JAVA

OOP paradigma

Programske paradigme

- Programski jezici se međusobno razlikuju po načinu na koji modeliraju realne probleme (programskim paradigmama koje podržavaju).

Programming paradigm
Obrazac koji služi kao doktrina/učenje koje se sledi u procesu programiranja

Programming technique
Strategija rešavanja problema koja se primenjuje u algoritmu.
Primer: strategija ‘podeli pa vladaj’

Programming style
Stril kojim je program napisan. (elegenacija ili nedostatak elegancije)

Programming culture
Sveukupan izraz jednog programera, koji je često usko povezan sa familijom programskih jezika – definišu je glavne paradigme, stilovi i programerske tehnike koje jedan programer koristi ili kojima vlada.

Programske paradigme

- (mentalni) model računara diktira način na koji se rešavanje problema opisuje.
- Paradigma programiranja – osnovni način struktuiranja misli tokom programiranja
- Jedna moguća podela:
  - imperativno vs. deklarativno
- Neke paradigme
  - (**imperativno**) Strukturno, **proceduralno programiranje** – Pascal, Basic,C

    koncept – “prvo uradi ovo, pa ovo”
  - (**deklarativno**) Paradigma **funkcionalnog programiranja** – program se sastoji iz definicija (matematički definisanih) funkcija, a računanje se svodi da evaluaciju vrednosti definisanih f-ja za zadate argumente (Lisp, Haskell)

    koncept – “izačunaj vrednost izraza”
  - (**deklarativno**) Paradigma **logičkog programiranja** – (Prolog), zasnovano na aksiomama i pravilima zaključivanja,

    koncept – “odgovori na pitanje”
  - (**imperativno**) **Objekto-orjentisana paradigma**

    koncept – “modeliranje fenomena realnih sistema definisanjem komunikacije između objekata tog sistema”

Primer – n!

proceduralno **C**

```
int fakt(int n)
{
  if (n==1) return 1;
  return n*fakt(n-1);
}


…
  printf("%d",fakt(10));
```

funkcionalno **LISP**

```
(defun fakt(n)
  (cond
    ((= n 0) 1)
    (t (* n (fakt (- n 1))))
   )
)

…
fakt(10)
```

objektno-orjentisano **Java**

```
class Calc
{
   static int fakt()
   {
     if (n==1) return 1;
     return n*fakt(n-1);
   }
}

…
System.out.println(Calc.fakt(10));
```

logičko **PROLOG**

```
fakt(0,1).
fakt(X,Y):- U is X-1,fakt(U,Z),
            Y is X*Z.


…
fakt(10,120).
fakt(10,X).
```

Objektno-orijentisana paradigma

- Objektno-orijentisano programiranje je metod implementacije programa po kojem su:
  - Programi organizovani kao kolekcije objekata koji sarađuju
  - Svaki objekat predstavlja primerak neke klase
  - Sve klase su članovi neke hijerarhije klasa u kojoj su klase povezane relacijama nasleđivanja
- Jezik je **objektno-orjentisan** ako i samo ako:
  - Podržava objekte koji su apstrakcije podataka sa interfejsom preko imenovanih lokacija i skrivenim lokalnim stanjem
  - Objekti imaju pridružen tip
  - Tipovi mogu nasleđivati atribute nadtipa
- Ako jezik ne podržava nasleđivanje naziva se **objektno-baziranim** jezikom

