

\$ OOP 04

\$ Uvod u OOP

Name: Ognjen Milinković[†]

Date: 9. mart 2021.

[†]ognjen_milinkovic@matf.bg.ac.rs

§ Sadržaj

1. C i strukture

Motivacioni primer

2. Nastanak OOP-a

Softverska kriza

3. Koncepti OOP-a

Objekat

Klasa

Polje

Metod

Enkapsulacija

Nasleđivanje

Polimorfizam

4. OOP u Javi

1. C i strukture

Motivacioni primer

2. Nastanak OOP-a

Softverska kriza

3. Koncepti OOP-a

Objekat

Klasa

Polje

Metod

Enkapsulacija

Nasleđivanje

Polimorfizam

4. OOP u Javi

- > Pravimo kviz
- > Potrebno da čuvamo pitanja
- > Pitanja imaju svoju:
 - * Postavku
 - * 4 ponuđena odgovora
 - * Indikator o tačnom odgovoru (0, 1, 2, 3)
- > Možemo reprezentovati C strukturu

§ Primer jednog pitanja

- > Šta je StringBuilder u Java svetu?
 - * Programski prevodilac
 - * Razvojno okruženje
 - * Najpoznatija knjiga za učenje jezika Java
 - * Klasa

§ Primer - pokušaj 1

```
int main() {  
    char postavka[] = "Sta je StringBuilder u Java svetu?",  
        odgovor1[] = "Programski prevodilac",  
        odgovor2[] = "Razvojno okruzenje",  
        odgovor3[] = "Najpoznatija knjiga za ucenje jezika Java";  
        odgovor4[] = "Klasa";  
  
    printf("%s\n * %s\n * %s\n * %s\n * %s\n",
           postavka, odgovor1, odgovor2,  
           odgovor3, odgovor4);  
  
    return 0;  
}
```

§ Primer - pokušaj 2

```
#define MAX 1024

typedef struct pitanje {
    char postavka[MAX];
    char odgovor1[MAX];
    char odgovor2[MAX];
    char odgovor3[MAX];
    char odgovor4[MAX];
    unsigned int tacan_odgovor;
} Pitanje;
```

§ Primer - pokušaj 2

```
// nastavak

int main() {
    Pitanje p = {
        "Sta je StringBuilder u Java svetu?",
        "Programski prevodilac",
        "Razvojno okruzenje",
        "Najpoznatija knjiga za ucenje jezika Java",
        "Klasa",
        3
    };
    printf("%s\n * %s\n * %s\n * %s\n * %s\n",
           p.postavka, p.odgovor1, p.odgovor2,
           p.odgovor3, p.odgovor4);
}
```



- > Tip sada daje semantički smisao
- > Sve relevantne podatke čuvamo u jednoj strukturi
- > Promene su centralizovane (menjamo strukturu)



- > Verovatno nijedna C funkcija ne zna da obradi naš tip
 - * Moramo implementirati svoje
 - * Na primer za ispis

§ Čemu uvođenje tipova

- > Program bolje modeluje domen
- > Funkcije našem tipu dodaju ponašanje
- > Implementacija funkcija definiše uvedeno ponašanje
- > Tipove ne bi ni uvodili ukoliko oni nemaju karakteristično ponašanje
 - * Koristili bi već postojeće tipove

§ Primer – dodavanje ponašanja

```
void prikazi(const Pitanje * p) {  
    printf("%s\n * %s\n * %s\n * %s\n * %s\n",  
        p->postavka, p->odgovor1, p->odgovor2,  
        p->odgovor3, p->odgovor4);  
}  
  
int main() {  
    Pitanje p = { "Sta je StringBuilder u Java svetu?",  
        "Programski prevodilac", "Razvojno okruzenje",  
        "Najpoznatija knjiga za ucenje jezika Java",  
        "Klasa", 3 };  
    prikazi(&p);  
    return 0;  
}
```

§ Problem

- > Razmislimo o sintaksi prikazi (*&p*)
- > Ako je veći kod nije nam jasno gde se nalazi funkcija
- > Šta je tip za koji je definisana?
- > Da li zaista naše pitanje ima ponašanje?
- > Ili smo ga veštacki napravili?

