Objekti i klase

Objektno orijentisano programiranje (**Object Oriented Programming, OOP**) je novi pristup realizaciji softvera kao modela realnog sveta.

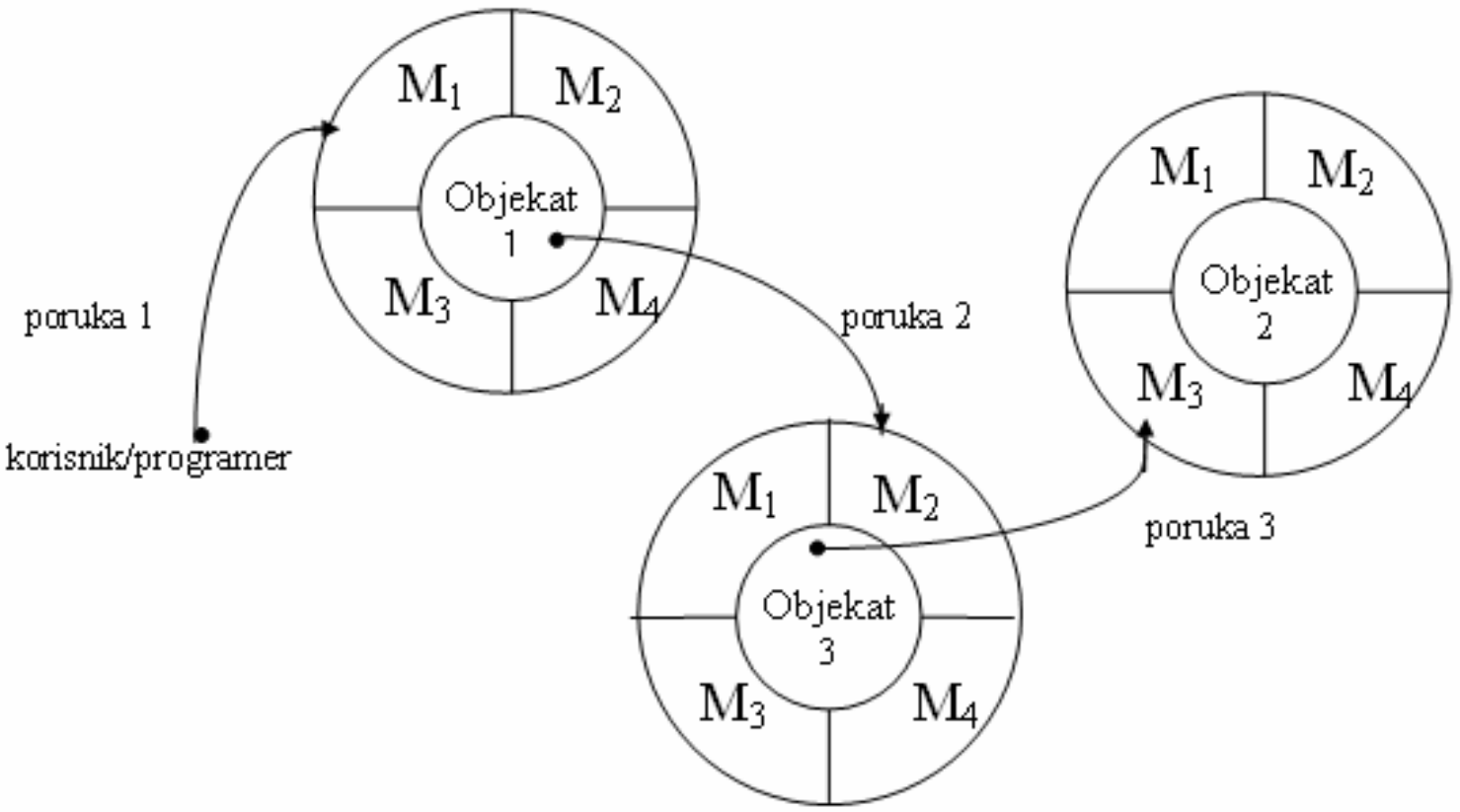
U objektnim programskim sistemima sve je predstavljeno kao objekat (procesi, tekst, U/I-operacije itd.).

