Advanced Java Programming

1. Napredno objektno programiranje

1. Objektno orijentisano programiranje u Javi

- Objektno orijentisano programiranje omogućava apstrahovanje modela zasnovanih na pojmovima iz realnog sveta.
- Klase predstavljaju šablone za izgradnju objekata.
- Osnovni postulati objekto orijentisanog programiranja su: apstrakcija, enkapsulacija, nasleđivanje i polimorfizam.
- Apstrakcija omogućava da se složeni entiteti iz realnog sveta modeluju sa nivoom detalja koji odgovara potrebama programske logike.
- Enkapsulacija se odnosi na postupak objedinjavanja informacija i funkcionalnosti unutar jedne celine – objekta i na kontrolu vidljivosti takvih klasnih članova izvan same klase.
- Nasleđivanje omogućava kreiranje objekata koji osobine i ponašanja nasleđuju od nekog drugog objekta.
- Jedna klasa može imati samo jednu roditeljsku klasu
- Nasleđivanje se postiže upotrebom ključne reči extends.
- Pristup objektu roditeljske klase unutar klase koja nasleđuje moguće je obaviti upotrebom ključne reči super.
- Ukoliko se konstruktor ne definiše, Java izvršno okruženje samostalno kreira podrazumevani konstruktor.
- Podrazumevani konstruktor nema parametara, a unutar njega se obavlja pozivanje konstruktora roditeljske klase.
- Polimorfizam se odnosni na višestruko značenje jednog istog pojma, što se često koristi za kreiranje više metoda istog naziva, ali nešto drugačije logike.
- Overriding predstavlja redefinisanje metoda u nasleđenim klasama.
- Overloading predstavlja preklapanje metoda u okviru jedne iste klase.

2. Apstraktne klase i interfejsi

- Apstraktna klasa služi kao obrazac, odnosno smernica koja diktira šta sve treba da sadrže klase koje je nasleđuju.
- Apstraktne klase se ne mogu instancirati.
- Deklarisanje klase kao apstraktne postiže se upotrebom ključne reči abstract.
- Apstraktne klase mogu da sadrže apstraktne metode.
- Apstraktne metode su metode koje imaju samo potpis i koje moraju da budu implementirane u nasleđenim klasama.
- U objektno orijentisanom programiranju, interfejs je dogovor o tome kakva ponašanja jedna klasa mora pružiti.
- Interfejsi primarno sadrže samo javne, apstraktne metode, ali pored njih mogu imati i konkretne javne i privatne metode i konstante.
- Interfejs se definiše ključnom rečju interface.
- Da bi neka klasa implementirala interfejs, koristi se ključna reč implements.
- Klasa koja implementira interfejs mora da obezbedi implementaciju svih apstraktnih metoda koje postoje u interfejsu.
- Interfejsi su referentni tipovi.

3. Klase unutar klasa

- Klase koje su deklarisane samostalno, izvan bilo koje druge klase, drugačije se nazivaju top-level klase (top-level classes).
- U programskom jeziku Java moguće je napraviti klasu koja se nalazi u okviru druge klase; takve klase se nazivaju ugnežđene klase.
- Statička ugnežđena klasa predstavlja člana klase u kojoj je definisana, i ona može biti instancirana bez prethodnog instanciranja matične klase; ovakve klase nemaju pristup objektnim članovima klase u kojoj su definisane.
- Nestatičke ugnežđene klase predstavljaju objektne članove, odnosno postoje samo u okviru objekata klasa u kojima se

nalaze; ovakve klase imaju pristup objektnim članovima klase u kojoj se nalaze.
- Lokalne klase su klase koje su definisane u okviru bloka; najčešće se ove klase nalaze u okviru tela metoda.
- Anonimne klase su klase bez imena; one omogućavaju da deklarisanje i instanciranje bude obavljeno u okviru jedne naredbe (izraza).
- Lambda izrazi predstavljaju kompaktnu sintaksu za kreiranje instance klasa koje implementiraju funkcionalni interfejs, odnosno interfejs sa jednom apstraktnom metodom.
- Referenca na metodu omogućava da se lambda izraz dodatno optimizuje kada implementacija metode iz funkcionalnog interfejsa podrazumeva isključivo poziv neke druge metode.

