Predavanja

OBJEKTNO-ORIJENTISANO PROGRAMIRANJE

2024/25.

Opšte informacije

NASTAVNIK	Ana Kaplarević-Mališić
SARADNICI	Andreja Živić
	Anđelina Maksimović
	Miloš Đuričić
MATERIJALI	IMI Moodle stranica
	Obaveštenja će biti objavljivana na oglasnoj tabli na Moodle stranici. Prijavite se da pratite predmet! Proverite koji mejl vam je vezan za IMI nalog.
KOMUNIKACIJA	mejl, časovi, konsultacije
	Teams – ako vam odgovara da napravimo grupu

Literatura i način polaganja

Poeni

50 + 50 (10+15+25)

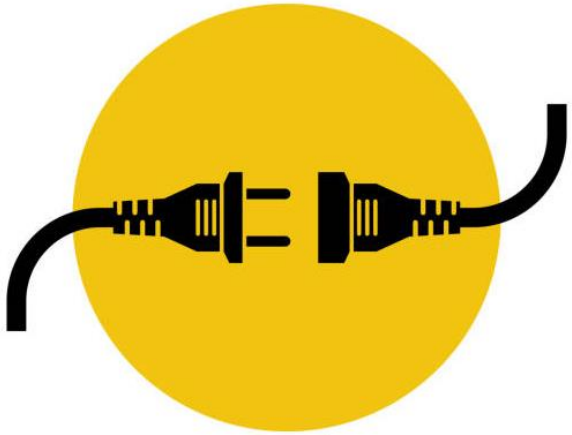
2 kolokvijuma + završni (E + A + B/seminarski)

Literatura

D. Poo, D. King, S. Ashok, *Object-oriented programming in Java*, Springer-verlag, 2008.

I. Horton, *Java2 - JDK 1.5*, CET, Beograd, 2006.

Od početka kursa od vas očekujem(o)



Na kraju kursa od vas očekujem(o)

Da razumete osnovne koncepte OOP-a

dovoljno da **u svakom trenutku** umete da ih u par rečenica i na primeru objasnite nekome ko je završio bilo koji uvodni kurs programiranja (OP, PiP1, ...)

Da umete da odgovorite na većinu pitanja koja se javljaju na testovima za Java SE Foundations sertifikat

Dobijaćete slične ili iste na ispitu

Da umete da pišete programe u Javi koji rade.

Programi koji podrazumevaju primenu OO koncepata.

Jednostavne grafičke aplikacije (igrice sa jednostavnom logikom) korišćenjem SWING biblioteke.

Agenda

za danas

Par pojmova pre početka*

Objektno-orijentisani v.s.
Proceduralni pristup

Objekti i klase

Java - Zdravo, svete!

Par pojmova pre početka*

AGENDA

Par pojmova pre početka*

Objektno-orijentisani v.s. Proceduralni pristup

Objekti i klase

Java - Zdravo, svete!

Određivanje n! korišćenjem različitih paradigmi programiranja

Imperativno programiranje

PROCEDURALNO PROGRAMIRANJE C

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

…
printf("%d",fakt(10));
```

Podaci u strukturama. Posao opisan funkcijom koja sadrži sve korake izračunavanja.

OBJEKTNO-ORJENTISANO PROGRAMIRANJE JAVA

```
class Calc  {
   static int fakt()
   {
     if (n==1) return 1;
     return n*fakt(n-1);
   }
}
…
System.out.println(Calc.fakt(10));
```

Posao opisan funkcijom. Funkcija ima izvršioca.

Deklarativno programiranje

FUNKCIONALNO PROGRAMIRANJE HASKEL

```
factorial 0 = 1
factorial n = n * factorial (n - 1)
…
main = print (factorial 10)
```

Svaki zahtev se svodi na evaluaciju izraza

LOGIČKO PROGRAMIRANJE PROLOG

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

…

?-fakt(10,X).
```

Svaki zahtev se svodi na određivanje logičkih posledica

Programske paradigme (i samo slični pojmovi)

Programski jezici se međusobno razlikuju po načinu na koji modeliraju realne probleme - **programskim paradigmama (paradigmama programiranja)** 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

Stil kojim je program napisan. (elegancija 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.

Kompajleri i interpreteri

Da bi bio izvršen program (pisan u nekom od viših programskih jezika) mora biti preveden na mašinski jezik.

