MODUL 7. ***Objektno orijentisano programiranje***

Softverska kriza je posljedica sljedećih problema softverske proizvodnje:

- Zahtjevi korisnika su se drastično povećali. Za ovo su uglavnom „krivi“ sami programeri: oni su korisnicima pokazali šta sve računari mogu, i da mogu mnogo više nego što korisnik može da zamisli. Kao odgovor, korisnici su počeli da traže mnogo više, više nego što su programeri mogli da postignu.
- Neophodno je povećati produktivnost programera da bi se odgovorilo na zahtjeve korisnika.To je moguće ostvariti najprije povećanjem broja ljudi u timu.
- Produktivnost se može povećati i tako što se neki dijelovi softvera, koji su ranije već negdje korišteni, mogu ponovo iskoristiti, bez mnogo ili imalo dorade. Laku ponovnu upotrebu koda (software reuse) tradicionalni način programiranja nije dopuštao.
- Povećani su drastično i troškovi održavanja. Potrebno je bilo naći način da projektovani softver bude što bolji i lakši za nadogradnju i modifikovanje. Za gore navedene promjene OOP je došlo kao odgovor.
- Sam naziv kaže da se pažnja posvećuje objektima.

Možemo kazati da su C++, Java i C# najrasprostranjeniji objektno orijentisani programski jezici u kojim se piše skoro 90% svih današnjih velikih softverskih projekata.

Sam naziv kaže da se pažnja posvećuje objektima.

Suštinski nedostatak proceduralnog pristupa sastoji se u tome što se previše pažnje posvećuje postupcima odnosno procedurama koji se obavljaju nad podacima (obično izvedenim u formi funkcija), a premalo samim podacima.

Podaci se posmatraju kao sirova hrpa činjenica, čija je jedina svrha postojanja da budu proslijeđeni nekoj funkciji koja ih obrađuje.

No, prije bi se moglo reći da je tačno obrnuto gledište:

- funkcije služe da bi opsluživale podatke. Funkcije postoje radi podataka, a ne podaci radi funkcija!
- **Klase** čine osnovu objektno orijentiranog pristupa programiranju. One predstavljaju složene tipove podataka slične strukturama, ali koje ne objedinjuju samo sirove podatke, nego i kompletan opis postupaka koji se mogu primjenjivati nad tim podacima.

Osnovna rješenja koja pruža OOP, a C++ podržava su:

Četiri postulata objektno orijentiranog programiranja:

1. **Sakrivanje informacija** (princip prema kojem se svi atributi klase definiraju isključivo kao privatni, dok se sav pristup atributima vrši preko funkcija članica);

2. **Enkapsulacija** (objedinjavanje u jednu cjelinu podataka i postupaka koji se mogu primjenjivati nad tim podacima);

3. **Nasljeđivanje**

4. **Polimorfizam**

Strukture dosta podsjećaju na klase.

Da se podsjetimo:

```
struct Ucenik
{
        int BrojUDnevniku;
        string Prezime;
        string Ime;
        int Uspjeh;
        int BrojOpravdanih;
        int BrojNeopravdanih;
}
```

Kod upotrebe ključne riječi “struct”, sva polja smatraju se javnim.

S druge strane, ključna riječ “class” sve atribute i metode za koje se eksplicitno ne kaže da su javni smatra privatnim, i na taj način podstiče sakrivanje informacija.

Klase omogućavaju programeru modeliranje objekata koji imaju atribute (predstavljene kao podaci) i ponašanje ili operacije (predstavljene kao funkcije ili metode)

Klasa predstavlja apstrakciju koja definiše zajednička svojstva i zajedničke postupke koji su svojstveni svim objektima nekog tipa.

Npr. Kreiramo klasu Mačak.

Za opis mačke koristimo podatke koji su zajednički za zve mačke:

