

Objektno orijentirano programiranje

1

Objekti

.....

- ❖ Objektno orijentirano programiranje: Postupak izrade aplikacija u kojem se svojstva apstraktnih ili stvarnih objekata modeliraju programskim klasama i objektima.
- ❖ Objekt
 - ❖ U stvarnom svijetu okruženi smo objektima (automobil, računalo, pas, drvo, ...). Svaki objekt definiran je stanjem i ponašanjem. Npr. za automobil stanje određuje trenutna brzina, količina goriva u spremniku i sl., a ponašanje može biti ubrzavanje, kočenje, skretanje i sl.
 - ❖ Objekt u programskom okruženju je skup varijabli i pripadnih funkcija. Varijable određuju stanje, a funkcije ponašanje objekta. Varijable objekta nazivaju se i varijable primjerka (*instance variables*) – svaki primjerak (instanca) određenog objekta sadrži vlastitu kopiju varijabli primjerka. Funkcije mijenjaju stanje objekta, a po potrebi mogu stvarati i nove objekte.

2

Klase

- ❖ Klasa je nacrt objekta, odnosno predložak po kojem je određen objekat stvoren.
- ❖ Npr. određeni tip automobila definiran je jednim nacrtom (odnosno skupom nacrti). Na temelju jednog nacrti moguće je proizvesti više primjeraka istog tipa automobila (tj. objekata), koji će se međusobno razlikovati po stanju i ponašanju.
- ❖ Podskup stanja i ponašanja (varijabli i funkcija) može biti zajednički svim objektima određene klase. Nazivamo ih varijablama i funkcijama klase.

3

Ovijanje (*encapsulation*)

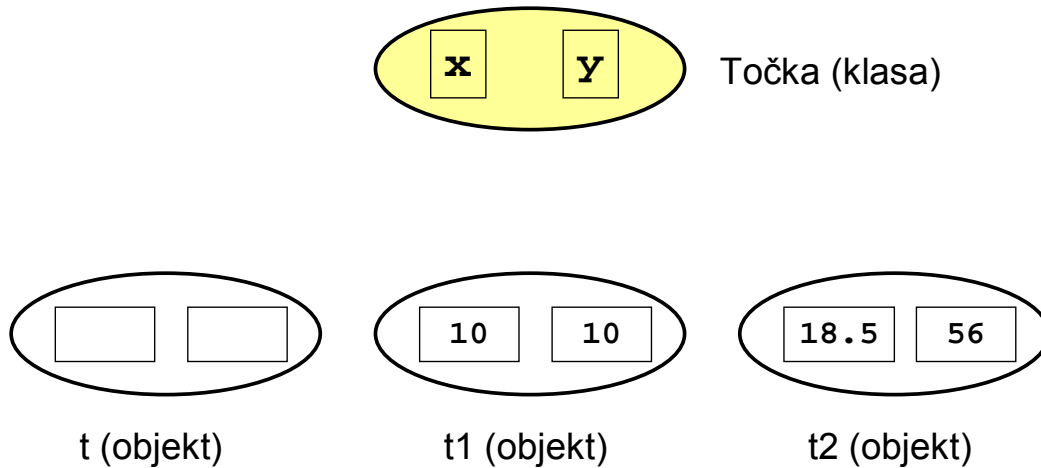
- ❖ Omogućuje skrivanje detalja implementacije određene klase, tako da korisnik određene klase komunicira s pripadnim objektima preko javnog sučelja, definiranog skupom javnih varijabli i funkcija.
- ❖ Na taj način moguće je izmijeniti unutarnju strukturu klase, pod uvjetom da sučelje objekta prema okolini ostane nepromijenjeno. Npr. korisnik automobila ne treba znati ništa o načinu rada motora. Izvođenje poboljšanja na motoru u cilju poboljšanja voznih svojstava ili pouzdanosti ne utječe na principe vožnje.
- ❖ Skrivanje detalja implementacije doprinosi modularnosti. Izvorni kod implementacije određene klase moguće je održavati neovisno o ostalim klasama, a za korištenje određene klase dovoljno je poznavati njeno sučelje.

4

Primjer: klasa Točka

.....

- ❖ Program definira prikazanu klasu i stvara sljedeće objekte:



5

Konstruktor i destruktor

.....

- ❖ **Konstruktor** je posebna funkcija, čije ime je jednako imenu klase, koja se automatski izvršava u trenutku stvaranja objekta.
- ❖ Najčešće se koristi za inicijalizaciju varijabli objekta.
- ❖ U pravilu se definira više konstruktora, ovisno o načinu inicijalizacije koji se koristi. C++ omogućuje definiranje više funkcija s istim imenom, koje se razlikuju prema broju i tipu argumenata (*overloading*).
- ❖ **Destruktor** je posebna funkcija, čije ime je jednako imenu klase (uz prefiks '~'), koja se automatski izvršava u trenutku uništenja objekta.

6

Ovijanje (*encapsulation*)

.....

- ❖ Ovijanje (*encapsulation*) je mehanizam kontrole pristupa varijablama i funkcijama klasa i objekata.

```
class Tocka {  
private:  
    double x;  
    double y;  
    . . .  
}
```

- ❖ Nije moguć direktan pristup privatnim članovima - ostvaruje se preko "*getter*" i "*setter*" funkcija.

7

Nasljeđivanje

.....

- ❖ Nasljeđivanje je mehanizam pomoću kojega je na temelju postojećih klasa moguće definirati nove i proširene klase.
- ❖ Svojstvo da funkcija podklase zamijeni funkciju nadklase istog imena i argumenata omogućuje korištenje nasljeđivanja za prilagođavanje svojstava klase čije ponašanje je blisko željenom.
- ❖ C++ omogućuje višestruko nasljeđivanje.

8