Napredno programiranje i programski jezici

02 OOP

Fakultet tehničkih nauka, Novi Sad
23-24/Z
Dunja Vrbaški

Objektno orijentisano programiranje

- paradigma
- modelovanje realnog sveta na formalni način
- pogodna za velike, poslovne, sisteme gde su jasno utvrđeni: učesnici, zadaci, komunikacija
- dekompozicija problema
- višestruka upotreba koda

Kad razmišljamo kako da napravimo neki sistem, obično prvo modelujemo učesnike u tom sistemu.

Prepoznajemo koje će se to klase objekata pojavljivati.

Odlučimo koje osobine i koja ponašanja su važna za naš sistem.
Prepoznajemo, definišemo i implementiramo klase.

Klasa

koncept, model
opisuje, predstavlja model ili šablon na osnovu kog se objekti kreiraju, menjaju, ponašaju, komuniciraju među sobom

```
class Student - ime, prezime
```

Objekat

jedan konkretan slučaj klase objekata
zauzima memoriju
ima konkretne vrednosti i ponaša se po pravilima definisanim u klasi

```
Student student1 “Petar” “Petrovic”
Student student2 “Ivana” “Ivanovic”
```

Klasa

koncept, model
opisuje, predstavlja model ili šablon na osnovu kog se objekti kreiraju, menjaju, ponašaju, komuniciraju među sobom

```
class Student - ime, prezime
```

tip podataka

Objekat

jedan konkretan slučaj klase objekata
zauzima memoriju
ima konkretne vrednosti i ponaša se po pravilima definisanim u klasi

```
Student student1 “Petar” “Petrovic”
Student student2 “Ivana” “Ivanovic”
```

instanca, promenljiva

Objekti su uvek u nekom **STANJU** i mogu nekako da se **PONAŠAJU**

Klasa (u osnovnom obliku) definiše **POLJA** koja će čuvati stanje i **METODE** koje će omogućiti različita ponašanja

Objekti su uvek u nekom **STANJU** i mogu nekako da se **PONAŠAJU**

Klasa (u osnovnom obliku) definiše **POLJA** koja će čuvati stanje i **METODE** koje će omogućiti različita ponašanja

`class Student`

- svi studenti imaju ime i prezime
  `polja: ime i prezime`
- i svi umeju da, na primer, ispišu svoje podatke
  `metod: ispisiPodatke`
  Svi objekti će kasnije na isti način ispisivati svoje podatke.

`Student student1 “Petar” “Petrovic”`

- `student1` ima (negde) u memoriji zapisano da su mu vrednosti polja `Petar` (`ime`) i `Petrovic` (`prezime`)
- kad god nam budu potrebni podaci o ovom učeniku možemo pozvati metod za ispis
  `ucenik1.ispisiPodatke`

Prilikom inicijalne faze modelovanja, prepoznajemo i izdvajamo karakteristike i ponašanja od značaja.

Softver za studentsku službu fakulteta

Student:
ime, prezime, mb
broj indeksa
smer, prosek

ispisiPodatke
upisiGodinu
položiIspit

Softver za studentsku ambulantu univerziteta

Student:
ime, prezime, mb
fakultet
alergije, vakcine

ispisiPodatke
zakaziPregled
kreirajUput

Prilikom inicijalne faze modelovanja, prepoznajemo i izdvajamo karakteristike i ponašanja od značaja.

Softver za studentsku službu fakulteta

Student:
ime, prezime, mb
broj indeksa
smer, prosek

ispisiPodatke
upisiGodinu
položiIspit

Softver za studentsku ambulantu univerziteta

Student:
ime, prezime, mb
fakultet
alergije, vakcine

ispisiPodatke
zakaziPregled
kreirajUput

Šta ako su to dva dela istog, univerzitetskog sistema?

Obično imamo više (velik broj) klasa

Student:
ime, prezime, mb
broj indeksa
smer, prosek

ispisiPodatke
upisiGodinu
poloziIspit

Profesor:
ime, prezime, mb
brojUgovora
katedra, zvanje
svi predmeti

ispisiPodatke
prosledi Ocene
zakaziIspit

Fakultet:
ime, adresa, žiro račun
svi studenti
svi profesori
svi ostali učesnici / ostale službe

ispisiPodatke
zapocniSkolskuGodinu
dodajNoviSmer
dodajNoviIspitniRok
generisiRaspored

Pojavljuju se još neke važni koncepti u OOP

Kako su klase (objekti) **povezani**. Na primer:

