Programski jezik Java

Osnovni koncepti programskog jezika Java

Sadržaj

- Uvod
- Osnovni elementi
- Kompozicija i nasleđivanje
- Rukovanje izuzecima
- Ulazno-izlazni podsistem
- Gotove klase

Uvod

- **Java**
  - programski jezik razvijen 1995. godine
  - James Gosling
  - Sun Microsystems
    - sada Oracle
- **Sintaksa**
  - slična C/C++
  - jednostavniji objektni model

Uvod

- **Java virtual machine (JVM)**
  - interpretira Java bytecode
    - „zamena“ za mašinski kod
    - svi programi se kompajliraju u bytecode
  - implementirane za skoro sve operativne sisteme
  - mogu da se implementiraju direktno na hardveru
  - omogućavaju nezavisnost od platforme

Uvod

- **Osobine Jave**
  1. jednostavna, objektno orijentisana i lako prihvatljiva sintaksa
  2. robustna i sigurna
  3. nezavisna od platforme i portabilna
  4. visoke performanse izvršavanja programa
  5. da se interpretira, poseduje niti i da je dinamička

Uvod

- **Performanse**
  - reputacija
    - sporo izvršavanje
    - neoptimizovano korištenje memorije
  - stvarnost
    - od verzije 1.1
      - ogromno poboljšanje u performansama
    - znatno manja razlika u odnosu na C/C++
      - pogotovo na današnjim računarima

Uvod

- **Automatsko rukovanje memorijom**
  - **garbage collector (GC)**
    - programer definiše vreme kreiranja objekta
    - GC sam uništava objekte
      - kada više nisu potrebni
    - sprečava curenje memorije
      - u tradicionalnom smislu
    - nepoznat trenutak pozivanja GC-a
      - zagarantovano prilikom nedostatka memorije na heap-u
  - skida teret rukovanja memorijom sa leđa programera

Uvod

- **Java edicije**
  - **Java Card**
    - aplikacije za smart kartice
  - Java Platform, Micro Edition (**Java ME**)
    - aplikacije za okruženja sa ograničenim resursima
  - Java Platform, Standard Edition (**Java SE**)
    - aplikacije za radne stanice
  - Java Platform, Enterprise Edition (**Java EE**)
    - aplikacije u velikim distribuiranim sistemima
    - Internet aplikacije

Sadržaj

Osnovni elementi

- **Paket**
  - sadrži grupu klasa, objedinjenu u istom prostoru imena
  - svaka klasa pripada paketu
    - neimenovani, podrazumevani paket
      - (default package)
  - U ***prvom redu*** klase, koji nije komentar, se definiše paket kome pripada
    - rezervisana reč **`package`**
    - `package imePaketa;`

Osnovni elementi

- **Paket**
  - ime paketa zajedno sa nazivom klase, daje njeno puno ime
    - ukoliko se klasa `TestNiz` nalazi u paketu `test`, njeno puno ime će biti `test.TestNiz`
  - rezervisana reč **`import`**
    - `import test.TestNiz;`
    - `import test.*;`

Osnovni elementi

- **Paket**
  - ime paketa implicitno određuje strukturu direktorijuma
    - poštovati pravila davanja imena kao i kod direktorijuma
    - paketi obično počinju malim slovom
  - paketi reprezentuju **hijerarhiju**
    - svaki nivo hijerarhije se razdvaja tačkom
      - rs.ac.uns.ftn.oop.figure
      - viewer.elements
      - controller.servlets
  - ❖ primer kreiranja paketa

Osnovni elementi

- **Komentari**
  - `//`
    - jednolinijski komentar
  - `/* */`
    - blok komentar
  - `/** */`
    - Javadoc komentar
    - mehanizam za automatsko generisanje dokumentacije

Osnovni elementi

- **Klasa**
  - rezervisana reč `class`
  - počinje velikim početnim slovom
  - fajl u kome je smeštena klasa
    - mora imati isto ime kao i klasa
      - case sensitive
    - ekstenzija `.java`
  - modifikator pristupa klasi
    - `public class ImeKlase { ... }`
  - modifikator pristupa elementa klase
    - piše se uz svako polje i metodu

Osnovni elementi

- **Klasa**
  - jedna klasa = jedan fajl
    - samo jedna public klasa može postojati u jednom fajlu
    - jedan fajl sadrži kompletnu klasu
  - sav kôd se piše unutar klase
    - **sve je objekat**
      - osim primitivnih tipova
        - » byte, short, int, long, float, double, boolean, char
      - primitivni tipovi poseduju wrappere
        - » pretvaraju ih u objekte