PREVOĐENJE JE MOGUĆE IZVESTI

U POTPUNOSTI PRE IZVRŠVANJA

- Prevođenje vrši prevodilac (kompajler) odgovarajućeg programskog jezika.
- Nakon što je jednom preveden, program u mašinskom jeziku se može izvršiti neograničen broj puta, ali, naravno, samo na određenoj vrsti računara.

C, C++

TOKOM IZVRŠAVANJA

- Prevođenje jedne po jedne naredbe prema potrebi od strane interpretera, programa koji se ponaša kao CPU s nekom vrstom dobavi-i-izvrši ciklusa. Da bi izvršio program, interpreter radi u petlji u kojoj uzastopno čita naredbe iz programa, odlučuje šta je potrebno za izvršavanje te naredbe, i onda je izvršava
- Interpreteri se mogu koristiti za izvršavanje mašinskog programa pisanog za jednu vrstu računara na sasvim različitom računaru.

Python, PHP, JavaScript

Strukture i tipovi

Tip podataka je određen (konačnim) skupom vrednosti i nekim paketom operacija i relacija nad elementima tog skupa.

Primer.

({-32768, ..., -1, 0, 1, ..., 32767}, +, -, * div, mod, =, <)

Struktura podataka je konglomerat podataka. Ona se formira od drugih (jednostavnijih) struktura i od tipova. Jedine operacije koje se nad strukturama mogu obavljati su operacije selekcije.

Primer. var **a**: **array** [1 .. 100] of int;

struktura **metod struktuiranja**

Metod struktuiranja ima sopstvene operacije selekcije.
Pr. Nizovi imaju indeksiranje (a[i]),
Slogovi imaju projektovanje (odabir polja sloga, a.ime)

eksplicitni metodi struktuiranja - array, record/structure, …
implicitni - preko pokazivaca

Zašto je dobro praviti nove tipove?

Zadatak: Napisati program koji izračunava matricu C prema sledećoj formuli

$$C = A^{-1} \cdot B^T + |Y| + Adj(X)$$

I verzija

Definišemo A, B, X, Y kao matrice i onda

```
for i := 1 to n do
       for j := 1 to n do
              (* rutina koja invertuje matricu A u A1 *);
for i := 1 to n do
       for j := 1 to n do
              B1[i,j] := B[j, i];
for i := 1 to n do
        for j := 1 to n do
              (* rutina koja mnozi A1 i B1 *);
(* itd *)
```

Proceduralna apstrakcija

II verzija

Dekomponovati zadatak na manje. Napisati procedure za manipulaciju strukturama podataka.

```
var a, b, a1, c, ...: ARRAY[1..100, 1..100] of integer;
BEGIN
    Inv(A, A1);
    Transpose(B, B1);
    MatMul(A1, B1, P);
    Adj(X, X1);
    ScalMul(Det(Y), Q);
    MatAdd(P, Q, C);
END;
```

Apstrakcija podataka 1/3

III verzija

Definisati tip - Dizajnirati odgovarajuće strukture podataka, napisati procedure za manipulaciju tim strukturama podataka.

```
var a, b, a1, c, ...: MATRIX;
begin
    NewMatrix(a);
    NewMatrix(a1);
    ReadMatrix(a);
    ReadMatrix(b);
    Invert(a,a1);
    ...
    WriteMatrix(c);
    DisposeMatrix(a);
    ...
end;
```

Apstrakcija podataka 2/3

Razdvojiti upotrebu od implementacije.

Upotreba

```
var a, b, a1, c, ...: MATRIX;
begin
    NewMatrix(a);
    NewMatrix(b);
    ReadMatrix(a);
    ...
    WriteMatrix(c);
    DisposeMatrix(a);
    ...
end;
```

Implementacija

```
type MATRIX = array[1..X, 1..Y] of real;
```

```
type MATRIX = ^MatrixEntry;
       MatrixEntry =  record
                       i, j: integer;
                     entry: real;
                     right, down: MATRIX
                     end;
```

```
NewMatrix(a) begin ... end;
```

Korisnik ne mora da zna da li su matrice predstavljene kao array ili preko pokazivača, već samo kako se ponašaju operacije nad vrednostima tog tipa, tako da piše program koristeći samo osobine operacija.

KORISNIKU TIPA NIJE BITNA IMPLEMENTACIJA, VEĆ MANIFESTACIJE (TJ. OSOBINE) OPERACIJA DATOG TIPA PODATAKA.

Abstract data type (ADT) – Osnovna ideja OOP-a

Apstrakcija podataka + Sakrivanje informacija = Apstraktni tip podatka

Data Abstraction + Information hiding = ADT

Apstraktni tip podataka je tip podataka čiju implementaciju korisnik ne zna.