- fakultet ima: departmane, katedre, centre, laboratorije, studijske programe
- studijski programi imaju različite akreditacije (kombinacija: predmeta, nastavnika, saradnika)
- predmeti pripradaju različitim st. programima
- svi st.programi imaju rukovodioca
- student pripada nekom studijskom programu / grupi za slušanje nastave
- student ima ocene iz predmeta
- saradnici su i studenti u isto vreme (master, dr)
- ...stvari se komplikuju

Komunikacija među objektima; šalju se poruke među objektima

- da se nešto izvrši (ispisiPodatke)
- da se nešto promeni (promeniAdresu, prosledi Ocene)
- desila se neka greška, šaljem poruku o tome
- kod mene se izmenilo nešto, šaljem svim zainteresovanim objektima informaciju o tome
- ...stvari se još više komplikuju

OOP implementacija

Sintaksa i semantika za implementaciju klasa i rad sa objektima

```
class Student
{
    // POLJA
    ime
    prezime
    brIndkesa
    ...

    // METODE
    ispisiPodatke()
    upisiGodinu()
    ...

};
```

```
int main()
{
    ...
    Student s
    s.ime = “Petar”
    ...
    s.ispisiPodatke()
    ...
}
```

```
class MyClass {
    int x;
};
```

```
int main()
{
    MyClass mc;
    return 0;
}
```

```
class MyClass {
    int x;
};
```

```
int main()
{
    MyClass mc;
    mc.x = 5;

    return 0;
}
```

```
class MyClass {
    int x;
};
```

```
int main()
{
    MyClass mc;
    mc.x = 5;

    return 0;
}
```

error: 'int MyClass::x' is private within this context|

```
class MyClass {

    // PRIVATE


    // PUBLIC



    // PROTECTED
};
```

```
class MyClass {

private:

...


public:

...

};
```

```
class MyClass {

private:

   int x;


public:

   int y;

};
```

```
int main()
{
    MyClass mc;
    //mc.x = 5;
    mc.y = 3;

    return 0;
}
```

```
class MyClass {

private:

   int x;
   int xx;


public:

   int y;
   int yy;

};
```

```
int main()
{
    MyClass mc;
    //mc.x = 5;
    //mc.xx = 55;

    mc.y = 3;
    mc.yy = 33;

    return 0;
}
```

Zašto bi nešto trebalo da bude `private`, a nešto drugo `public`?
Zašto imamo ovaj mehanizam?

```
class MyClass {

private:
    int x;

public:
    void print() {
        cout << "x = " << x << endl;
    }

};
```

```
int main()
{
    MyClass mc;
    mc.print();

    return 0;
}
```

Šta će biti ispisano?

```
class MyClass {

private:
    int x;

public:
    void setX(int xx) {
        x = xx;
    }

    void print() {
        cout << "x = " << x << endl;
    }

};
```

```
int main()
{
    MyClass mc;
    mc.setX(5);
    mc.print();

    return 0;
}
```

```
class MyClass {

private:
    int x;

public:
    void setX(int xx) {
        x = xx;
    }

    int getX() {
        return x;
    }

    void print() {
        cout << "x = " << x << endl;
    }

};
```

```
int main()
{
    MyClass mc;
    mc.setX(5);
    mc.print();

    int y = mc.getX();

    return 0;
}
```

```
class MyClass {

private:
    int x;

public:
    void setX(int xx) {
        x = xx;
    }

    int getX() const {
        return x;
    }

    void print() const {
        cout << "x = " << x << endl;
    }

};
```

```
int main()
{
    MyClass mc;
    mc.setX(5);
    mc.print();

    int y = mc.getX();

    return 0;
}
```

```
void setX(int xx) {
    x = xx;
}

ILI

void setX(int x) {
    (*this).x = x;
}

ILI

void setX(int x) {
    this -> x = x;
}
```

```
int main()
{
    MyClass mc;
    mc.setX(5);

    return 0;
}
```

```
class MyClass {

private:
    int x;

public:
    void setX(int xx) {
        x = xx;
    }

    int getX() const {
        return x;
    }

    void print() const {
        cout << "x = " << x << endl;
    }
};
```

```
void print(const MyClass &mc) {
    cout << "Ispis SF: x = " <<
    mc.getX() << endl;
}
```

```
int main()
{
    MyClass mc;
    mc.setX(5);
    mc.print();
    print(mc);

    return 0;
}
```

Slobodne funkcije vs metode

```
class MyClass {

private:
    int x;

public:
    void setX(int xx) {
        x = xx;
    }

    int getX() const {
        return x;
    }

    void print() const {
        cout << "x = " << x << endl;
    }
};
```

```
void print(const MyClass &mc) {
    cout << "Ispis SF: x = " <<
    mc.getX() << endl;
}
```

```
int main()
{
    MyClass mc;
    mc.setX(5);
    mc.print();
    print(mc);

    return 0;
}
```

Šta će se desiti?