4. Događaji

- Programiranje koje podrazumeva praćenje i obradu događaja u toku izvršavanja aplikacije naziva se event-based
- Rukovanje događajima podrazumeva sledeće činioce: objekat događaja, generator događaja i slušaoce događaja.
- Generator događaja je objekat koji proizvodi događaj.
- Objekat događaja sadrži informacije o događaju – referencu na objekat unutar koga je došlo do događaja i razne druge, opcione podatke.
- Slušaoci događaja su objekti koji bivaju obavešteni o pojavi događaja.
- Proces kojim neki objekat postaje slušalac često se naziva pretplata na događaj, a sami slušaoci – pretplatnici.
- Osnovna klasa kojom se predstavljaju događaji u Javi je EventObject.
- Slušaoci događaja najčešće se predstavljaju korišćenjem interfejsa, koji moraju da implementiraju sve klase čiji objekti treba da budu obavešteni o pojavi nekog događaja.
- Slušaoci se unutar klase generatora čuvaju unutar niza ili kolekcije, pri čemu klasa generatora poseduje i metode za dodavanje i uklanjanje slušalaca.
- Emitovanje događaja podrazumeva prolazak kroz niz (listu) slušalaca i pozivanje metode koja je definisana interfejsom slušaoca.

5. Refleksija

- Mogućnost programskog jezika da u toku izvršavanja programa manipuliše arhitekturom klasa naziva se refleksija.
- Korišćenjem refleksije moguće je obaviti instanciranje tipova i pozivanje metoda.
- U programskom jeziku Java, funkcionalnost refleksije nalazi se u paketu java.lang.reflect.
- Klasa Class se koristi za reprezentovanje konkretnih klasa.
- Klasa Field predstavlja polja klase, klasa Method predstavlja metode, dok klasa Parametar predstavlja parametre metoda.
- U svetu refleksije konstruktori se predstavljaju klasom Constructor, a referenca na sve konstruktore jedne klase dobija se pozivanjem metode getConstructors().
- Refleksija je veoma korisna kada je podatke iz jednog oblika potrebno pretvoriti u neki drugi, prilikom manipulacije podacima baze podataka ili korišćenja JSON formata.

6. Dizajn šabloni

- Softverski dizajn šabloni opisuju rešenja veoma čestih problema do koji dolazi prilikom dizajniranja unutrašnje strukture softverskih proizvoda.
- Dizajn šabloni nisu gotova rešenja, već samo generalizovana, koncizna uputstva koja se mogu koristiti za konkretnu implementaciju rešenja.
- Svi softverski dizajn šabloni se dele na tri grupe: na šablone kreiranja, strukturne šablone i šablone ponašanja.
- Singleton je dizajn šablon koji propisuje postojanje samo jedne instance nekog tipa.

- Observer omogućava da se stanje jednog objekta emituje nekim drugim objektima koji su zainteresovani za promene unutrašnjeg stanja takvog objekta.
- Factory šablon se koristi kada je potrebno kreirati posredničku klasu koja će služiti za centralizovano kreiranje objekata.
- Decorator je strukturni šablon koji omogućava izmenu funkcionalnosti postojeće klase, pri čemu Decorator klasa nije u direktnoj hijerarhijskoj vezi sa klasom koju dekoriše.
- JavaBeans je konvencija koja definiše klase koje implementiraju interfejs Serializable i poseduju javni konstruktor bez parametara i privatna polja sa odgovarajućim get i set

2. Strukture podataka

7. Enumeracije

- Enumeracije su liste imenovanih konstanti koje definišu novi tip podatka, odnosno, enumeracija je složeni tip podatka koji se sastoji iz fiksnog broja konstanti.
- Enumeracija se kreira korišćenjem ključne reči enum.
- Enumeracije su korisne uvek kada je potrebno definisati set vrednosti koje predstavljaju kolekciju konstanti za koje znamo da se neće menjati.
- Najznačajniji elementi koji se mogu naći unutar enumeracija jesu konstante, koje se drugačije nazivaju enumeracione konstante.
- U programskom jeziku Java, enumeracije su specijalna vrsta klasa koje se u pozadini predstavljaju klasom Enum.
- Svaka enumeracija automatski poseduje nekoliko metoda: values(), valueOf(), ordinal().
- Svaka enumeraciona konstanta jeste objekat koji je tipa enumeracije i podrazumevano poseduje svojstva name i ordinal.
- Najznačajnija upotrebna vrednost enumeracija jeste njihovo kombinovanje sa uslovnim konstrukcijama.
- Unutar enumeracija se mogu definisati proizvoljna polja i metode.