Programeru se da ime tipa i paket procedura. Implementacija tipa je SAKRIVENA. Dakle, apstrahovana je jedna dimenzija problema: implementacija tipa.

Objektno-orijentisani v.s. Proceduralni pristup

AGENDA

Par pojmova pre početka*

Objektno-orijentisani v.s. Proceduralni pristup

Objekti i klase

Java - Zdravo, svete!

Objektno-orijentisani v.s. Proceduralni pristup modelovanju

U jednoj stambenoj zgradi na svakom od m spratova (m<10) ima po n stanova (n<10). Napisati program za pomoć u radu Kućnog saveta na sledeći način:

- Za svaki stan je poznat broj i površina stana, ime vlasnika, starost vlasnika i njegov radni status (nezaposlen, zaposlen, penzioner).
- Podaci o stanovima i njihovim vlasnicima se nalaze u fajlu, od prvog ka poslednjem spratu. Stanovi se numerišu tako tako da prva cifra predstavlja sprat na kome se stan nalazi, a druga redni broj stana na spratu.
- Treba odrediti predsednika Kućnog saveta, a za predsednika bira najmlađi penzioner u zgradi.
- Odrediti stanare koji će učestvovati u krečenju zgrade. Kućni savet odlučuje o tome, tako što bira pet najmlađih nezaposlenih stanara da obave krečenje.

Proceduralni pristup modelovanju

- Odrediti šta je potrebno od podataka i definisati strukturu/e

```
stan=record
    broj:string;
    povrsina:integer;
    ime:string[6];
    godiste:integer;
    posao:status;
end;
```

- Uočiti zadatke iz kojih se sastoji ceo posao, a potom se na osnovu uočenih zadataka definisati procedure za izvršavanje

```
Unos
Predsednik
Krečenje
```

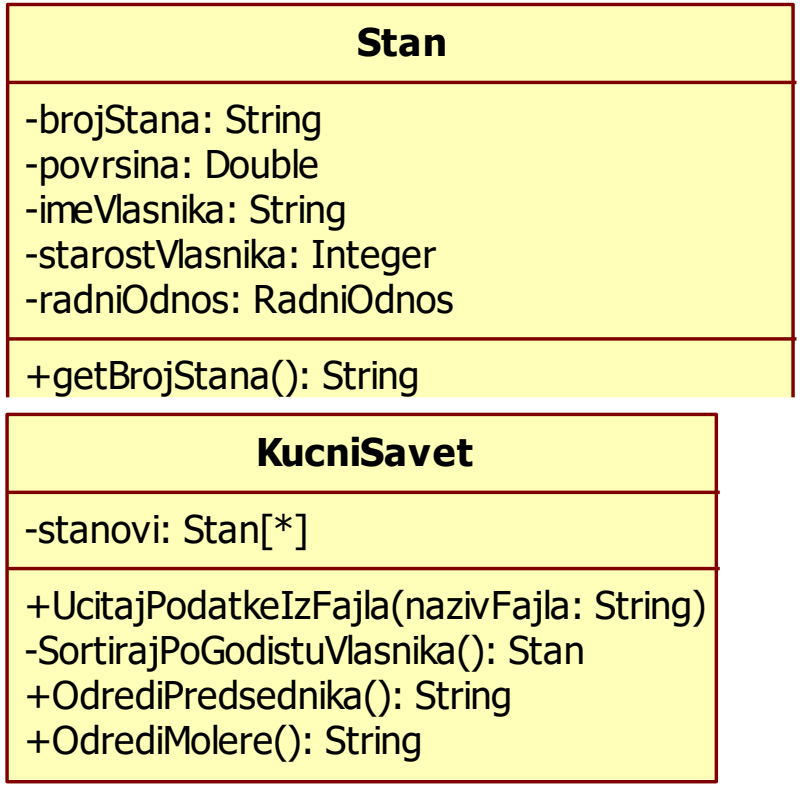
- U glavnom delu programa treba odrediti redosled pozivanja napisanih procedura

```
begin
  unos(a,m,n);
  sortg(a,b,k,m,n);
  predsednik(b,k);
  krecenje(b,k);
end.
```

OO pristup modelovanju

- Razmatra se kompletan sistem u kome se obavlja posaoUočavaju se učesnici i aktivnosti koje oni znaju da obavljaju sa dostupnim podacima (sakrivanje podataka);

