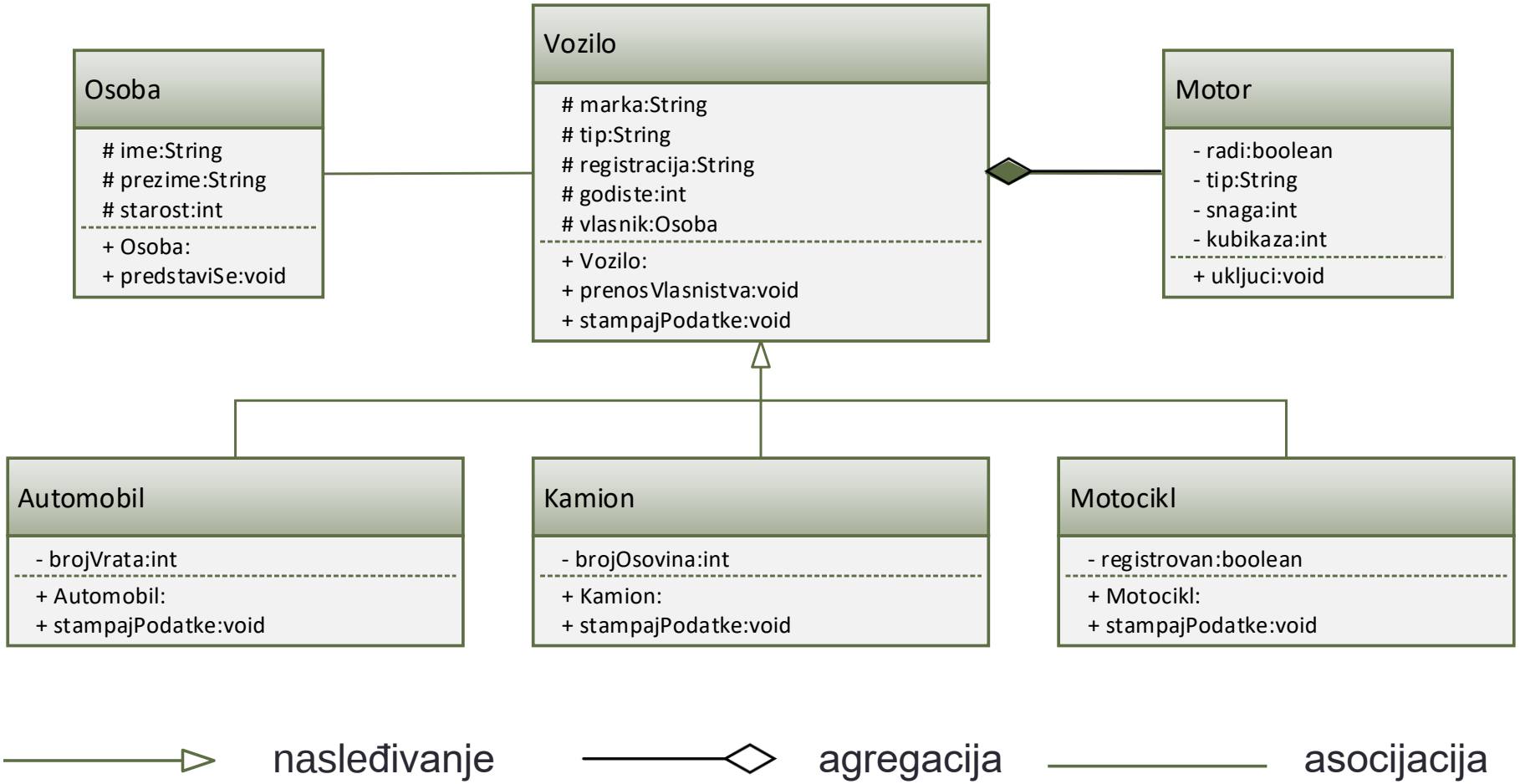
Slaganje – primer 2.5 Vozilo i Motor

```
...

public void stampajPodatke() {
                System.out.println("Informacije o motoru:");
                System.out.println("Ukljucen:" + pribaviRadi() +
                                        " Tip: " + pribaviTip() +
                                        " Snaga: " + pribaviSnagu() +
                                        " Kubikaza: " + pribaviKubikazu());
        }
}
```