8. Kolekcije

- Kolekcije su tipovi podataka koji sadrže više vrednosti jednog ili različitih tipova.
- Pored toga što služe za smeštanje podataka, kolekcije pružaju mogućnost i za pristupanje podacima i manipulaciju podacima.
- Tipovi kojima se predstavljaju Java kolekcije nalaze se unutar paketa java.util, koji predstavlja jedan od osnovnih ugrađenih paketa u programskom jeziku Java.
- Najznačajniji interfejsi kojima se definišu kolekcije u Javi su Iterable, Collection, List, Set, Map, Queue...
- Najznačajniji tipovi kolekcija u zavisnosti od njihovih osobina su: liste, stekovi, setovi, redovi i mape.
- Najčešće korišćena lista u programskom jeziku Java je ArrayList.
- Stack je klasa kojom je moguće kreirati kolekcije koje se zasnivaju na postulatima LIFO strukture podataka.
- LinkedList je klasa koja omogućava kreiranje kolekcija čiji elementi poštuju FIFO ustrojstvo.
- HashSet je klasa za kreiranje kolekcija sa jedinstvenim elementima bez posebnog redosleda.
- HashMap je implementacija kolekcije parova ključeva i vrednosti.

9. Generički tipovi podataka

- Pojam generički odnosi se na nešto što nije unapred specifično određeno, odnosno na nešto što može imati široki spektar različitih značenja.
- U programiranju, koncept generičkih tipova podataka omogućava kreiranje tipova podataka koji mogu biti parametrizovani.
- Rukovanje generičkim tipovima predstavlja najviši stepen apstrakcije u objektno orijentisanom programiranju.
- Parametrizacija generičkih tipova obavlja se navođenjem naziva tipa unutar trouglastih zagrada.
- Generički tipovi mogu rukovati samo referentnim tipovima.
- Generički tipovi se mogu koristiti i bez parametrizacije i tada je podrazumevani tip generičkih parametara Object.
- Diamond operator (<>) uklanja potrebu za eksplicitnim navođenjem tipa argumenta u pozivu konstruktora generičkih tipova.
- Wildcard se označava karakterom znak pitanja (?) i reprezentuje nepoznati tip.

10. Tokovi

- Tokovi su sekvence objekata nad kojima je moguće pozivati različite metode za obradu podataka kako bi se proizveo željeni rezultat.
- Kolekcije su zadužene za čuvanje podataka, a tokovi za sprovođenje operacija nad njima.
- Tokovi se gotovo uvek zasnivaju na nekom zasebnom izvoru podataka, koji može biti niz, kolekcija ili neko eksterno skladište iz koga se podaci uvoze u aplikaciju.
- Skup operacija koje se mogu izvršiti nad jednim tokom drugačije se naziva Stream Pipeline.
- Sve operacije nad tokovima mogu biti posredne ili terminalne.
- Svaka posredna, odnosno međuoperacija nad tokom kao svoju povratnu vrednost ima novi tok.
- Terminalnom operacijom završava se obrada jednog toka i dolazi do emitovanja rezultata obrade.
- Operacije nad tokom uvek se završavaju finalnom, odnosno terminalnom operacijom.
- Kroz jedan tok se može proći samo jednom, i to uvek unapred.
- Tokovi omogućavaju da se nad podacima na veoma lak način obave neke uobičajene operacije: iteracija, filtriranje, sortiranje, uklanjanje duplikata, redukcija...