§ Drugačiji pristup

```
typedef struct pitanje {  
    char postavka[MAX];  
    char odgovor1[MAX];  
    char odgovor2[MAX];  
    char odgovor3[MAX];  
    char odgovor4[MAX];  
    unsigned int tacan_odgovor;  
  
    // Pokazivac na funkciju koja prima  
    // 'const struct pitanje * p' a njen tip je 'void'  
    void (*prikaziPitanje)(const struct pitanje * p);  
} Pitanje;
```

§ Drugačiji pristup

```
void prikazi(const struct pitanje * p) {
    printf("%s\n * %s\n * %s\n * %s\n * %s\n",
           p->postavka, p->odgovor1,
           p->odgovor2, p->odgovor3, p->odgovor4);
}

int main() {
    Pitanje p = {"Sta je StringBuilder u Java svetu?",
                 "Programski prevodilac", "Razvojno okruzenje",
                 "Najpoznatija knjiga za ucenje jezika Java",
                 "Klasa", 3,
                 prikazi
    };
    // Obратiti paznju na sintaksu
    p.prikaziPitanje(&p);
    return 0;
}
```

- > Više ne govorimo o **proceduri** kao u prethodnom primeru
- > prikaziPitanje se sada nalazi **unutar** strukture Pitanje
- > Funkcija je **vezana za konkretno** pitanje
- > Sada pitanje **nudi** korisniku strukture neke usluge:
 - * prikaziPitanje

§ Napomena

Naravno postoje i dalje mane:

- > Zašto prikaziPitanje i dalje mora da prihvati pokazivač?
- > Pri pravljenju pitanja moramo proslediti i pokazivač na funkciju