Slaganje – dijagram klasa

- Klasa `Vozilo` i klase `Motor` i `Osoba`



Zadatak za vežbanje

- Zadatak 1: Osmisliti, nacrtati dijagrame klasa i realizovati u Javi klase `Institucija`, `Ucionica`, `Zaposleni` (koja nasleđuje klasu `Osoba`) i `Racunar` koje bi se mogle koristiti u programu za evidenciju zaposlenih i inventara neke obrazovne institucije. Spiskove učionica, zaposlenih i računara po učionicama čuvati u odgovarajućim nizovima (svaki sa najviše 20 članova). U kojim međusobnim odnosima se nalaze pomenute klase (nasleđivanje, asocijacija, agregacija, kompozicija)?
- Klase testirati kreiranjem objekata u glavnom programu i pozivanjem izabranih metoda. Nacrtati UML dijagram klasa rešenja.

POLIMORFIZAM

Objektna paradigma

- OOP je deo **objektne paradigme** koja obuhvata osnovne objektne koncepte:
  - **apstraktni tipovi podataka** (engl. *abstract data types*)
  - **enkapsulacija** (engl. *encapsulation)*
  - **nasleđivanje** (engl. *inheritance*)
  - **polimorfizam** (engl. *polymorphism*)




Polimorfizam

- Četvrti ključni koncept u okviru **objektne paradigme**
- **Polimorfizam** (engl. polymorphism) je **svojstvo da različiti objekti mogu da odgovaraju na iste poruke na različite načine**. Sam termin znači “mnoštvo oblika”.
- Polimorfizam praktično omogućava “virtuelizaciju” objekata.
- Polimorfni metodi mogu se **adaptirati na specifičnosti objekta** nad kojim su pozvani. Polimorfizam se može definisati i kao svojstvo da se prilikom poziva odaziva odgovarajuća verzija metode klase čiji su naslednici dali nove verzije.

Polimorfizam – primer 3.1

- Realizovati klasu `Oblik` sa atributima `boja (String), tip (String), brojStrana (int),` standardnim konstruktorom i konstruktorom koji postavlja inicijalne vrednosti, kao i metodama za promenu boje oblika i štampanje podataka o obliku.
- Realizovati klase `Kvadrat` (dodatni atribut `duzinaStranice` tipa `double`) i `Krug` (dodatni atribut `poluprecnik` tipa `double`) koje nasleđuju klasu `Oblik` i implementiraju metode za računanje površine i obima.
- Klase testirati kreiranjem više objekata u glavnom programu i pozivanjem metoda za računanje površine i obima i štampu podataka.

Primer 3.1 – klasa `Oblik`

```
public class Oblik {
        private String boja;
        private String tip;
        private int brojStrana;

        Oblik(){}

        Oblik(String boja, String tip, int brojStrana){
                this.boja = boja;
                this.tip = tip;
                this.brojStrana = brojStrana;
        }

        ...
```

Primer 3.1 – klasa `Oblik`

```
...
public void postaviBoju(String novaBoja) {
        this.boja = novaBoja;
}

public String pribaviBoju() {
        return this.boja;
}

public String pribaviTip() {
        return this.tip;
}


...
```

Primer 3.1 – klasa Oblik

```
        ...

        public int pribaviBrojStrana() {
                return this.brojStrana;
        }
        public double racunajObim() { }

        public double racunajPovrsinu() { }

        public void stampajPodatke() {
                System.out.println(   pribaviBoju() +" "+
                                      pribaviTip() +" "+
                                      pribaviBrojStrana());
        }
}
```

Primer 3.1 – klasa Krug

```
public class Krug extends Oblik {
        private double poluprecnik;

        Krug(String boja, int brojStrana, double poluprecnik){
                super(boja, "Krug", brojStrana);
                this.poluprecnik = poluprecnik;
        }

        public double racunajPovrsinu() {
                return this.poluprecnik*this.poluprecnik*Math.PI;
        }

        public double racunajObim() {
                return 2*this.poluprecnik*Math.PI;
        }

        public void stampajPodatke() {
                System.out.println(pribaviBoju() +" "+ pribaviTip() +" "+
                                        pribaviBrojStrana()+" " +
                                        racunajPovrsinu()+" "+
                                        racunajObim());
        }
}
```

Primer 3.1 – klasa Kvadrat

```
public class Kvadrat extends Oblik {
        private double stranica;

        Kvadrat(String boja, int brojStrana, double stranica){
                super(boja, “Kvadrat”, brojStrana);
                this.stranica = stranica;
        }

        public double racunajPovrsinu() {
                return this.stranica*this.stranica;
        }

        public double racunajObim() {
                return 4*this.stranica;
        }

        public void stampajPodatke() {
                System.out.println(pribaviBoju() +" "+ pribaviTip() +" "+
                                        pribaviBrojStrana()+" "+racunajPovrsinu()
                                        +" "+ racunajObim());
        }
}
```