- Kućni savet
  zna koji su stanovi u zgradi
  ume da formira listu stanova,
  izbor predsednika
  određuje učesnike krečenja
- Stan
  zna sve o sebi
  ume da pruži uvid u svoje podatke



OOP – izvršavanje posla

```
public class OOP_primer {
   public static void main(String[] args) {
        KucniSavet kucniSavet = new KucniSavet();
        kucniSavet.UcitajPodatkeIzFajla("UlazniPodaci.txt");

        System.out.println("Predsednik saveta je: " + kucniSavet.OdrediPredsednika());

         System.out.println("Moleri su:");
        String[] moleri = kucniSavet.OdrediMolere();
        for (int i = 0; i < moleri.length; i++) {
            System.out.println(moleri[i]);
        }
    }
}
```

OOP vs. Proceduralno programiranje

Proceduralno programiranje

Osnovni gradivni blok programa je **funkcija**.

Postavljeni zadatak se rešava tako što se razbije na niz manjih zadataka od kojih se svaka može implementirati u jednoj funkciji, tako da je **program niz funkcijskih poziva**.

Objektno-orijentisano programiranje

Osnovnu ulogu imaju **objekti** koji sadrže i podatke i funkcije (metode). Podaci koje objekat sadrži predstavljaju njegovo stanje, dok pomoću metoda on to stanje da može menja i komunicira sa drugim objektima.

Program se konstruiše kao skup objekata koji međusobno komuniciraju.

Motivacija

OOP je nastalo kao jedna od posledica softverske krize.

Softverska kriza 60-tih:

- kasne isporuke
- probijanje rokova i budžeta
- loš kvalitet
- nezadovoljavanje potreba
- slaba pouzdanost

Kompleksnost softvera zahtevala je promene u stilu programiranja.

Cilj je bio da se:

- proizvodi pouzdan softver
- smanji cena proizvodnje softvera
- razvijaju ponovo upotrebljivi moduli
- smanje troškovi održavanja
- smanji vreme razvoja softvera

Objekti i klase

AGENDA

Par pojmova pre početka*

Objektno-orijentisani v.s. Proceduralni pristup

Objekti i klase

Java - Zdravo, svete!

Tip, implementacija tipa i upotreba

Tip

Svaki student ima ime i nayiv fakulteta.
Ume da da informacije o sebi.

Klasa – implementacija tipa

```
class Student {
        String ime;
        String fakultet;

        public String getIme() { }
        public String getFakultet() { }
}
```

Objekat primerak klase/tipa - Upotreba tipa

```
Student s = new Student();
System.out.println(s.getFakultet());
```

Klasa i objekat

Klasa je tip, a objekat primerak tipa.

Objekat se kreira na osnovu šablona opisanog klasom.

Ono što nije sakriveno od korisnika, pre svega metodi – interface.

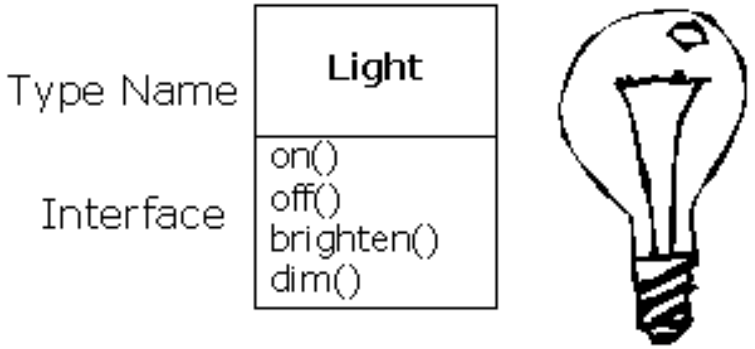
podaci = stanje objekta

opisuju stanje

+

procedure = metodi

objekat



- pripadaju objektu
- menjaju stanje objekta
- njima se opisuje tzv. ponašanje objekta

“Zdravo, Svete!” u Javi

AGENDA

Par pojmova pre početka*

Objektno-orijentisani v.s. Proceduralni pristup

Objekti i klase

Java - Zdravo, svete!

Java

Just Another Vague Acronim - JAVA

1991. tvorac Jave – James Gosling, Sun Microsystems

Stvorio jednostavan, platformski nezavistan jezik
Namenjen pokretanju elektronskih uređaja (Interaktivna TV, inteligentne rerne, telefoni,..)