Objektno programiranje zapravo najsličnije pisanju simulacija za realne objekte

- koncept procedura i podataka (koji se koristi u tradicionalnim višim programskim jezicima) zamenjen konceptom objekata i poruka:
  - objekt predstavlja pakovanje informacija i opis za njihovu manipulaciju (skup operacija i procedura koje se nad datim podacima mogu izvršiti), a poruka je specifikacija jedne od manipulacija objektom.

- Za razliku od podataka koji su nepromenljivi (ili vrlo malo promenljivi), objekti su vremenski promenljivi i imaju stanja koja odgovaraju realnom svetu.
- Za razliku od procedura koje opisuju kako se izvodi određeni postupak obrade, poruka sadrži opis šta pošiljalac želi da se uradi, a primalac određuje šta će se tačno dogoditi, odnosno on obavi posao ili prenese poruku drugim objektima.
- Objekat ima svoje unutrašnje stanje čija je realizacija nedostupna drugim objektima i operacije (metode) koje se nad njim spolja mogu izvršavati, slika 1.

Objekti i klase

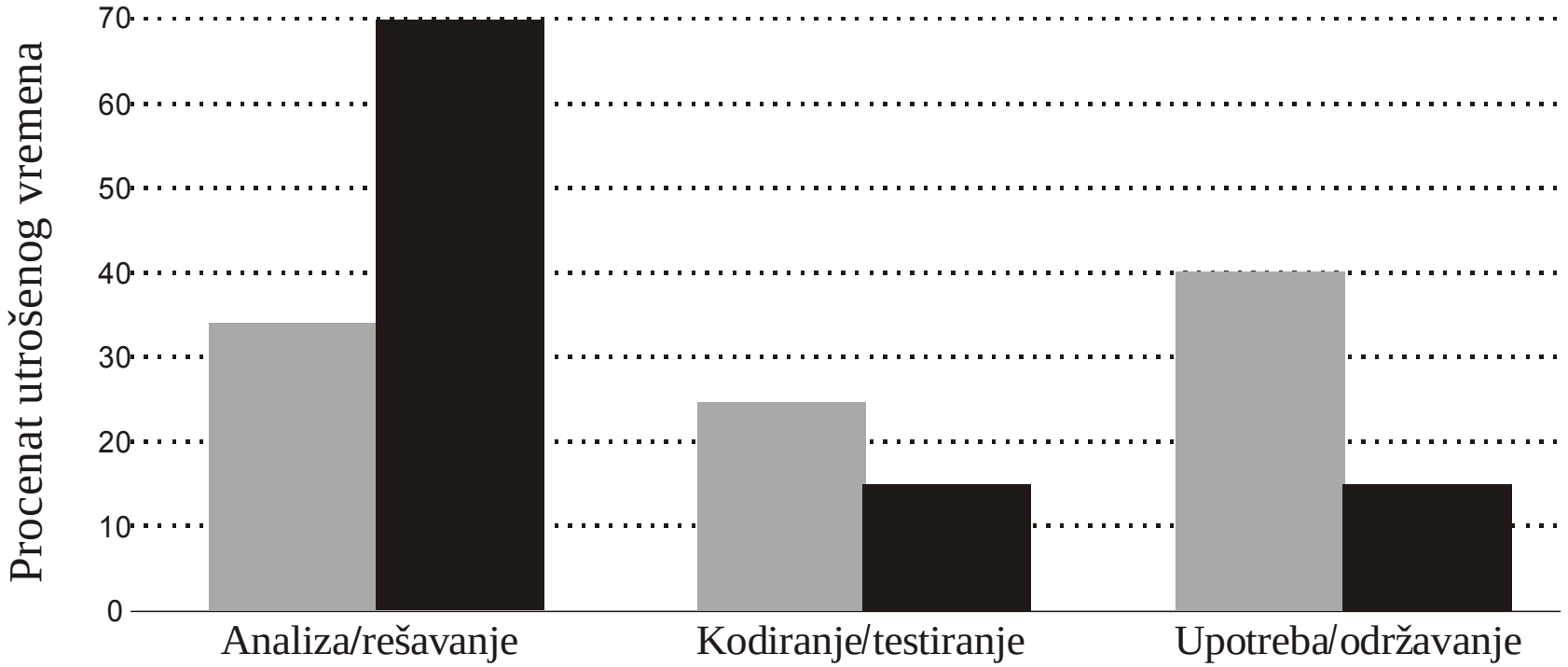


(a) Objekti šalju poruke jedan drugome

Metod 1	Metod 2	Metod 3	Metod 4	*Korisnik klase zahteva (odabira) metod pomoć poruke*
Interna implementacija metoda				*klasa određuje stvarni sadržaj svojih metoda*

b) Klase

Raspodela utrošenog vremena u raznim fazama razvoja programa



- Modularna organizacija softvera, modeliranje realnog sveta sa tačke gledišta korisnika, jednostavna komunikacija korisnika sa računarskim okruženjem, mogućnost višestrukog korišćenja istog kôda, biblioteke programskih modula koje se mogu dalje proširivati, laka modifikacija i proširivanje postojećeg programa bez ponovnog kodiranja i brzi razvoj aplikacija

Pravila objektnog programiranja