3. Tekst, datumi i vreme

11. Napredni rad sa tekstom

- U programskom jeziku Java, tekstualni podaci se predstavljaju korišćenjem tipa String.
- String-ovi se mogu kreirati kao da su vrednosti prostih tipova.
- String-ovi se mogu nadovezivati korišćenjem operatora sabiranja, metode concat() ili StringBuilder klase.
- String-ovi su nepromenljivi (immutable).
- Klasa String poseduje veliki broj metoda koje je moguće koristiti za manipulaciju tekstualnim vrednostima.
- Metoda isBlank() omogućava nam da proverimo da li određena promenljiva poseduje neku konkretnu tekstualnu vrednost koja ne predstavlja prazne karaktere.
- Podelu jedne String vrednosti na linije moguće je obaviti korišćenjem metode lines().
- Korišćenjem metode indent() moguće je uvući redove teksta za određeni broj praznih mesta.
- Ukoliko je jednu String vrednost potrebno ponoviti veći broj puta, odnosno jedan String nadovezati proizvoljan broj puta na samog sebe, moguće je koristiti metodu repeat().
- Text blocks je funkcionalnost jezika koja omogućava veoma lako definisanje višelinijskog teksta, a karakteriše se upotrebom tri karaktera navodnika na početku i na kraju teksta.
- Regularni izrazi omogućavaju kreiranje šablona na osnovu kojih se mogu definisati kriterijumi pretrage prilikom rada sa tekstualnim vrednostima.
- Regularni izrazi se mogu koristiti u kombinaciji sa klasama Pattern i Matcher i nekoliko metoda koje se nalaze unutar klase String.
- Za uspešno korišćenje regularnih izraza, neophodno je savladati sintaksu za njihovo kreiranje.

12. Datumi i vreme

- Rukovanje vremenom i datumom predstavlja jedan od osnovnih koncepata bilo kog programskog jezika.
- Rukovanje datumom i vremenom korišćenjem Java platforme omogućeno je kroz poseban skup funkcionalnosti koji se naziva Date and Time API.
- Date and Time API nalazi se unutar paketa java.time.
- U Javi postoje dva formata za predstavljanje vremena: mašinski i svima nama poznati,
- Za rukovanje datumom koristiti se klasa LocalDate.
- Za rukovanje vremenom koristi se klasa LocalTime.
- Za objedinjeno rukovanje datumom i vremenom koristi se klasa LocalDateTime.
- Klasom Period predstavlja se period između dve tačke u vremenu.
- Za formatiranje i parsiranje datuma i vremena koristi se klasa DateTimeFormatter.

4. Rad sa podacima

13. Ulazno/izlazni tokovi

- Ulazno/izlazni tokovi su osnovni mehanizmi koji omogućavaju da jedan Java program dobije neke podatke iz spoljašnjeg sveta, ali i da podatke koji postoje unutar programa emituje spoljašnjem svetu.
- Ulazni tokovi se koriste kako bi Java program pročitao neke podatke, a izlazni kako bi podatke poslao, odnosno upisao.
- Bez obzira na to kojeg su tipa i na koji način funkcionišu, svi tokovi za jedan program predstavljaju isto: jednu sekvencu podataka.
- Tokovi koji rukuju podacima u 8-bitnom obliku drugačije se nazivaju bajt tokovi.
- Tokovi karaktera, za razliku od bajt tokova, manipulišu direktno karakterima.
- Baferovani tokovi koriste bafer, odnosno specijalan memorijski prostor koji se ponaša kao posrednik između izvorišta ili odredišta i Java programa, čime se ubrzava i pojednostavljuje čitanje i upisivanje podataka.
- Tokovi koji omogućavaju direktno rukovanje podacima prostih tipova, uključujući i objektni tip String, nazivaju se tokovi podataka (data streams).
- Sve Java aplikacije poseduju standardne, unapred dostupne objekte koji predstavljaju ulazne i izlazne tokove koji omogućavaju interakciju sa konzolom; reč je o tokovima dostupnim korišćenjem svojstava.in i System.out.
- Klasa Scanner omogućava nešto udobnije čitanje podataka sa konzole, uvođenjem mogućnosti automatske detekcije tipova i konvertovanja vrednosti.

14. Serijalizacija

- Serijalizacija se odnosi na koncept konvertovanja objekata koji se nalaze unutar nekog programa u oblik koji se lako može sačuvati ili proslediti.
- Kako bi se podaci koji su serijalizovani ponovo vratili u svoj izvorni, objektni oblik, pribegava se procesu deserijalizacije.
- Serijalizacija i deserijalizacija često se drugačije nazivaju marshalling i unmarshalling, respektivno.
- Postoje dve najčešće korišćene vrste serijalizacije: binarna i tekstualna.
- Binarna serijalizacija podrazumeva smeštanje sadržaja objekata u binarnu formu.
- Tekstualna serijalizacija podrazumeva skladištenje u XML, JSON ili neki drugi tekstualni format.
- Binarna serijalizacija se može obaviti korišćenjem tokova objekata, odnosno klasa ObjectOutputStream i ObjectInputStream.
- XML serijalizacija se može obaviti upotrebom klase XMLEncoder.
- JSON serijalizacija se može obaviti korišćenjem brojnih eksternih biblioteka; jedna od njih je biblioteka Gson.