Osnovni elementi

- **Klasa**
  - ne postoje pokazivači
    - **svi objekti se prenose po adresi (referenci)**
    - **svi primitivni tipovi se prenose po vrednosti**
  - ne postoji preklapanje operatora
  - ne postoji destruktor
    - GC obavlja posao

Osnovni elementi

- **Klasa**
  - izvršiva
    - ukoliko poseduje main metodu
      - `public static void main(String[] args) { ... }`

❖ primer kreiranja klase

➢ zadatak 1

- U programskom jeziku Java napisati klase Krug i JSTrougao (jednakostranicni trougao). Osim potrebnih polja, klase treba da imaju metode koje racunaju obim, odnosno površinu. Napisati test program koji testira funkcionalnost ovih klasa.

Osnovni elementi

- **Niz**
  - sekvenca **objekata** ili **elemenata prostog tipa**
    - svi elementi su **istog** tipa
    - pod jednim imenom za identifikaciju
      - naziv niza
  - operator indeksiranja `[ ]`
  - referenca na niz
    - `int[] a`
    - `int a[]`

Osnovni elementi

- **Niz**
  - inicijalizacija niza
    - `int[] a = {1, 2, 3, 4, 5};`
      - samo na mestu definisanja niza
    - `int[] a = new int[5];`
      - bilo gde u kôdu
      - potrebno je dodatno zadati vrednosti elementima niza

Osnovni elementi

- **Niz**
  - **`length`** – broj elemenata u nizu
    - svaki niz poseduje ovaj član
    - može se čitati ali ne i menjati
  - prvi element se nalazi na lokaciji **`0`**
  - poslednji element je na lokaciji **`length - 1`**
    - `RuntimeException` ukoliko prekoračimo veličinu niza

Osnovni elementi

- **Niz**
  - `java.util.Arrays.toString(niz)`
    - ispis niza
  - ❖ primer nizova
  - ➢ zadatak 2
    - U programskom jeziku Java napisati klasu koja u okviru main metode kreira niz od 5 objekata klase Pravougaonik i pronalazi pravougaonik sa najvećom površinom

Sadržaj

- Uvod
- Osnovni elementi
- Kompozicija i nasleđivanje
- Rukovanje izuzecima
- Ulazno-izlazni podsistem
- Gotove klase

Kompozicija i nasleđivanje

- **Motivacija**
  - klasa bi trebalo da predstavlja korisnu jedinicu koda
    - idealan slučaj
    - potrebno je iskustvo i pronicljivost kako bi se napravio višekratno upotrebljiva klasa
  - ponovno korišćenje kôda je jedna od najvećih prednosti OOP

Kompozicija i nasleđivanje

- **Kompozicija**
  - sastavljanje klasa od već postojećih klasa
    - često se poredi sa relacijom „ima“
      - „auto ima motor“
  - vrlo je fleksibilna
    - objekti članovi su obično privatni
      - nevidljivi drugim programerima
    - promena unutrašnje strukture ne znači nužnu promenu klijentske aplikacije

Kompozicija i nasleđivanje

- **Kompozicija**
  - ❖ primer kompozicije
  - ➢ zadatak 3
    - Napisati klasu PP3Prizma koja modeluje pravu pravilnu trostranu prizmu. Prizmu modelovati kao kompoziciju baze (objekat klase JSTrougao) i omotača (objekat klase Pravougaonik). Klasa treba da sadrži metode koje računaju površinu i zapreminu prizme. Napisati test program koji testira funkcionalnost klase.

Kompozicija i nasleđivanje

- **Nasleđivanje**
  - način „kloniranja“ već postojećih klasa
    - izmena ili dodavanje sadržaja
  - nasleđivanjem mi stvaramo novu klasu
    - rezervisana reč **`extends`**
  - sadrži sve članove postojeće klase
    - i privatne članove iako su oni skriveni i nedostupni
  - kopira „interfejs“ postojeće klase
    - sve poruke koje možemo poslati objektima postojeće klase možemo poslati i objektima izvedene

Kompozicija i nasleđivanje

- **Nasleđivanje**
  - jednostruko nasleđivanje
    - višestruko nasleđivanje **ne postoji** u Javi
      - mehanizam implementacije interfejsa
  - redefinisanje metoda
    - sve metode su virtuelne, nema posebne rezervisane reči
      - tzv. late binding (dinamičko povezivanje koda)
    - opciono `@Override`