1. C i strukture

Motivacioni primer

2. Nastanak OOP-a

Softverska kriza

3. Koncepti OOP-a

Objekat

Klasa

Polje

Metod

Enkapsulacija

Nasleđivanje

Polimorfizam

4. OOP u Javi

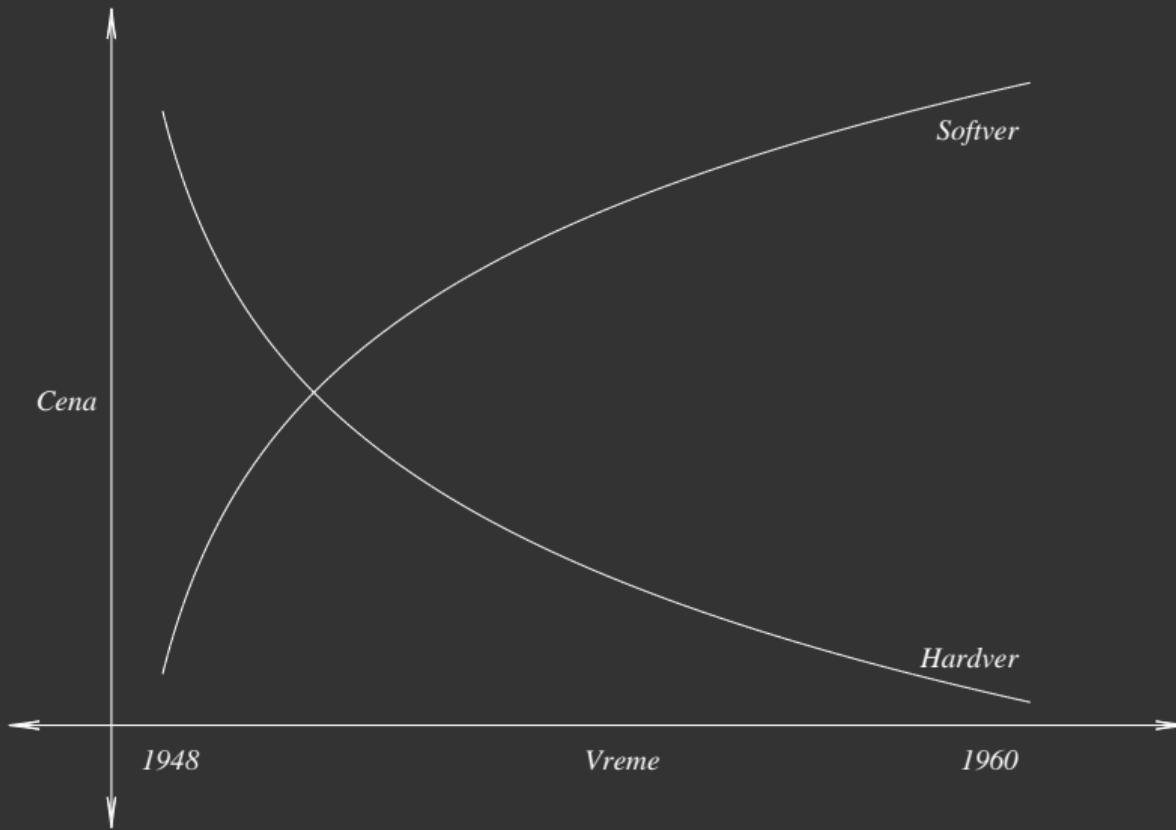
§ Programiranje u početku

- > Programiranje se posmatralo kao **pisanje logike**
- > Opisivali smo procedure koje nešto izračunavaju
- > Ali nismo se previše bavili samim **podacima**
- > Nije iznenađujuće ako se uzme u obzir da je programiranje nastalo kao automatizacija računanja

§ Softverska kriza

- > 60ih i 70ih raste računarska moć
- > Rastu i naši zahtevi od računara i programera
- > Zahteve treba prevesti u kod
- > Kompleksniji zahtevi zahtevaju kompleksnije programe
- > Tu nastaje problem
 - * Tadašnji stil programiranja nedovoljno prilagođen
 - * Industrija ne uspeva da proizvede dovoljno brzo softver
 - * Softver nije u stanju da iskoristi punu moć hardvera

§ Softverska kriza



§ Problemi

Neki od problema:

- > Projekti traju predužno
- > Projekti probijaju budžet
- > Softver je niskog kvaliteta
- > Softver je nedovoljno efikasan
- > Softver troši previše memorije
- > Kod je izuzetno teško održavati
- > Teško dodavati funkcionalnosti u program

- > Objektno-orientisano programiranje¹ nastaje kao novi stil programiranja
- > Rešava navedene probleme (u velikoj meri)
- > Program više ne posmatramo kao niz procedura
- > Program je sada za nas **interakcija između objekata**
- > Pri razvoju softvera se bavimo **strukturom** i **ponašanjem** objekata

¹na dalje OOP

§ Nastanak OOP-a

- > Zvanično² prvi objektni jezik je Simula (nastao sredinom 60ih)
- > Uvodi koncepte objekta, klase, podklase...
- > Jezik inicijalno zamišljen za **izvođenje simulacija**

²Iako je to uvek teško precizirati kod programskih jezika

§ Šta su u stvari neki od naših programa

- > Ako razmislimo, naši programi nekad i jesu u stvari simulacije
 - * Tekst editor - simulira pisanje i brisanje po papiru
 - * Elektronska pošta - simulira slanje i primanje pisama
 - * Muzički plejer - simulira reprodukciju muzike
 - * Video plejer - simulira projekciju videa
 - * TODO menadžer - simulira TODO liste

§ Šta su u stvari neki od naših programa

- > Video igre - nekada zaista simuliraju (deo) realnosti



§ Šta su u stvari neki od naših programa

- > Video igre - a nekada žalimo što neki njihovi delovi ne postoje u realnosti



§ Efekat OOP-a

- > OOP izuzetno doprinoeо načinu razvoja programa
- > Predstavlja prirodan način (paradigmu) programiranja za mnoge situacije
- > U velikoj meri doprinoeо rešavanju softverske krize

1. C i strukture

Motivacioni primer

2. Nastanak OOP-a

Softverska kriza

3. Koncepti OOP-a

Objekat

Klasa

Polje

Metod

Enkapsulacija

Nasleđivanje

Polimorfizam

4. OOP u Javi

§ Koncepti OOP-a

- > Objekat
- > Klasa
- > Polje - opisuje stanje
- > Metod - opisuje ponašanje
- > Enkapsulacija
- > Nasleđivanje
- > Polimorfizam

§ Koncepti OOP-a - Objekti u realnom svetu

Objekti u realnom svetu poseduju:

- > Stanje
- > Ponašanje

Na primer pas:

- > Stanje: ime, boja, rasa
- > Ponašanje: laje, maše repom, spava, gladan, grize

Primer sa početka koji prikazuje Pitanje u jeziku C:

- > Stanje: postavka, odgovor1, ..., odgovor4, tacan_odgovor
- > Ponašanje: prikaziPitanje

Objekat

- > Modeluje objekat iz realnog sveta
- > Ima stanje koje se čuva u **poljima**
- > Ima ponašanje koje se čuva u **metodima**