1995. Java se lansira na SunWorld-u,

obavljuje se kod i dokumentacija Jave na Internetu

1996. Sun razvija JDK 1.0

1999. Pojavljuje se JDK 1.2 - Java 2 SDK - Software Development Kit

2006. Sun objavljuje veliki deo Jave kao slobodan i otvoren kod pod GPL licencom

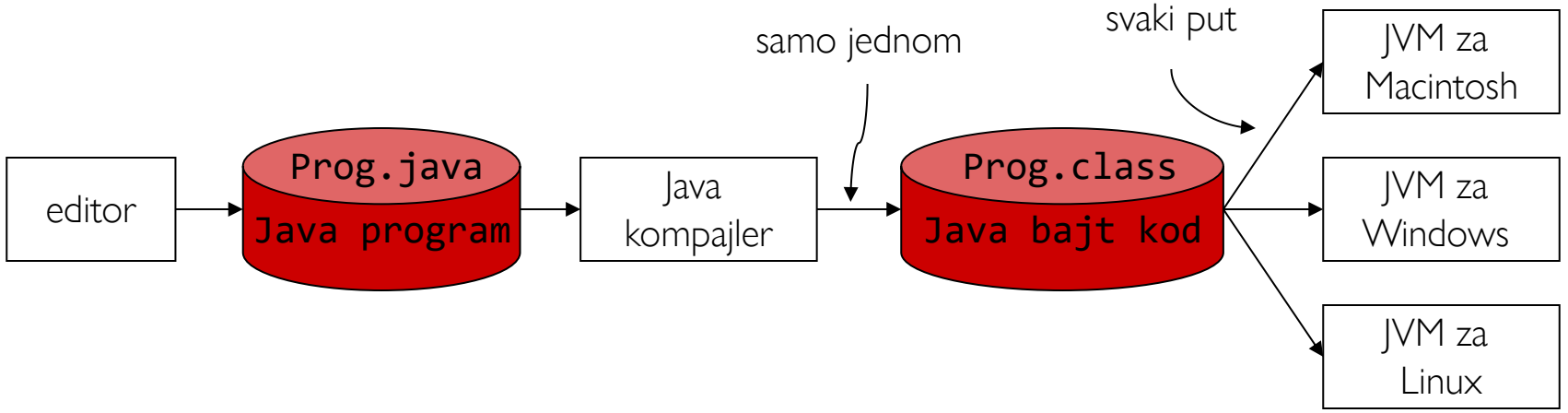
2010. Java postaje vlasništvo Oracle-a

Poslednja SE (Sttandard Edition) verzija sa LTS (Long Term Support) - Java 21

Trenutna SE verzija – Java 23

Kako je omogućena prenosivost?

- Projektanti Jave su se odlučili za upotrebu kombinacije kompajliranja i interpretiranja. Programi pisani u Javi se prevode u mašinski jezik virtuelnog računara, tzv. Java Virtual Machine.
- Mašinski jezika za Java Virtual Machine se zove Java bytecode.
- Sve što je računaru potrebno da bi izvršio Java bajt kod jeste interpreter. Takav interpreter oponaša virtuelnu Java mašinu i izvršava program.



Jednom preveden kod se može izvršiti na bilo kojoj mašini koja ima (odgovarajuću) Java virtuelnu mašinu

Zdravo, Svete!

```
// HelloWorld.java

public class HelloWorld {
    public static void main(String[ ] args) {
        System.out.println("Hello, World");
    }
}
```

- svaka Java aplikacija mora sadržati barem jednu klasu s metodom

  `main(String[] args)`

- počinje svoje izvršavanje pozivom metoda **`main`**
- ovako napisan program se prevodi izvršavajući

  `javac HelloWorld.java`

- Ako nema grešaka prevodilac javac kreira datoteku **`HelloWorld.class`** koja sadrži bytecode instrukcije za JVM.
- A pokreće se pozivom JVM uz prosledjivanje bajtkoda

  `java HelloWorld`

Osnovne karakteristike jezika

- Objektna orijentacija

  podržava sve koncepte objektno orijentisanog programiranja

  sintaksa slična C++, OO model jednostavniji

- Prenosivost (*portability*)

  Java programi se prevode u byte kod koji nije mašinski jezik nijednog konkretnog računara, već se izvršava na JVM (Java Virtuelna Mašina)

- Sigurnost

  JVM pruža zaštitu od virusa koji bi se prenosili kroz izvršni kod

- Robusnost

  Stroga provera tipova, proveravani izuzeci, sakupljanje đubreta

- Efikasnost

  JIT (Just In Time) prevodioci