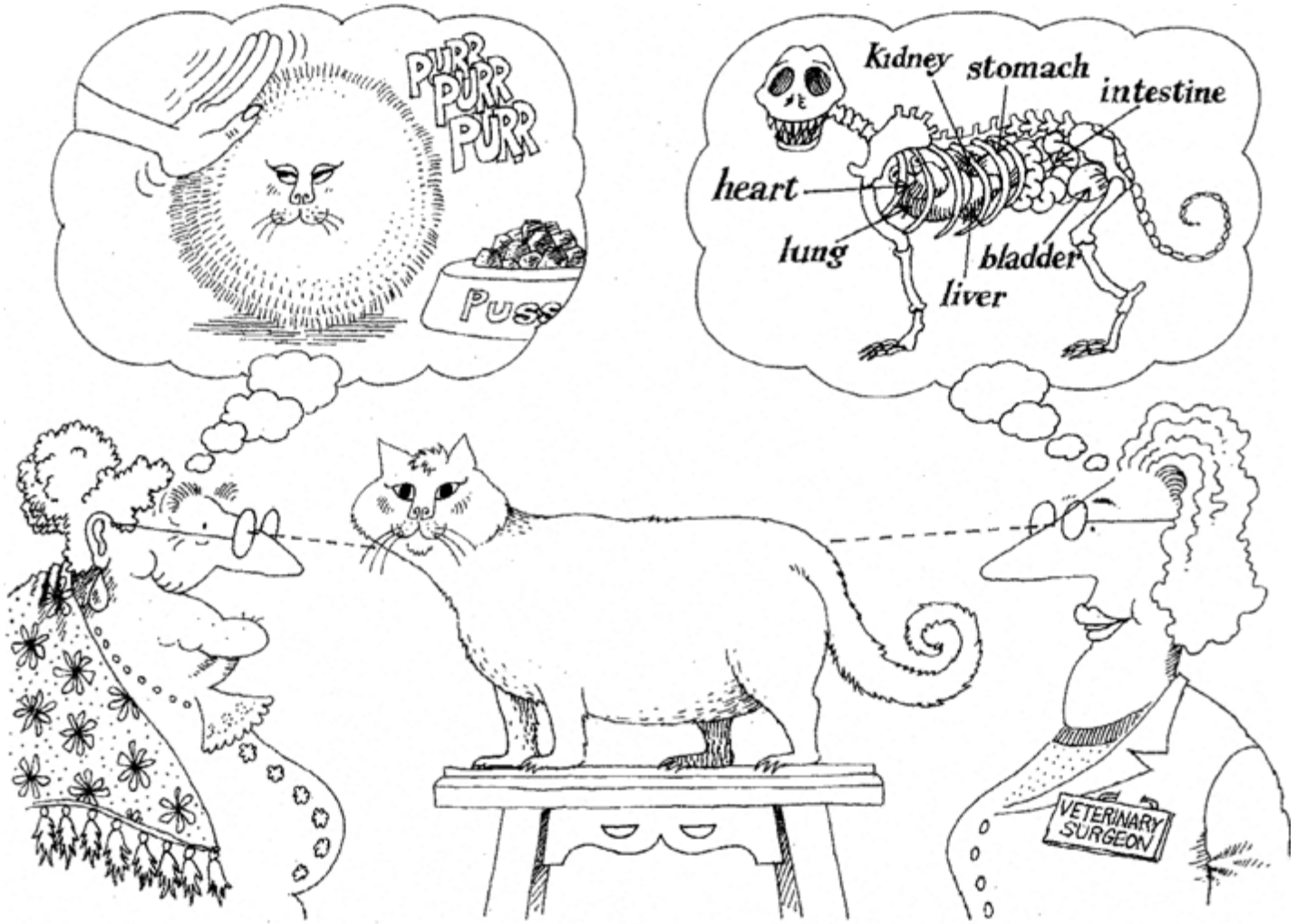
G. Booch at al., *Object-oriented analysis and design with applications*, 3rd edition, Addison-Wesley, 2007.

Principi OO modela

- Osnovni :
  - apstrakcija
  - kapsulacija
  - modularnost
  - hijerarhija
- Dodatni:
  - tipizacija
  - konkurentnost
  - perzistencija

Apstrakcija

- Apstakcija je uprošćeni opis sistema kojim se naglašavaju samo neke osobine ili detalji.