Kompozicija i nasleđivanje

- **Nasleđivanje**
  - **polimorfizam**
    - prilikom nasleđivanja i obrazovanja hijerarhije
    - svaki izvedeni tip se može posmatrati kao osnovni (kao neki od direktnih potomaka)
      - obrnuto ne važi
  - zabrana nasleđivanja klase
    - rezervisana reč **`final`**

Kompozicija i nasleđivanje

- **Nasleđivanje**
  - klasa **`java.lang.Object`** predak svih klasa napisanih u Javi
    - implicitno nasleđena tj svi objekti u javi imaju isti interfejs u osnovi, stoga su istog tipa
    - određena zajednička funkcionalnost tj znamo da možemo izvršiti neke osnovne operacije
      - `clone()`
      - `equals(Object obj)`
      - `toString()`
      - `hashCode()`
      - `finalize()`
    - olakšava rad GC-a jer uvek se zna kog je tipa klasa

Kompozicija i nasleđivanje

- **Nasleđivanje**
  - **apstraktna klasa**
    - `abstract` rezervisana reč
    - ne može da se instancira
    - može da ima apstraktne metode
      - ista rezervisana reč kao i za apstraktnu klasu
    - obično se prave za potrebe definisanja hijerarhije
  - apstraktne metode mogu biti definisane isključivo u apstraktnoj klasi

Kompozicija i nasleđivanje

- **Nasleđivanje**
  - ❖ primer nasleđivanja
  - ➢ zadatak 4
    - Napisati apstraktnu klasu GeometrijskoTelo sa apstraktnim metodama za izračunavanje površine, odnosno zapremine. Napisati klasu Kvadar koja nasleđuje klasu GeometrijskoTelo i redefiniše potrebne metode. Napisati klasu Kocka koja nasleđuje klasu Kvadar.

Kompozicija i nasleđivanje

- **Nasleđivanje**
  - **interfejsi**
    - `interface` ključna reč
    - unapređuju koncept apstrakcije
    - **implementiraju** se od strane klasa
      - nasleđuju se ako želimo da pravimo novi interfejs
    - jedna klasa može da imlementira proizvoljan broj interfejsa
      - i pored toga da nasledi jednu klasu
      - emuliranje višestrukog nasleđivanja

Kompozicija i nasleđivanje

- **Nasleđivanje**
  - ❖ primer nasleđivanja uz implementaciju interfejsa
  - ➢ zadatak 5
    - Napisati interfejs GeometrijskoTelo sa apstraktnim metodama za izračunavanje površine, odnosno zapremine. Napisati klasu Kvadar koja implementira GeometrijskoTelo i redefiniše potrebne metode. Napisati klasu Kocka koja nasleđuje klasu Kvadar.

Sadržaj

- Uvod
- Osnovni elementi
- Kompozicija i nasleđivanje
- Rukovanje izuzecima
- Ulazno-izlazni podsistem
- Gotove klase

Rukovanje izuzecima

- **Rukovanje greškama za vreme rada**
  - ne mogu se sve greške otkriti za vreme prevođenja
  - povećava se robusnost sistema
  - obrada izuzetaka je **jedini zvanični način** na koji Java prijavljuje greške
    - sprovodi je prevodilac Jave
    - umesto vraćanja posebne vrednosti ili postavljanja indikatora

Rukovanje izuzecima

- **Vanredno stanje**
  - *eng. exceptional condition*
  - sprečava nastavak rada tekuće metode ili programskog bloka
  - iskakanje iz tekućeg okruženja omogućava **izuzetak** (*eng. exception*)
    - pravi se novi objekat u dinamičkoj memoriji sa operatorom new
    - zaustavlja se aktuelna putanja izvršavanja i obradu preuzima mehanizam obrade izuzetaka

Rukovanje izuzecima

- **Izuzetak**
  - generisanje („bacanje“) izuzetaka
    - izuzeci se kreiraju kao i ostali objekti sa operatorom new
    - koren hijerarhije klasa `Throwable`
      - nasleđuje klasu `Object`
      - nasleđuju je klase `Error` i `Exception`
      - nasleđivanjem klase `Exception` (ili eventualno `Throwable`) možemo praviti naše izuzetke
    - definicija liste izuzetaka koje metoda generiše
      - rezervisana reč **`throws`**
      - `public void myMethod() throws MyException {...}`
    - generisanje izuzetka sa rezervisanom rečju **`throw`**
      - unutar metode koja generiše izuzetak tog tipa
      - `throw new Exception();`