15. Fajlovi i folderi

- Fajl sistem jeste hijerarhijska organizacija fajlova i foldera, na čijem se vrhu nalazi koreni (root) folder.
- Unix i Linux kombinuju sve fajl sisteme u jedan virtualni fajl sistem, dok kod Windows operativnih sistema postoji po jedan fajl sistem za svaku kreiranu particiju.
- Funkcionalnosti za rukovanje fajl sistemom nalaze se u paketu java.io.
- Jedna od osnovnih klasa za rukovanje sistemom fajlova i foldera je klasa File.
- Putanja je lokacija određenog fajla na fajl sistemu; postoje dve vrste putanja.
- Apsolutna putanja je putanja do nekog fajla/foldera na fajl sistemu koja počinje simbolom korenog foldera.
- Relativne putanje ne počinju simbolom korenog foldera, već se oslanjaju na trenutnu lokaciju.
- Koristeći ugrađene funkcionalnosti klase File moguće je kreirati, uklanjati i menjati fajlove i foldere i dobijati razne informacije o fajl sistemu.

5. Rukovanje mrežom

16. Uvod u internet mrežne protokole

- Kompjuteri međusobno komuniciraju korišćenjem specijalnih hardverskih uređaja koji se nazivaju mrežne kartice.
- Internet je najveća danas poznata kompjuterska mreža, odnosno mreža svih mreža.
- Jezik kojim kompjuteri komuniciraju preko mreže drugačije se naziva protokol.
- Protokoli koji se koriste na internetu objedinjeno se nazivaju internet protokoli, a veoma često se oni mogu pronaći i pod nazivom TCP/IP.
- U komunikaciji kompjutera preko mreže učestvuju brojni protokoli, koji su logički podeljeni na četiri sloja: aplikativni, transportni, internet i mrežni.
- Aplikativni sloj direktno koriste aplikacije na jednom kompjuteru kako bi uputile ili pročitale poruku preko interneta; najznačajniji protokoli ovog sloja su HTTP, SMTP, IMAP, FTP, POP, SSL...
- Transportni sloj obezbeđuje isporuku poruka; najznačajniji protokoli ovog sloja su TCP i UDP.
- Internet sloj je zadužen za logičko adresiranje podataka; najznačajniji protokol ovog sloja je IP.
- Mrežni sloj direktno rukuje transportovanjem podataka kroz fizičku mrežu.
- Komunikacija kompjutera preko mreže zapravo je komunikacija između dva programa.
- Jedan kraj u komunikaciji između dva programa preko mreže drugačije se naziva
- Socket je određen IP adresom i brojem porta.
- Čest oblik komunikacije kompjutera preko mreže podrazumeva request–response model, koji počiva na međusobnom smenjivanju zahteva i odgovora između servera i klijenata.

17. TCP socket programiranje

- Klijent–server je model komunikacije koji diktira postojanje dva različita tipa kompjutera na mreži: klijenata i servera.
- Server ima ulogu slušaoca, zato što nadgleda sve aktivnosti na zadatom portu i, nakon što do aktivnosti dođe, inicijalizuje socket i obavlja određeni posao.
- Klijent inicira komunikaciju sa serverom i, nakon što prosledi svoj zahtev, očekuje od servera odgovor, čime se uspostavlja uzajamna razmena podataka.
- U Javi postoji nekoliko klasa koje se koriste za kreiranje TCP servera i TCP klijenata, koje se nalaze u paketu net.
- Za kreiranje TCP servera koristi se klasa ServerSocket, a za kreiranje klijenta klasa Socket.

