Objektno orijentisano programiranje (OOP)

- Koncept koji u razvoju aplikacija i računarskih programa koristi "objekte" i njihove interakcije. Baziran je na nekoliko tehnika koje uključuju ***encapsulation***, ***modularity***, ***polymorphism***, and ***inheritance***. Kao osnovni programski koncept u razvoju softvera nije šire korišćen sve do početka 1990-ih. Međutim, danas ovaj koncept podržavaju mnogi savremeni programski jezici
- Koreni OOP sežu u 1960-e kada hardver i softver postaju složeniji a ovaj pristup se razvija sa ciljem diskretizacije programskih blokova - koda, odnosno same logike programiranja, kao i omogućenje ponovnog korišćenja već napisanog koda.
- **Simula** je prvi programski jezik koji uvodi ovaj koncept (objekti, klase, subklase, virtualni metodi, korutine, diskretna simulacija) i to kao nadskup Algol-a. Korišćena je za fizičko modeliranje korišćeno u analizi kretanja brodova i utovarenog tereta kroz teretne luke.
- OOP se može posmatrati kao skup kooperirajućih objekata, nasuprot tradicionalnom pristupu koji se može razumeti kao grupa zadataka koja se izvršava (podprograma). U okviru OOP svaki objekat sposoban je da prima poruke, obrađuje podatke i da šalje poruke drugim objektima.
- Objekat se može posmatrati kao nezavisan uređaj (mađina) sa posebnom nezavisnom ulogom i namenom. Akcije ili operatori na objektima su tesno povezani sa samim objektom. Strukture podataka teže da imaju sopstvene operatore, ili ih u najmanju ruku "nasleđuju" od sličnih objekata ili klasa. Tradicionalni pristup, pak, posmatra podatke i ponašanje odvojeno.

Osnovni koncepti objektnih modela

• **KLASE** (classes)	– abstraktni tipovi podataka
• **OBJEKTI** (objects)	– predstavljaju abstraktne strukture podataka
• **METODI** (methods)	– predstavljaju mogućnosti (moguća delovanja)
• **PORUKE** (message passing)	– proces slanja podataka drugom objektu
• **NASLEDJIVANJE** (inheritance)	– nasleđivanje atributa i ponašanja od klase roditelja
• **ENKAPSULACIJA** (encapsulation)	– skrivanje funkcionalnih detalja klase od objekata koji šalju poruke
• **APSTRAKCIJA** (abstraction)	– pojednostavljenje modeliranjem klase prilagođene problemu
• **POLIMORFIZAM** (polymorphism)	– sposobnost objekata da na istu akciju odgovore na različit način

- **OBJEKAT** je poseban entitet[1] koji se može posebno identifikovati, koji sadrži skup operacija i sposoban je da beleži podatke o efektima operacija na stanje. Otuda, objekt je u suštini isto što i apstraktna struktura podataka. Objekat se karakteriše preko:
  - ✓ stanja,
  - ✓ operacija,
  - ✓ identiteta.

  Takodje, objekti se često predstavaljaju i kao ***apstraktne strukture*** koje sadrže
  - ✓ parametre,
  - ✓ atribute,
  - ✓ metode (procedure) i koji «žive» u modelu nakon kreiranja

[1] U ovom slučaju koristi se opšte značenje pojma "*entitet*", u smislu «*bilo kojeg objekta ili komponente sistema, koji zahteva eksplicitnu predstavu unutar modela*»

Interakcija objekata podrazumeva *direktni pristup parametrima i atributima*, *mogućnost pozivanja metoda (procedura)* istog ili drugih objekata i *sinhronizaciju «življenja»* sa drugim objektima u modelu.

Stanje objekta nije ništa drugo do skup informacija koje objekt čuva, a te se informacije mogu menjati u vremenu kao posledica operacija nad objektom, ali se ne mogu menjati spontano. Različite komponente stanja ponekad se nazivaju atributima objekta.

Operacija označava proceduru koja preuzima ili zadaje stanja objekta i/ili vrednost jednog ili više atributa objekta.

Identitet je u stvari način da se objekti razlikuju čak i onda kada su istog tipa, u istom stanju i sadrže iste operacije.

Programski kod objekta pisan u skriptu simulacionog alata MODSIM III.

```
TYPE
   movingObj = OBJECT
   x, y : REAL;
   ASK METHOD ObjInit;
   ASK METHOD stop;
END OBJECT {movingObj};
```

```
OBJECT movingObj;
   ASK METHOD ObjInit;
   BEGIN
      ... programski kod za inicijalizaciju objekta
   END METHOD {ObjInit};

   ASK METHOD stop;
   BEGIN
   ... programski kod za zaustavljanje objekta
   END METHOD {stop};
END OBJECT {movingObj};
```

OSNOVNA KARAKTERISTIKA PROGRAMSKIH JEZIKA OBJEKTNE SIMULACIJE JE DA PODRŽAVAJU KONCEPT OBJEKTA KAO OSOBINU JEZIKA.

- **KLASA** predstavlja "obrazac", ("matricu", "kalup" ili "šablon"), za kreiranje objekata. Tako objekti koji pripadaju istoj klasi sadrže iste operacije, imaju ista moguća stanja i atribute. Pojedinačni procesi predstavljaju se pojedinačnim objektima odgovarajuće klase pa se karakteristike procesa predstavljaju atributima klase.