Primer 3.1 – klasa Main

```
public class Main {
    public static void main(String[] args) {

            Krug kr = new Krug("Crvena",1, 2.0);
            Kvadrat kv = new Kvadrat("Bela",4, 1.5);

            kr.stampajPodatke();
            kv.stampajPodatke();
    }
}
```

Polimorfizam – primer 3.2

- Realizovati klasu `Zena`, izvedenu iz klase `Osoba`, koja ima i atribut `devojackoPrezime`. Objekti klase `Zena` treba da odgovaraju na poruku `predstaviSe`, ali dame skoro nikada ne otkrivaju svoje godine. Zato objekat klase `Zena` treba da ima funkciju `predstaviSe`, samo što će ona izgledati nešto drugačije, svojstveno izvedenoj klasi `Zena` – bez saopštavanja podataka o starosti, ali sa devojačkim prezimenom.
- Klase testirati kreiranjem više objekata u glavnom programu i pozivanjem metoda za predstavljanje.

Primer 3.2 – klasa Osoba

```
public class Osoba {
	private String ime;
	private String prezime;
	private int starost;

	Osoba() {}

	Osoba(String ime, String prezime, int starost){
		this.ime = ime;
		this.prezime = prezime;
		this.starost = starost;
	}

	public String pribaviIme(){
		return this.ime;
	}

	...
```

Primer 3.2 – klasa Osoba

```
...
public String pribaviPrezime(){
        return this.prezime;
}

public int pribaviStarost(){
        return this.starost;
}

public void postaviIme(String ime){
        this.ime = ime;
}

public void postaviPrezime(String prezime){
        this.prezime = prezime;
}
...
```

Primer 3.2 – klasa Osoba

```
        ...

        public void postaviStarost(int starost){
                this.starost = starost;
        }

        void predstaviSe() {
                System.out.println("Ime: " + pribaviIme() +
                                        " Prezime: " + pribaviPrezime() +
                                        " Starost: " + pribaviStarost());
        }


}
```

Primer 3.2 – klasa Zena

```
public class Zena extends Osoba {
        String devojackoPrezime;

        Zena(String ime, String prezime, String devojackoPrezime,
             int starost){
                super(ime, prezime, starost);
                this.devojackoPrezime = devojackoPrezime;
        }

        public String pribaviDevojackoPrezime(){
                return this.devojackoPrezime;
        }

        public void postaviDevojackoPrezime(String devojackoPrezime){
                this.devojackoPrezime = devojackoPrezime;
        }
        ...
```

Primer 3.2 – klasa Zena

```
        ...

        void predstaviSe() {
                System.out.println("Ime: " + pribaviIme() +
                                " Prezime: " + pribaviPrezime() +
                                " Devojacko prezime: " +
                                pribaviDevojackoPrezime());
    }
}
```

Primer 3.2 – klasa Main

```
public class Main {
	public static void main(String[] args) {

		Osoba o = new Osoba("Ivana","Ivanovic",32);
		Zena z = new Zena("Ivana","Ivanovic","Petrovic",32);

		o.predstaviSe();
		z.predstaviSe();

	}
}
```

Zadatak za rad na času

- Realizovati klase `Nastavnik`, `Asistent` i `NenastavniRadnik`, izvedene nasleđivanjem iz klase `Zaposleni`. Klasa `Nastavnik` ima dodatne atribute `zvanje` (tipa `String`) i `brojSCIRadova` (tipa `int`), klasa `Asistent` ima dodatne atribute `mentor` (tipa `String`) i `godinaDoktorskihStudija` (tipa `int`), a klasa `NenastavniRadnik` ima dodatne atribute radnoMesto (tipa String) i godineStaza (tipa int). Za svaku od klasa realizovati metodu `predstaviSe` i `racunajPlatu` uzimajući u obzir specifične atributa za svaku od klasa. Napomena: platu za nastavnike računati kao `60000+brojSCIRadova*3000`, kod asistenata kao `40000+godinaDoktorskihStudija*2000`, a kod nenastavnih radnika kao `30000+godineStaza*500.`
- Klase testirati kreiranjem više objekata u glavnom programu i pozivanjem metoda za predstavljanje i računanje plate.

APSTRAKTNE KLASE I INTERFEJSI