- Struktura programa treba što više da liči na strukturu samog problema tako da:
- Svaku zasebnu logičku celinu ili ideju treba realizovati kao klasu,
  - Svaki pojedinačni postupak obrade treba realizovati kao objekat neke klase;
  - Ako dve ili više klasa imaju nešto važno kao zajedničko svojstvo, onda treba napraviti klasu koja sadrži to što im je zajedničko (nasleđivanje svojstava);
- Kako većina klasa u jednom programu ima neke zajedničke osobine treba napraviti jednu univerzalnu osnovnu klasu. Ovu klasu treba projektovati vrlo pažljivo;
- Ne koristiti globalne objekte i globalne podatke;
- Ne koristiti globalne funkcije, itd.

APSTRAKCIJA PODATAKA

- Programski jezik mora programeru ponuditi mogućnost sistematskog definisanja novih tipova podataka, zajedno sa operacijama koje su nad njima mogućne.
- Apstrakcija je princip ignorisanja onih osobina nekog objekta koje nisu relevantne u datoj situaciji, tj. usredsređivanje na bitne stvari.

- Za svaki objekat definišu se njegovi ***atributi*** (**attributes**), odnosno *svojstva* (**properties**) koja se mogu menjati samo preko dozvoljenog skupa operacija - ***metoda*** (**methods**), odnosno *ponašanja* (**behaviors**)[1].
- Dakle, apstraktni tip podataka sadrži ne samo dozvoljeni skup vrednosti nego i specifikaciju operacija koje su legalne za novi tip podataka. Struktura podataka je konkretan način ostvarivanja apstraktnog tipa podataka.

  [1] U raznim knjigama koriste se različiti nazivi za svojstva i ponašanje objekata: resursi (**resources**) ili privatne promenljive članovi (**private members variables**) za svojstva, i usluge (**services**) ili operacije (**operations**) i funkcije članovi (**member functions**).