Apstrakcija

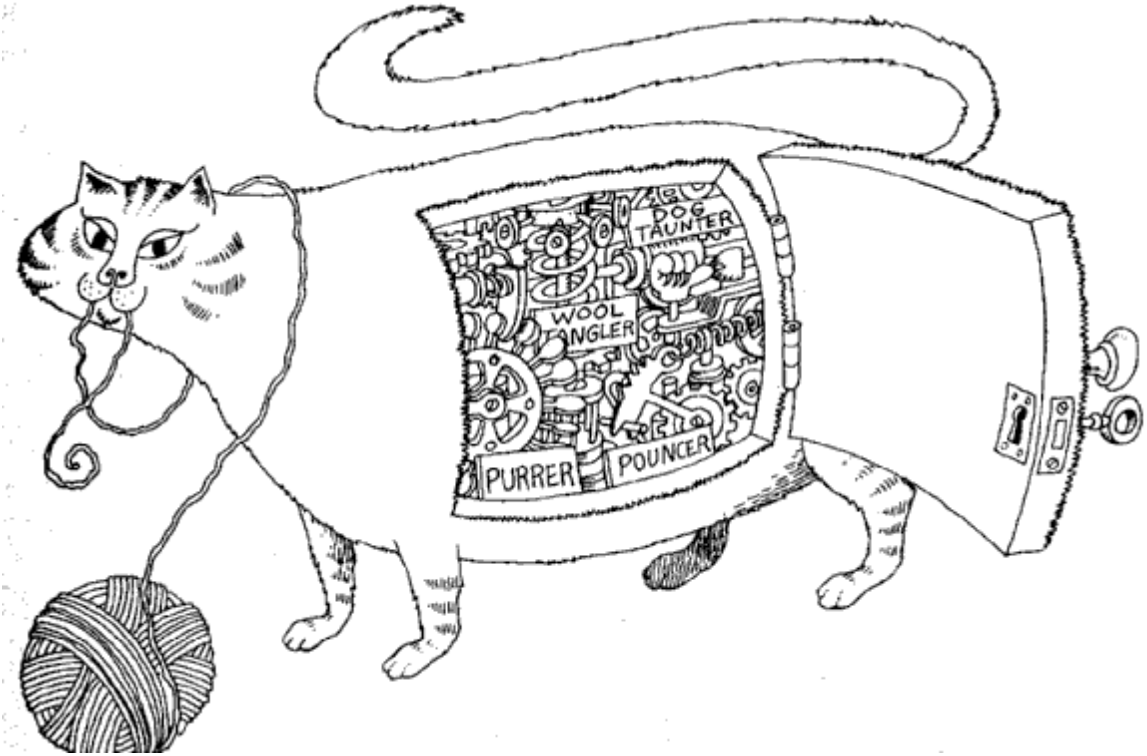
- (Booch) Apstrakcija je postupak kojim se ističu bitne karakteristike objekta (koje ga razlikuju od objekata drugih vrsta) i time definišu konceptualne granice (iz perspektive posmatrača).
- Apstrakcija:
  - Entiteta
  - Akcije

Abstraction: Temperature Sensor
Important Characteristics: temperature location
Responsibilities: report current temperature calibrate

Abstraction: Active Temperature Sensor
Important Characteristics: temperature location setpoint
Responsibilities: report current temperature calibrate establish setpoint

Kapsulacija

- Proces skrivanja onih elemenata apstrakcije koji definišu strukturu i ponašanje.
- Služi da razdvoji konceptualni interfejs od implementacije apstrakcije.



Kapsulacija

- Interfejs apstrakcije čini ono što je od nje dostupno spolja.

Abstraction: Heater
Important Characteristics: location status
Responsibilities: turn on turn off provide status

Interfejs apstrakcije Heater predstavlja sve što je klijentu/korisniku potrebno da zna o klasi Heater.

```
class Heater {
public Heater(location)
       void turnOn()
       void turnOff()
       boolean isOn()
private ....
};
```