Polja

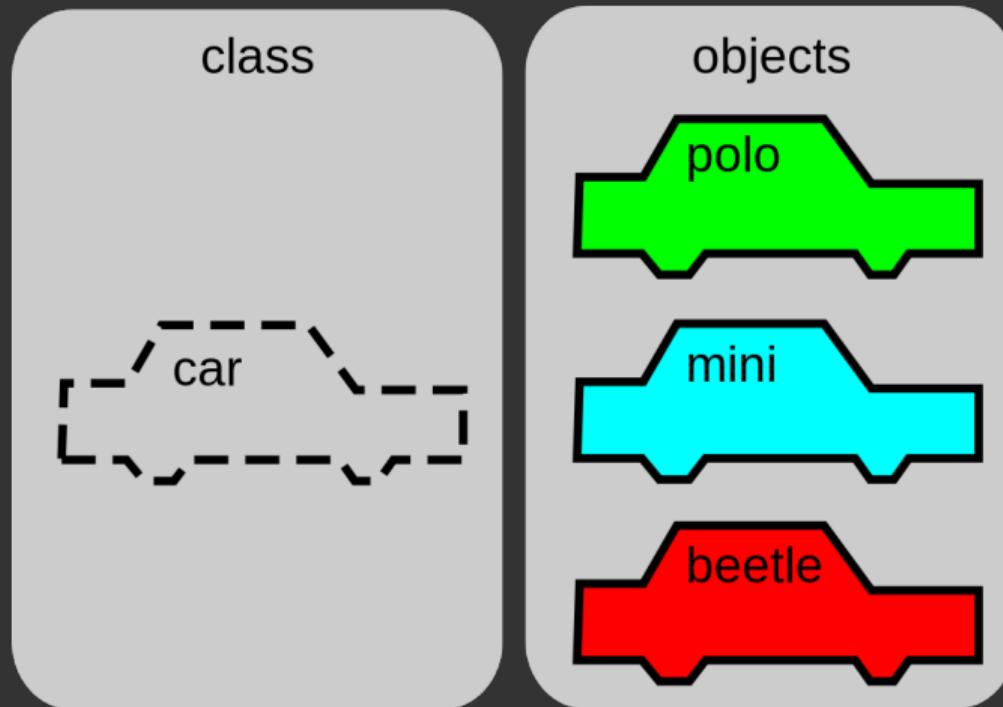
- > Reprezentujemo promenljivama (npr. `String ime;`)

Metodi

- > Reprezentujemo funkcijama (npr. `void masiRepon()`)

Klasa

- > Apstrahuje skup objekata koji ima dovoljno sličnosti
- > Definiše stanje i ponašanje za apstrahovane objekte
- > Objekat predstavlja instancu (primerak) klase



§ Koncepti OOP-a - Polje

Naziva se još:

- > Atribut
- > Svojstvo

Polje:

- > Opisuje stanje objekta
- > U programskim jezicima je to najčešće promenljiva nekog tipa

Metod:

- > Opisuje ponašanje objekta
- > U programskim jezicima je to najčešće funkcija

§ Koncepti OOP-a - Enkapsulacija

- Slika vredi hiljadu reči



§ Enkapsulacija

- > Slika vredi hiljadu reči



§ Enkapsulacija

- > Želimo da od spoljašnjeg sveta sakrijemo sve nepotrebne informacije
- > Spoljašnjem svetu nudimo usluge u obliku funkcija (metoda)
- > Korisnik se ne opterećuje **kako** metod radi, bitno mu je **šta** radi
- > Korisnik klase ne mora poznavati njenu **internu implementaciju**
 - * Implementacija može biti izuzetno složena

§ Nasleđivanje

Zamislimo sledeću hijerarhiju za Zaposlene:

- > Zaposleni
 - * Programer
 - * Junior programer
 - * Senior programer
 - * Dizajner
 - * Grafički dizajner
 - * Direktor
 - * Generalni direktor
 - * Izvršni direktor
 - * Sekretar
- > Da li uočavate neke sličnosti koje poseduju junior i senior programer?
- > Šta je to što ima senior a nema junior?
- > Šta je to što imaju junior i senior a nema (teorijski) niko drugi

§ Nasleđivanje

Nasleđivanje

- > Klasa B nasleđuje klasu A i to:
 - * Klasa B je **istovremeno** A
 - * Klasa B je i šira od klase A
 - * Dodatno ponašanje
 - * Dodatno stanje
 - * Klasa B je **specijalizacija** klase A
 - * Klasa A je **generalizacija** klase B