  *OBJEKTNI JEZIK JE **ZASNOVAN NA KLASAMA** AKO JE TAJ KONCEPT UGRADJEN U OSOBINE JEZIKA, I AKO SVAKI OBJEKT PRIPADA NEKOJ KLASI.*

- **METOD** predstavlja mogućnost ili sposobnost objekta da "nešto uradi", "sprovede u delo"
- **PORUKA** predstavlja način komunikacije objekata
- **NASLEDJIVANJE** označava osobinu objektnih jezika koja omogućuje prenošenje atributa klase i/ili operacija na objekte izvedene iz te klase.

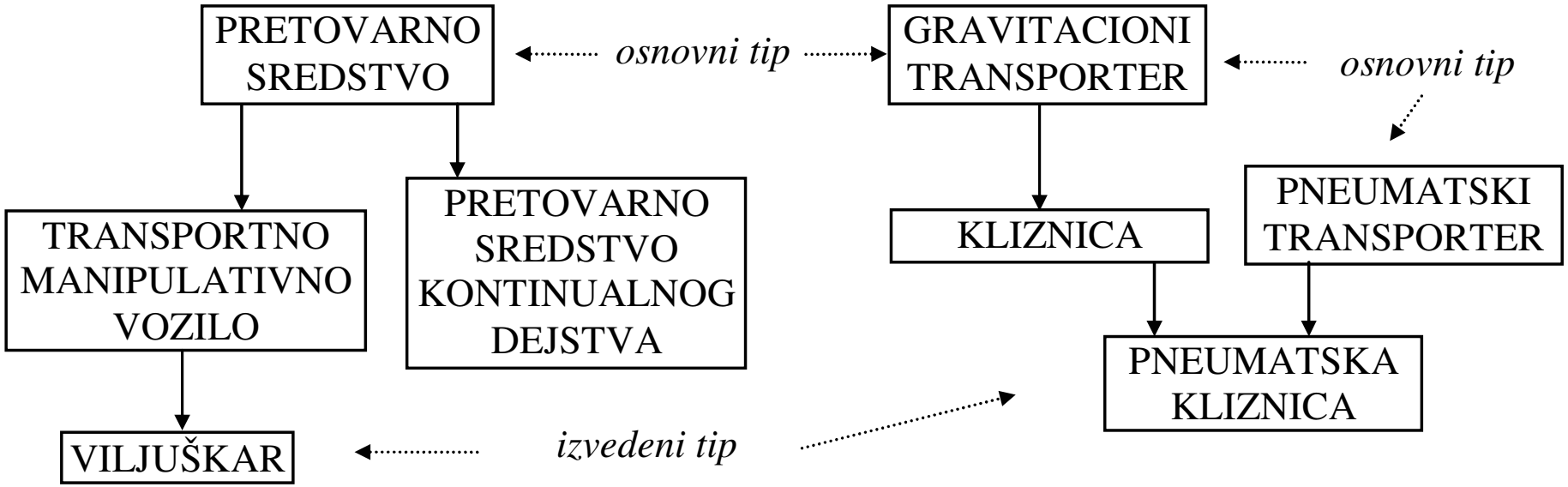
  Značaj nasledjivanja je u stvaranju uslova za ponovno korišćenje programskog koda jer se stanja i operacije definišu u osnovnoj klasi, a potom koriste u podklasama.

  Klasa B nasledjuje klasu A ako objekti B predstavljaju podskup objekata A. Klasa B podržavaće sve operacije klase A, mada one mogu biti izvedene na drugačiji način, a klasa B, takodje, može podržavati i dodatne operacije koje ne postoje u A.

  Kaže se i da je klasa B podklasa (*subclass*), dete (*child*) ili izvedena klasa (*derived class*), dok je A superklasa (*superclass*), roditelj (*parent*) ili osnovna klasa (*base class*).

Koncept nasledjivanja podrazumeva hijerarhijski odnos u kome se atributi klasa i operacije prenose sa objekata na višem hijerarhijskom nivou na objekte nižeg nivoa.

Neki od programskih jezika podržavaju višestruko nasledjivanje, koje podrazumeva da se na izvedenu klasu prenose atributi i operacije sa više od jedne superklase ili roditelja, a prosto, ili jednostruko, nasledjivanje znači da se atributi izvedene klase nasledjuju samo od jedne superklase



- **POLIMORFIZAM** (mnoštvo oblika) označava sposobnost realizovanja različitih akcija kao rezultat delovanja iste pobude.

  Praktično, ova mogućnost znači da za slučaj upućivanja iste poruke različitim objektima, oni mogu ispoljavati različito ponašanje.

  Na primer, ukoliko se radi o pretovarnom sistemu u kome rade gasni, dizel i elektroviljuškar, predstavljeni odgovarajućim objektima koji sadrže proceduru za dopunu energije, jedna ista poruka "dopuni energiju" u slučaju gasnog viljuškara značiće dopunu rezervoara sa tečnim naftnim gasom, kod dizel viljuškara dopunu dizel goriva, a kod elektroviljuškara punjenje baterije.

Iako je reč o savremenom konceptu, primena objektno orijentisanog programiranja u simulaciji nije novost, jer jos početkom 70-ih simulacioni programski jezik SIMULA koristi ovaj pristup. Šta više, jednu od karakteristika objektnog pristupa - "NASLEDJIVANJE" poseduje prvi specijalizovani jezik baziran na procesno interaktivnom pristupu - GPSS, danas svakako i jedan od najpoznatijih jezika diskretne simulacije, čija se prva IBM verzija pojavila daleke 1961. godine. Naime, SPLIT komanda u GPSS-u kreira specifikovani broj kopija procesa (transakcija), sa identičnom strukturom podataka i procedura, pri čemu svaka transakcija ponaosob može biti numerisana i tako imati sopstveni identitet.

OBJEKTI (Moduli Arene i Flexsim-a)