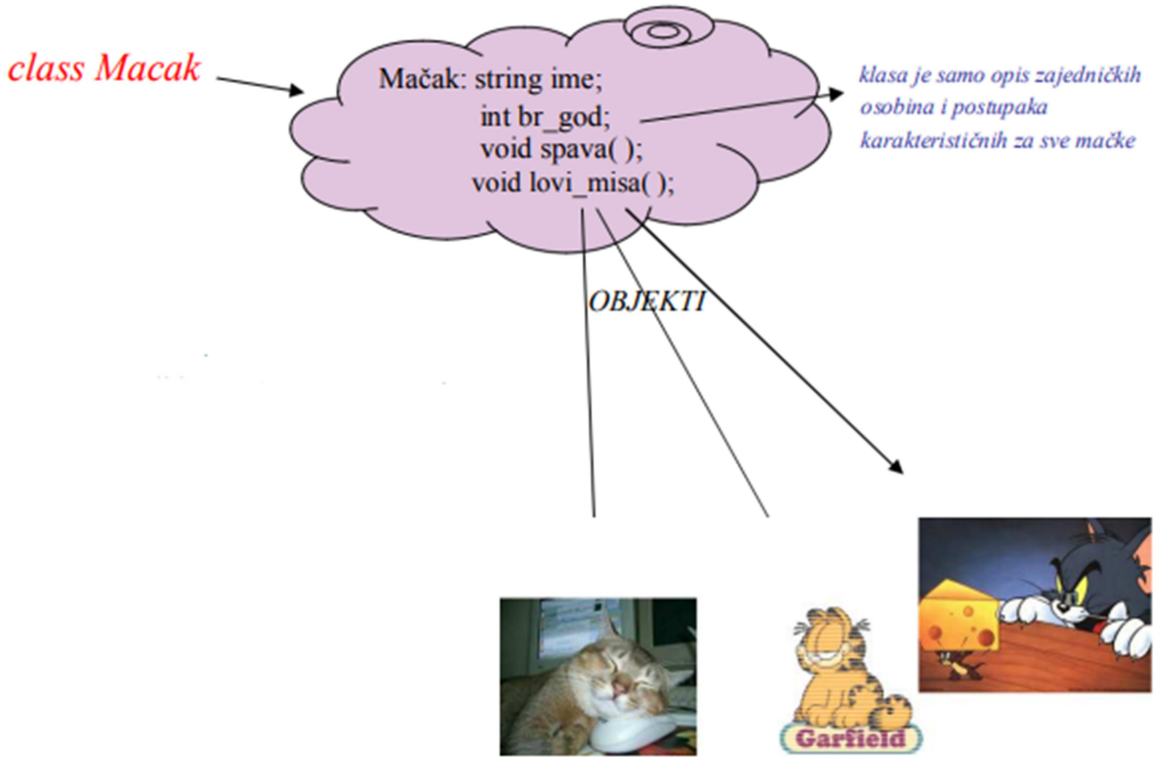
- ime,

- broj godina,

- prede,

- lovi miša...

ime, broj godina i slični podaci se predstavljaju kao atributi a ponašanje mačke kao što je prede, lovi miša i sl. predstavlja se u klasi kao metoda (funkcija).



Klasa je samo opis, a objekat je stvarna, konkretna realizacija tog opisa.

Primjer *Automobil*

Podaci ili atributi koji opisuju automobil bi mogli biti:

- Marka
- Model
- Boja
- GodinaProizvodnje
- VrstaGoriva
- Potrošnja
- Registracija
- DatumRegistracije
- SljedeciDatumRegistracije
- VrstaGuma (Ljetne,Zimske)

Broj atributa zavisi od same primjene. Stanja koja bi se mogla definisati preko funkcija (metoda) bi bila:

- KadaJeSljedeciDatumRegistracije,

- PostaviZimskeGume;

- GodineStarosti

...

Kada definišemo objekat MojAuto tipa Automobil možemo postaviti podatke i pratiti njegova stanja. Dozvoljeno je formirati N objekata i pratiti vozni park sa potpuno istim kodom uz definisanje ID broja pojedinog objekta (automobila).

Primjer:

```
#include<iostream>
#include<string>
using namespace std;

class Macak   {
public:                                          // način pristupa članovima klase
        string ime;                              //atribut
        string prede ( )                         // ovo je metoda koja opisuje ponašanje mačke
                {
                        return " ZZ Z zz...z.ZZZ";
                }
};                                               // na kraju klase obavezno ide };

int main( ) {
        Macak spavalica; // kreiramo objekat
        cout<<"unesi ime:";
        cin>>spavalica.ime;             // pomoću kreiranog objekta obracamo se članovima klase
        cout<<"Macak "<<spavalica.ime <<" voli da prede "<<spavalica.prede( )<<endl;
        return 0;
}
```

Članovi klase su :

- ***atributi*** *(podaci, osobine) koji opisuju stanje objekta i*
- ***metode*** *(funkcije, ponašanje) namjenjene definisanju operacija nad podacima klase*

Primjer klase Datum

```
#include<iostream>
using namespace std;
class Datum
{
   int Dan;
   int Mjesec;
   int Godina;
   public:
bool PostaviDatum(int d,int m,int g)
   {
      if(d<0 || d>31)
      {
        cout<<"Pogresan dan, obratite paznju na opseg "; // Set metoda
        return false;
      }
        else
        {
          Dan = d;
        }

      Mjesec = m;
      Godina = g;
   }
   void UzmiDatum(int izbor) // Get metoda
   {
      switch(izbor)
      {
        case 1:
            cout<<Dan<<"/"<<Mjesec<<"/"<<Godina<<endl;
            break;
        case 2:
            cout<<Dan<<"."<<Mjesec<<"."<<Godina<<endl;
            break;
      }
   }
};

int main()
{
   Datum D1;
   int d,m,g;
   cout<<"Unesite datum:  dan-> (1-31) ";
   cin>>d;
   cout<<"Unesite datum:  mjesec-> (1-12) ";
   cin>>m;
   cout<<"Unesite datum:  godina -> (0-9999) ";
   cin>>g;
```

```
    if (D1.PostaviDatum(d,m,g))
    {
     cout<<"Izaberite koji format prikaza zelite  1-(dd/mm/gggg),2-(dd.mm.gggg>),3-
(dd,NazivMjeseca,gggg)  : ";
     int izbor;
     cin>>izbor;
     D1.UzmiDatum();
    }


    return 0;
}
```

Vježba:

1. Dopuniti prikaz datuma pod opcijom 3 da prikazuje u formatu -(dd,NazivMjeseca,gggg)
2. Napraviti metod koji određuje da li je unesena godina prijestupna.
3. Relizovati klasu „Ucenik“ sa atributima Id,Prezime,Ime,Uspjeh.
   a) Napraviti 5 instamci klase Ucenik
   b) Napraviti metodu za unos učenika;
   c) Napraviti metodu za ispis učenika za odabrani uspjeh.

   Objašnjenje:

   Ako su unešeni sljedeći podaci:

ID	Prezime	Ime	Uspjeh
1	Miki	Maus	3
2	Kremenko	Džo	5
3	Paja	Patak	2
4	Mornar	Popaj	3
5	Pink	Panter	4

   Ispis bi se ostvario na sljedeči način:

   Unesite uspjeh : 3

   PrikazUcenikaZaOdabraniUspjeh(3)

   Metoda bi trebala pokazati sljedeći rezultat

   Učenici koji imaju uspjeh 3

ID	Prezime	Ime	Uspjeh
1	Miki	Maus	3
2	Mornar	Popaj	3