Java

Šta je Java?

- Java je objektno orijentisani programski jezik
- Java kod se kompajlira u byte-code koji se izvršava u Java virtuelnoj mašini (JVM)
- Java se intenzivno koristi za razvoj serverskih aplikacija (backend aplikacija), kojima se može pristupati korišćenjem klijentskih aplikacija razvijenih u Javi ili bilo kom drugom programskom jeziku

OOP principi

Objektno orijentisano programiranje (OOP)?

- Programski model u kome se dizajn aplikacije koncentriše na model podataka, njihove međusobne veze i funkcionalnosti koje mogu da obave, a ne na same funkcije

- Podaci se reprezentuju objektima

- Objekti predstavljaju instance definisanih tipova - ***klasa***, koje opisuju osobine i ponašanje objekta

Osnovni principi OOP

- Enkapsulacija
- Apstrakcija
- Nasleđivanje
- Polimorfizam

Osnovni principi OOP

- Enkapsulacija
  - ideja da se podacima u objektu pristupa i manipuliše isključivo putem javnog interfejsa - metoda objekta
  - možemo da kažemo da sam objekat “zna” šta i kako može da se radi sa njegovim podacima
  - podaci su najčešće zaštićeni (Java, C++, C# štite privatne podatke, dok Python na primer na nivou jezika nema striktnu zaštitu, ali postoji konvencija naziva koja ovo obezbeđuje delimično)

Osnovni principi OOP

- Zašto je ova ideja enkapsulacija dobra?
  - funkcionalnost (nečega) je definisana na jednom mestu, a ne rasuta u kodu
  - ta funkcionalnost je definisana na logičnom mestu, tamo gde se nalaze podaci sa kojom ta funkcionalnost treba da nešto i radi
  - podaci u objektu se ne menjaju neočekivano i nepredviđeno spolja
  - kada se poziva metod treba da znamo objekat čiji metod pozivamo, šta metod očekuje od ulaznih parametara i šta nam vraća - ne moramo da znamo KAKO radi

Osnovni principi OOP

- Apstrakcija
  - usko povezana sa enkapsulacijom
  - procesom apstrakcije definišemo koje osobine objekata iz realnog sveta su nam bitne
  - definišemo klase, osobine objekata, tipove podataka i njihovo ponašanje - programski ovo predstavlja definiciju klase - praktično novog tipa podataka.
  - Kada se napravi jedan primerak klase dobija se objekat (programski) sa kojim može da se radi.

Osnovni principi OOP

- Nasleđivanje
  - način da se iskoristi već postojeći kod, bez potrebe da se sve kopira
  - omogućava da se definišu podtipovi, posebne klase koje ispoljavaju zajedničke osobine sa roditeljskom klasom (klasama), ali i posebnosti koje roditelji nemaju
  - hijerarhije klasa

Osnovni principi OOP

- Polimorfizam
  - “pojava u više oblika”
  - jedno ime više oblika
  - manifestuje se tako što je moguće imati više metoda sa istim nazivom, ali različitim ponašanjem
  - overriding - polimorfizam za vreme izvršavanja - tokom izvšavanja se utvrđuje koja metoda će biti izvršena. Odlučuje se na osnovu trenutnih svojstava objekta
  - overloading - polimorfizam koji se razrešava za vreme kompajliranja, tada se povezuje koja metoda će biti izvršavana

Osnovni principi OOP

- Logika izvršavanja aplikacije se poštovanjem prethodnih principa svodi na upravljanje razmenom poruka (poziva dostupnih metoda) između objekata koji su kreirani tokom životnog ciklusa aplikacije

Nasleđivanje i drugi tipovi veza između objekata

Šta je nasleđivanje

- Jedan od fundamentalnih koncepata OOP
- Omogućava da se izbegne copy-paste kodiranje
- Ponovna upotrebljivost koda
- Omogućava da se nova klasa izvede iz postojeće, i proširi definiciju (dodavanjem novih atributa ili novih metoda ili redefinisanjem roditeljske metode)
- Na ovaj način podaci i ponašanje koje su zajednički i za osnovnu i za izvedenu klasu ne moraju se ponovo definisati

Osnovno nasleđivanje

- U Javi se sve klase implicitno (bez posebnog navođenja) nasleđuju iz klase ***Object***

- Ako prilikom definisanja klase nije navedena klasa iz koje se nasleđuje onda je to klasa Object

O čemu voditi računa pri nasleđivanju

- Izvedena klasa nikada ne implementira **manje** funkcionalnosti nego osnovna

- Nasleđivanje ne bi trebalo da je preduboko

- Nasleđivanje koristimo da izbegnemo redundantnost koda

- Nasleđivanje se koristi tamo gde je to prirodna veza između klasa, gde objekat nasleđene klase uvek JESTE i primerak roditeljske klase

    - U drugim situacijama koristi se asocijacija ili kompozicija, koje opisuju veze između objekata različitih klasa

Asocijacija i kompozicija

- Opisuju veze koje se uspostavljaju između objekata različitih klasa
- **Asocijacija** opisuje vezu kada jedan objekat koristi neki drugi.
  Primer: objekat klase Student može kao svoj podatak član sadržati listu objekata Knjiga koji predstavljaju knjige pozajmljene iz biblioteke.
- **Kompozicija** opisuje vezu celina-delovi, objekat koji predstavlja celinu, kao svoje podatke članove sadrži objekte drugih klasa. On bez njih bi ne bio kompletan. Istovremeno, iako je moguće kreirati nezavisno objekte tih drugih klasa (delova), njihova upotrebljivost se u potpunosti ispoljava samo kada su deo celine. Primer klasa Automobil se sastoji od delova Motor, Menjač, Točak…
- Ovakve veze je neprirodno (a i pogrešno) realizovati preko nasleđivanja (niti je student knjiga, niti je knjiga student, kao što ni Automobil nije Motor…)