Rukovanje izuzecima

- **Obrada izuzetaka**
  - **ispitni blok**
    - *eng. try block*
    - rezervisana reč **`try`**
    - „isprobava“ pozive metoda
    - izuzetak ne izaziva iskakanje iz metode ili programa
    - sve što se testira stavlja se u *try block*

Rukovanje izuzecima

- **Obrada izuzetaka**
  - **blok za obradu izuzetaka**
    - *eng. exception handler*
    - rezervisana reč **`catch`**
    - mehanizam za obradu izuzetaka
    - postoji za svaki tip izuzetaka koji se obrađuje
    - dolazi nakon *try block*-a
    - traži se prvi blok čiji argument odgovara tipu izuzetka
    - specifičniji blok pre uopštenijeg
      - najuopšteniji - `Exception e`

Rukovanje izuzecima

- **Obrada izuzetaka**
  - **block finally**
    - uvek se izvršava
    - možemo odraditi neke akcije koje ne zavise od toga da li se izuzetak dogodio ili ne
    - dolazi na kraju, iza svih blokova za obradu izuzetaka
    - koristi se kada je potrebno da se neki element vrati u prvobitno stanje
      - zatvaranje otvorene datoteke, zatvaranje mrežne konekcije ...
    - ne postoji u C++
      - oslanjamo se na destruktore

Rukovanje izuzecima

- **Obrada izuzetaka**
  - **primer rukovanja izuzecima:**

```
try {
        id = Integer.parseInt(br.readLine());
} catch(NumberFormatException nfe) {
        System.out.println("Identifikacioni broj
        mora biti celobrojna vrednost!");
        //nfe.printStackTrace();
} catch (Exception e) {
        //do something
        e.printStackTrace();
} finally {
        //do something
}
```

Sadržaj

- Uvod
- Osnovni elementi
- Kompozicija i nasleđivanje
- Rukovanje izuzecima
- Ulazno-izlazni podsistem
- Gotove klase

Ulazno-izlazni podsistem

- **Serijalizacija objekata**
  - pretvaranje objekta u niz bajtova iz kojeg se taj objekat u potpunosti može rekonstruisati
  - snimanje objekata u datoteku ili bazu podataka
    - sa mogućnošću njihove ponovne rekonstrukcije
  - Java se brine za sve pojedinosti
    - klasa mora da implementira interfejs `Serializable`
      - nema nijedne metode za implementaciju samo se klasa proglasi serijalizabilnom
  - svi podobjekti moraju implementirati isti interfejs
    - primitivni tipovi se automatski serijalizuju
  - prate se sve reference koje objekat sadrži

Ulazno-izlazni podsistem

- **Serijalizacija objekata**
  - rezervisana reč **`transient`**
    - nekada ne želimo da serijalizujemo određeni podobjekat
      - zbog mogućeg narušavanja bezbednosti
      - primer: lozinke
  - **statička polja se ne serijalizuju**
    - to moramo sami da uradimo
    - posebne metode za snimanje i učitavanje ili izračunavanje vrednosti

Ulazno-izlazni podsistem

- **Ulazno-Izlazni podsistem**
  - **`java.io`** paket
  - standardna biblioteka za ulazno-izlazne operacije
    - ulazni i izlazni tokovi podataka
    - čitači i pisači kao proširenje tokova
      - problemi sa internacionalizacijom i lokalizacijom
    - klase **`RandomAccessFile`** i **`File`**
  - klasa **`RadnomAccessFile`**
    - osnovna manipulacija sa datotekom
    - slučajan pristup
  - klasa **`File`**
    - manipulisanje datotekama i direktorijumima
    - pročitati javadoc

Ulazno-izlazni podsistem

- **Ulazno-Izlazni podsistem**
  - **tokovi** omogućuju prenos podataka
    - memorija, datoteke, cevi (*pipes*)...
    - unificiran pristup bez obzira na lokaciju
    - metode su bajt orijentisane
    - koriste se u svim slučajevima osim kod čitanja Unicode karaktera
  - **pisači** i **čitači**
    - *reader* i *writer* klase
    - problem kod bajt orijentisanih tokova su 16-bitni Unicode karakteri
    - primer: klase **`FileReader`**, **`FileWriter`**, **`BufferedReader`** i **`PrintWriter`**
    - koriste se u slučaju čitanja Unicode karaktera