Korisnički definisani tipovi podataka

- **Type (Structure)** Osoba
  sPrezime **As String** * 15
  sIme **As String** * 15
  sPunoIme **As String** * 30
  DatRod As Date
  sAdresa **As String** * 30
  nBrojStana **As Integer**
  nPostanskiBroj **As Integer**
  sPreduzece **As String** * 20
  lTelefon As Long
  sEmail **As String** * 20
  sNapomene **As String** * 255
- **End Type (Structure)**

- Postoje dve strategije za stvaranje objekata i zadavanje vrednosti promenljivima primerka:

*Korišćenjem (ako ih programski jezik ima) specijalnih metoda u samoj klasi: kreiranje-objekata (**object-creation**), koje istovremeno inicijalizuju neke ili sve promenljive primerka;*

*Korišćenjem generičkog metoda: novi-objekt (**new**) za kreiranje primerka klase bez inicijalizacije promenljivih primerka.*

- Objekti koje se više neće koristiti trebalo bi da budu izbačeni iz operativne memorije ili čak uništeni, odnosno vrši se oslobađanje memorije. To se može postići na dva načina:

*Pomoću eksplicitne operacije za brisanje (**dispose**, **delete**) u Visual Basicu se koristi sledeća naredba:* ***Set MyObject=Nothing***

*Pomoću implicitnog uklanjanja objekta kad on više nije potreban ni jednoj promenljivoj ili nadređenom objektu (**parent**) u programskom okruženju.*

- Prvi korak u projektovanju objektno orijentisanog modela jeste uočavanje i opisivanje objekata.

1. Uočavanje, izbor, odnosno identifikovanje objekata;
2. Definisanje metoda, odnosno ponašanja tih objekata;
3. Izbor podataka koji su od značaja, odnosno definisanje svojstava objekata.

- Postupak je iterativan jer se u toku rada otkrivaju novi podaci i osobine i ponašanja.

Identifikovanje objekata

- Za svaku aplikaciju postoji određeni skup objekata koji su od interesa. Određivanje tog skupa objekata nije jasno određen zadatak niti je jednoznačan.
- Da bi nešto predstavljalo objekat ono mora imati svoju ulogu u aplikaciji, odnosno mora imati stvari koje može da uradi (metodi, ponašanja) i stvari koje zna (svojstva).
- Preporuke za izbor objekata: U zahtevu ***predmeti koje je student izabrao, prijavio i položio*** imenice: “student” i “predmet” i predstavljaju objekte - kandidate za aplikaciju.

Definisanje ponašanja

- Ponašanje nekog objekta određuje šta objekat može da uradi, odnosno kako će se ponašati.
- Ponašanje, odnosno metodi sadrže, operacije koje objekti izvršavaju, manipulaciju podacima koju dati objekat zahteva i interakcije koje objekat dozvoljava.
- Metode je moguće pronaći analizom glagola koji definišu zahteve ili ponašanje objekata u aplikaciji.
- Student “**bira**“ predmete koje “**sluša**“ i “**polaže**“, “**prijavljuje ispite**“ i “**polaže**“ ih.