Modularnost

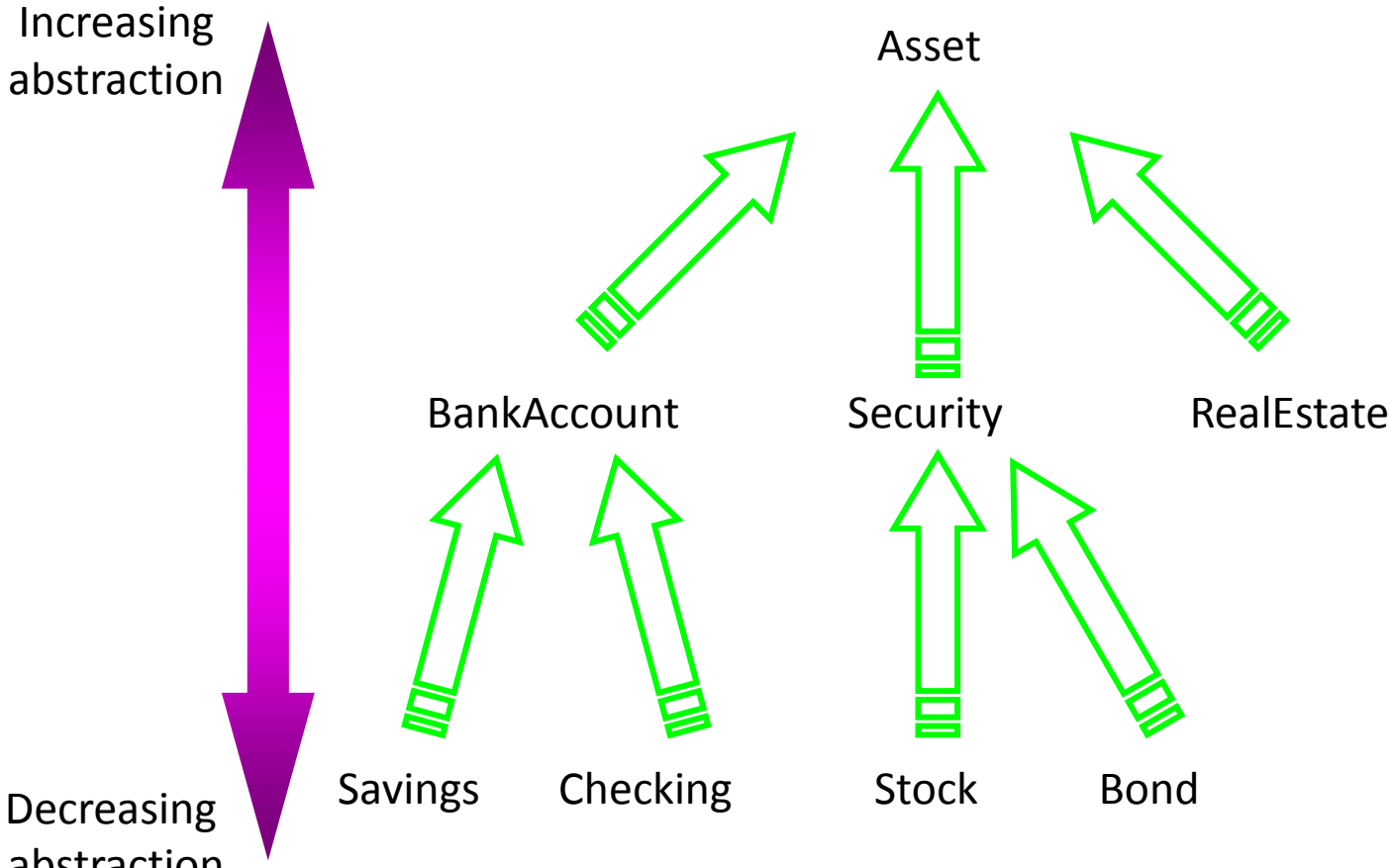
- Sistem koji se razlaže na skup kohezivnih i slabo spregnutnih modula se naziva **modularnim**.
- Moduli su fizičke jedinice (nezavisno se prevode) koje predtavljaju komponenete sistema i mogu se održavati nezavisno.