Ulazno-izlazni podsistem

- **Ulazno-Izlazni podsistem**
  - primeri tokova
    - klase **`ObjectOutputStream`** i **`ObjectInputStream`**
    - metode **`writeObject()`** i **`readObject()`**
    - poželjno korištenje metoda **`close()`** i **`flush()`**
  - učitavanje sa tastature
    - `BufferedReader br = new BufferedReader(new InputStreamReader(System.in));`

Ulazno-izlazni podsistem

- **Ulazno-Izlazni podsistem**
  - zadatak 6
    - napisati interfejs **Radi**
      - *metode*: radi() i odmaraj()
    - napisati klasu **Osoba**
      - *polja:* **jmbg**, **ime**, **prezime** (string) i **godiste** (int)
      - *metode*: get i set, konstruktor i toString
    - napisati klasu **Radnik** koji nasleđuje klasu **Osoba** i implementira interfejs **Radi.**
      - *dodatna polja*: **plata**, **staz** (int) i **radiTrenutno** (boolean)

Ulazno-izlazni podsistem

- **Ulazno-Izlazni podsistem**
  - zadatak 6
    - napisati klasu **Robot**
      - *polja*: **naziv**, **proizvodjac** (string), **serijskiBroj** i **godiste** (int)
      - *metode*: get i set, konstruktor i toString
    - napisati klasu **Radilica** koja nasleđuje klasu **Robot** i implementira interfejs **Radi.**
      - *dodatna polja*: datumPregleda (Date), **radnik**(Radnik) i **uPogonu** (boolean)
    - napisati klasu **Test** koja proverava klase tako što poziva njihove metode
      - klase se kreiraju pomoću parametara unešenih sa tastature

Sadržaj

- Uvod
- Osnovni elementi
- Kompozicija i nasleđivanje
- Rukovanje izuzecima
- Ulazno-izlazni podsistem
- Gotove klase

Gotove klase

- **Klasa `String`**
  - nije samo niz karaktera, posebna klasa
  - **immutable**
    - ne može se promeniti nakon konstruisanja
    - može biti samo napravljen novi string
  - poređenje se obavlja pomoću metode `equals()`
    - nikako sa == jer se tako porede samo reference a ne sadržaj
  - konkatenacija sa operatorom +

Gotove klase

- **Klasa `String`**
  - Korisne metode
    - `str.length()`
    - `str.charAt(i)`
    - `str.indexOf(s)`
    - `str.substring(a,b)`
    - `str.substring(a)`
    - `str.equals(s)`
    - `str.equalsIgnoreCase(s)`
    - `str.startsWith(s)`
    - `str.split(regex)`
  - ❖ primer korišćenja klase `String`

Gotove klase

- **Klasa `ArrayList`**
  - kolekcija elemenata
  - generičnost od Jave 1.5
  - moguće postaviti tip elemenata klase
    - kolekcija elemenata definisanog tipa
  - za prolazak se može koristiti sintaksa **for** petlje
  - ❖ primer korišćenja klase `ArrayList`

Gotove klase

- **Javadoc**
  - generator dokumenata
    - HTML format
    - iz samog source koda
  - http://docs.oracle.com/javase/6/docs/api/
  - ❖ primer Javadoc-a

Gotove klase

- **Gotove klase**
  - zadatak 7
    - napisati klasu **Predmet**, koja sadrži polja *broj* (tipa int), *ime* (tipa String), *tekst* (tipa String).
    - napisati klasu **Evidencija** koja modeluje evidenciju predmeta u sudu
      - polja: *nazivSuda* (tipa String), *datum* (tipa Date) i *listu predmeta* (izaberite neku dinamičku strukturu)
      - metode:
        - za ispis predmeta na osnovu broja
        - za ispis predmeta na osnovu imena
        - za dopisivanje novog teksta predmeta (u nastavku postojećeg teksta)
    - napisati kratak test program

Pitanja i komentari

Sadržaj

- Uvod
- Osnovni elementi
- Kompozicija i nasleđivanje
- Rukovanje izuzecima
- Ulazno-izlazni podsistem
- Gotove klase

Kraj prezentacije

Programski jezik Java

Osnovni koncepti programskog jezika Java