Definisanje svojstava

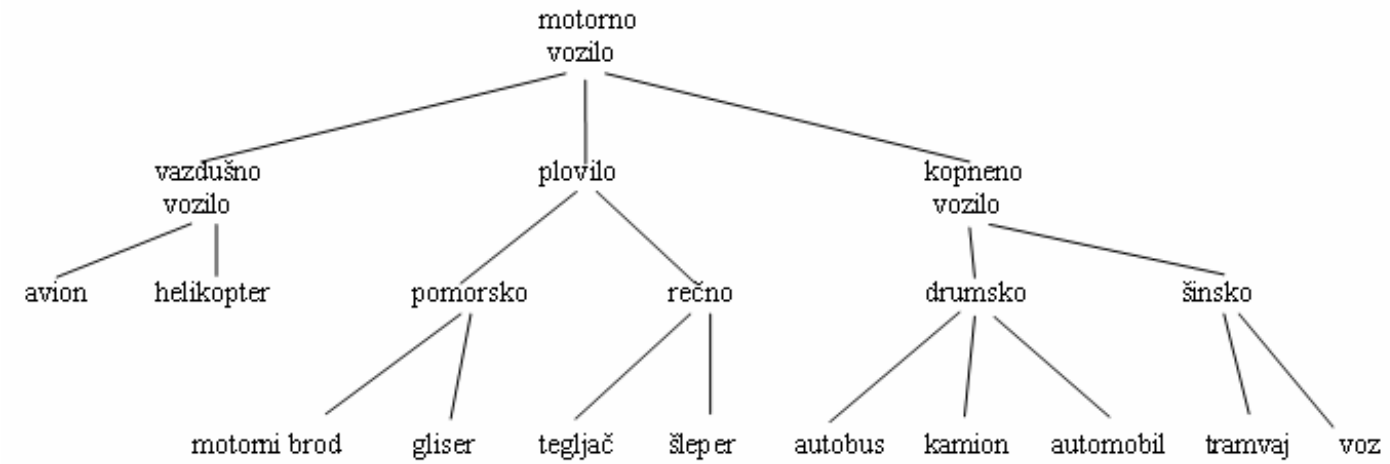
- Svojstva određuju šta će objekat znati, a čine ih podaci koji su pridruženi objektu.

- Pri određivanju podataka treba uzeti u obzir ne samo osnovne podatke, koji su potrebni za opis tog objekta (na primer ime, broj indeksa, itd.), već sve podatke koji su potrebni u aplikaciji, odnosno treba formirati spisak svih svojstava.

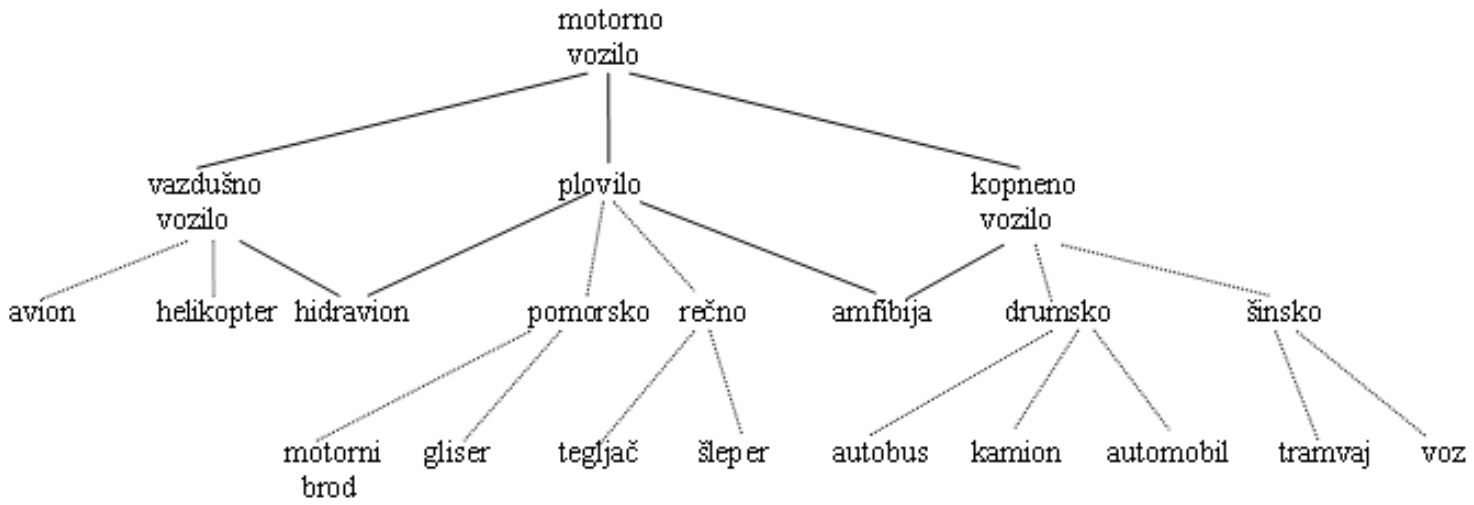
Klase

- Klasa (**class**) je struktura podataka koja na sintaksnom nivou uključuje i operacije nad tim podacima.
- Klasa je skup objekata, a objekti su konkretni primeri, primerci, klasa.
- Klasa je novi tip podataka koji sadrži dve komponente: tip objekta (koji nosi informacije o strukturi podataka) i skup metoda (tj. operacija koje se mogu primeniti na sve objekte te klase).