Modularnost podrazumeva pakovanje apstakcija u manje jedinice

Hijerarhija

- Hijerarhija je uređivanje apstrakcija.
- Nasleđivanje – hijerahija određena ***is-a*** relacijom
- Sadržanje – hijerahija određena ***part-of*** relacijom (agregacija/kompozicija)
- Klasifikacija, generalizacija, specijalizacija
  - Klasifikacija je proces određivanja i svrstavanja objekata sa istim karakteristika u klase.
  - Generalizacija je postupak pronalaženja sličnosti između klasa/apstrakcija i definisanja sličnosti u novoj generalizovanoj klasi.
  - Specijalizacija proces određivanja razlika među objektima jedne klase i definisanja novih potklasa koje sadrže razlike.

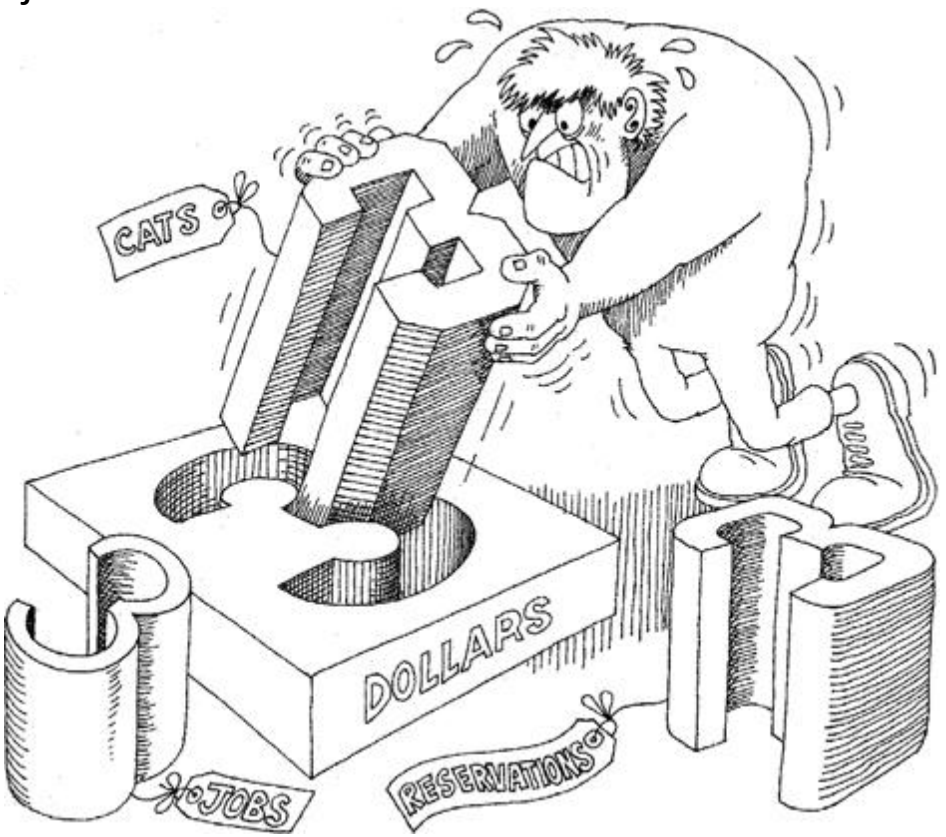
Hijerarhija



Elements at the same level of the hierarchy should be at the same level of abstraction

Tipizacija

- Tipizacija se odnosi na nemogućnost (ili ograničenu mogućnost) da se objekti različitih klasa međusobno razmenjivati.
- Statička i dimanička tipizacija – vreme kada se ime (promenljive) vezuje za tip.
- Polimorfizam – osobina da se objekat kojem se pristupa kao objektu osnovne klase ponaša različito.



Konkurentnost i perzistencija

- Nisu obavezni da bi softver bio kvalifikovan kao OO.
- **Konkurentnost** – moguće je definisanje aktivnih i pasivnih objekata. Aktivni objekat ima svoju nit kontrole.
- **Perzistencija** – postojanje objekta se proteže kroz
  - Vreme – nastavlja da postoji nakon nestanka njegovog stvaraoca
  - Prostor – moguće premeštanje iz adresnog prostora u kom je stvoren.