18. Rukovanje HTTP-om

- World Wide Web se definiše kao skup dokumenata (resursa) u kojem je svaki takav dokument identifikovan korišćenjem specijalne adrese.
- Svoje funkcionisanje WWW servis direktno zasniva na HTTP, odnosno HTTPS protokolima.
- HTTP protokol razvijen je specijalno za potrebe WWW servisa; web ne bi postojao bez ovog protokola.
- HTTP je takozvani request–response protokol, što znači da njegov način funkcionisanja počiva na međusobnom smenjivanju zahteva i odgovora između dva kompjutera.
- U komunikaciji kompjutera korišćenjem HTTP protokola razlikuju se dva tipova uređaja: klijenti i serveri.
- Klijentski kompjuter je onaj sa koga je korišćenjem nekog web browsera upućen zahtev.
- Server je kompjuter koji prihvata i obrađuje zahtev i na kraju odgovara isporukom određene poruke ili resursa.
- Svi resursi na webu su jednoznačno određeni specijalnim adresama koje se nazivaju Uniform Resource Locator ili skraćeno URL.
- HTTP komunikacija se zasniva na međusobnom smenjivanju zahteva i odgovora; pri tome, zahtevi i odgovori poseduju

određenu strukturu koja je utvrđena pravilima HTTP protokola.

19. UDP socket programiranje

- UDP je skraćenica za **U**ser **D**atagram **P**
- Datagram je poruka čija isporuka nije zagarantovana.
- Funkcionalnosti za rad sa UDP-om nalaze se unutar paketa java.io.
- Za rad sa UDP-om u Javi postoje dve osnovne klase: DatagramSocket i DatagramPacket.
- DatagramSocket je klasa koja sluša i očekuje UDP pakete.
- DatagramPacket predstavlja UDP poruku.

6. Niti

20. Uvod u višenitno programiranje

- Procesi su način na koji operativni sistem izoluje i razdvaja izvršavanje programa.
- Izvršavanje jednog programa može se postići korišćenjem jednog ili više procesa.
- Nit je najmanja izvršna jedinica koja može postojati unutar procesa.
- Jedan proces može imati jednu ili više niti.
- Svaki proces mora imati makar jednu nit, koja se naziva primarna.
- Programiranje koje podrazumeva izvršavanje programa kroz nekoliko niti naziva se višenitno programiranje.
- Konkurentno izvršavanje označava obavljanje većeg broja poslova tokom istog vremenskog perioda.
- Mogućnost izvršavanja većeg broja programa tokom istog perioda kroz vremensku raspodelu procesorskih resursa drugačije se naziva multitasking.
- Paralelno izvršavanje predstavlja obavljanje većeg broja poslova u istom trenutku.

21. Rukovanje nitima

- Niti se u Javi predstavljaju korišćenjem klase Thread.
- Postoje dva načina na koje je moguće određeni kod izvršiti u zasebnoj niti: implementacija Runnable interfejsa ili nasleđivanje klase Thread.
- Oba pristupa podrazumevaju implementaciju metode run(), unutar koje je potrebno smestiti logiku niti.
- Kada se implementira interfejs Runnable, klasa Thread služi kao omotač za Runnable klasu, pomoću koga se njome manipuliše u kontekstu niti.
- Nit se pokreće metodom start(), klase Thread.
- Izvršavanje niti se može pauzirati na određeno vreme, korišćenjem metode sleep().
- Niti se mogu parametrizovati baš kao i bilo koji drugi objekti, i to korišćenjem konstruktora, polja, set metoda...
- Zaustavljanje niti korišćenjem metode stop() se smatra nebezbednim, pa se savetuje zaustavljanje upotrebom kontrolne vrednosti.

22. Konkurencija i sinhronizacija

- Niti primarno komuniciraju deljenjem resursa.
- Prilikom komunikacije niti može nastati nekoliko potencijalnih problema koji se rešavaju sinhronizacijom niti.
- Da bi se neki resurs u Javi sinhronizovao, koristi se modifikator synchronized ili istoimena izjava.
- Kada se neka metoda označi kao sinhronizovana, samo jedna nit u jednom trenutku može da je koristi.
- Tokom izvršavanja niti može se upravljati i korišćenjem metoda wait(), notify() i notifyAll().
- Metoda wait() se koristi za pauziranje izvršavanja neke niti.
- Metoda notify() se koristi da obavesti nit koja je u stanju čekanja da je resurs ponovno postao slobodan i da može da nastavi svoje izvršavanje.
- Metoda notifyAll() se koristi kada je potrebno obavestiti sve niti koje su na čekanju da mogu da nastave svoje izvršavanje.