Objekti su povezani preko relacija, a osnovna tri tipa ovih relacija su: *potklase* (**subclasses**), *kontejneri* (**containers**) i *saradnici, kolaboratori-kooperativne klase* (**collaborators**).



Generičko stablo klase motornih vozila



Stablo klase motornih vozila sa višestrukim nasleđivanjem

Potklasa

- Potklasa je relacija tipa “***je***”, odnosno objekat u jednoj klasi *je* podtip neke druge klase.
- Klasifikacije su u prirodi i svakodnevnom životu česte: automobil ***je*** potklasa motornih vozila, profesor ***je*** potklasa zaposlenih osoba .
- Ove relacije se otkrivaju u toku projektovanja aplikacije ili postupcima specijalizacije i detaljisanja ili postupcima generalizacije i agregacije.
  - potklase nasleđuju sve atribute svoje natklase (deca nasleđuju svojstva roditelja),
  - potklase nasleđuju sve operacije svoje natklase (deca nasleđuju ponašanje roditelja).

Kontejner

- Kontejner je relacija tipa “***ima***” ili “***sadrži***”, odnosno jedan objekat može u sebi da ***ima*** ili ***sadrži*** druge objekte, ili može čak biti sastavljen od drugih objekata.
- Na primer, automobil ima motor, sedišta, kočnice, upravljački mehanizam itd.
- Tipičan primerak kontejnera u Visual Basicu su obrasci (**form**). Obrasci inače predstavljaju korisnički interfejs, odnosno ono što korisnik vidi i sa čime neposredno radi. Svaki obrazac sadrži razne kontrole (labele, tekstualna polja, komandnu dugmad, okvire, itd.).

Objekti saradnici - kolaboratori

- Kolaboratori, saradnici su dva objekta koji nisu direktno povezani već jedan objekat ***koristi*** drugi objekat radi postizanja nekog cilja. Objekti su u relaciji tipa “koristi”.
- Ljudi koriste svoje automobile, u listi za plate se koristi kalendar i štampač.
- Ako neki objekat koriste drugi objekti, onda on mora biti definisan u toj aplikaciji.
- Objekti saradnici otkrivaju se u procesu provere da li u aplikaciji postoje svi objekti koji su potrebni nekom objektu.

- ***Enkapsulacija*** ("skrivanje informacija") je princip prema kojem su detalji iz programske strukture "nevidljivi".
- Komunikacija između različitih struktura može da se vrši samo na tačno definisane načine, korišćenjem unapred određenih procedura.
- Ovaj princip pojednostavljuje međusobnu povezanost modula u programu što omogućava bolju kontrolu rada sistema.
- Osim toga, smanjuje se međuzavisnost modula čime se obezbeđuje lakša i jednostavnija modifikacija pojedinih modula.

Nasleđivanje

- Neka je klasa ***Osoba*** definisana kao tip objekta koji ima sledeće podatke:

((IME, char),(PREZIME,char),(DAT_ROĐ,date))

- nad kojima se može izvršiti sledeći skup operacija (metoda):

{*štampaj_ime, štampaj_prezime, štampaj_datrođ, promeni_prezime*},

- Jedan konkretan primerak ove klase bi bio:

((IME, Ana), (PREZIME, Kiš),(DAT_ROĐ,1503955))

- Klasa *Profesor* kao potklasu klase *Osoba*, koja ima sledeća svojstva (podatke):

((KATEDRA, char), (RAD_MES,char), (PLATA,single), (DAT_ZAP,date))

- nad kojima se može izvršiti sledeći skup operacija (metoda):

{*štampaj_katedru, povećaj_platu, štampaj_datzap, promeni_katedru,. . .*},

- Jedan konkretan objekat klase *Profesor* bi bio:

((IME, Ana), (PREZIME, Kiš), (DAT_ROĐ,1503955), (KATEDRA,NRT), (RAD_MES,Profesor), (PLATA,990), (DAT_ZAP,290179))

- Svi objekti klase ***Profesor*** automatski će naslediti i sva obeležja natklase ***Osoba***:

IME, PREZIME i DAT_ROĐ, a tipu objekta *Profesor* treba dodati obeležja: RAD_MES tipa *char*, KATEDRA tipa *char*, PLATA tipa *single* i obeležje DAT_ZAP tipa *date*.

Objekti klase *Profesor* nasleđuju i sve metode iz klase *Osoba*, a dodaju im se još neke: *povećaj_platu, štampaj_platu, promeni_radmes, promeni_odeljenje* itd.

- klasu *Student* kao potklasu klase *Osoba*, koja ima sledeća svojstva (podatke):

((KATEDRA, char),(
BR_IND,char),(DAT_UPIS,date), ...)

- nad kojima se može izvršiti sledeći skup operacija (metoda):

{*štampaj_katedru, štampaj_datupis, promeni_katedru,. . .*},

Nasleđivanje objekata, prototipovi i izaslanici

- Nasleđuju se ne samo promenljive klasa već i promenljive primeraka. Ovaj oblik nasleđivanja zove se delegiranje (**delegation**).
- Ovaj mehanizam je osnova za realizaciju sistema prototipova koji daju mogućnost primene postupka *odozdo nagore* za rešavanje zadataka pomoću računara, jer sada i objekti-prototipovi mogu da proizvode objekte-primerke.
- Na ovaj način stvara se mogućnost da se najpre napravi koncept, a zatim se on generalizuje tako što se naznači koji aspekt koncepta treba da se izmeni.

- Jedan pravougaonik kao jedan zaseban objekat i kvadrat kao poseban slučaj pravougaonika.
- Naime, kvadrat liči na pravougaonik: ima četiri stranice i četiri ugla od 90^0, ali su mu sve stranice iste.
- *Prvi pravougaonik koji je nacrtao postaje mu prototip za sve ostale pravougaonike, kao i za prvi kvadrat.*
- *Kada na osnovu pravougaonika nacrta prvi kvadrat on mu postaje prototip za sve ostale kvadrate itd*.
- Pravougaonik je opisan osobinama: *centar*, *dijagonala* i *odnos stranica*

- Stanje objekta tipa pravougaonik (**REC**) potpuno je definisano informacijama: *centar*, *dijagonala* i *odnos stranica*.
- On takođe ima i niz operacija koje opisuju njegovo ponašanje: *Pomeranje*, *Rotiranje*, *Dimenzionisanje* itd.
- Dakle, ovaj pravougaonik-objekat je primerak klase pravougaonika na koji se mogu primeniti sledeće poruke i selektori:
- *Pomeri*, *Rotiraj*, *Redimenzioniši* itd.,
- i ima pristupne metode:
- *Centar*, *Dijagonala* i *Odnos*.

- Kvadratni objekat KV1 sličan je objektu pravougaonik; *operacije su iste, a jedina razlika je to što je odnos stranica uvek jednak* 1.
- Znači objekat KV1 možemo posmatrati kao izaslanika objekta REC.
- To dalje znači da će iz objekta REC biti nasleđene sve operacije i promenljive koje nisu ponovo definisane u objektu KV1.
- Samo operacija *redimenzioniši* biće ponovo definisana u objektu kvadrata sve ostale se prosleđuju pravougaoniku.