Primer

- > Junior programer nasleđuje klasu Programer
 - * Junior jeste programer
 - * Junior programer jeste specijalizacija pojma programer
 - * Programer je generalizacija pojmove junior programer i senior programer

§ Nasleđivanje

- > Omogućava nam da lako proširimo funkcionalnosti
- > Prirodno opisuje hijerarhije (jer i jeste hijerarhija)
- > Omogućava hijerarhijski polimorfizam

§ Polimorfizam

- Da li životinje mogu da ispuštaju zvuk?

Polimorfizam

§ Polimorfizam

- > Da li životinje mogu da ispuštaju zvuk?
- > // da

Polimorfizam

§ Polimorfizam

- > Da li životinje mogu da ispuštaju zvuk?
- > // da
- > Da li postoji jedan tip životinje ili više?

Polimorfizam

§ Polimorfizam

- > Da li životinje mogu da ispuštaju zvuk?
- > // da
- > Da li postoji jedan tip životinje ili više?
- > // više

Polimorfizam

§ Polimorfizam

- > Da li životinje mogu da ispuštaju zvuk?
- > // da
- > Da li postoji jedan tip životinje ili više?
- > // više
- > Da li životinje ispuštaju isti zvuk?

Polimorfizam

§ Polimorfizam

- > Da li životinje mogu da ispuštaju zvuk?
- > // da
- > Da li postoji jedan tip životinje ili više?
- > // više
- > Da li životinje ispuštaju isti zvuk?
- > // ne, zavisi od tipa životinje

Polimorfizam

§ Polimorfizam

- > Da li životinje mogu da ispuštaju zvuk?
- > // da
- > Da li postoji jedan tip životinje ili više?
- > // više
- > Da li životinje ispuštaju isti zvuk?
- > // ne, zavisi od tipa životinje

Polimorfizam

- > U zavisnosti od tipa životinje, životinja se drugačije oglašava

§ Polimorfizam

> Više o enkapsulaciji, nasleđivanju i polimorfizmu na narednim časovima

1. C i strukture

Motivacioni primer

2. Nastanak OOP-a

Softverska kriza

3. Koncepti OOP-a

Objekat

Klasa

Polje

Metod

Enkapsulacija

Nasleđivanje

Polimorfizam

4. OOP u Javi

- > Java nastala kao OO jezik
- > Podržava sve navedene koncepte (i šire)

§ Primer klase Pitanje

```
class Pitanje {  
    private String postavka, odgovor1,  
        odgovor2, odgovor3, odgovor4;  
    private int tacanOdgovor;  
  
    // Konstruktor - konstruise objekat klase Pitanje  
    public Pitanje(String postavka, String odgovor1,  
        String odgovor2, String odgovor3,  
        String odgovor4, int tacanOdgovor)  
    {  
        this.postavka = postavka;  
        this.odgovor1 = odgovor1;  
        this.odgovor2 = odgovor2;  
        this.odgovor3 = odgovor3;  
        this.odgovor4 = odgovor4;  
    }  
}
```

§ Primer klase Pitanje

```
// ...  
  
// GET-eri, omogucavaju da procitamo stanje objekta  
public String getPostavka() { return postavka; }  
public String getOdgovor1() { return odgovor1; }  
public String getOdgovor2() { return odgovor2; }  
public String getOdgovor3() { return odgovor3; }  
public String getOdgovor4() { return odgovor4; }  
public int tacanOdgovor() { return tacanOdgovor; }  
  
// Metod koji omogucava da proverimo da li je odgovor tacan  
public boolean proveriOdgovor(int i) {  
    if (i == tacanOdgovor)  
        return true;  
    else  
        return false;  
}
```

§ Primer klase Pitanje

```
// ...  
  
// Metod koji omogucava da dobijemo stringovnu  
// reprezentaciju naseg objekta.  
public String toString() {  
    String t = postavka + "\n";  
    t += "* " + odgovor1 + "\n";  
    t += "* " + odgovor2 + "\n";  
    t += "* " + odgovor3 + "\n";  
    t += "* " + odgovor4;  
    return t;  
}  
  
}
```

Zahvalnica

Ovu prezentaciju je napravio kolega Nemanja Mićović.
Hvala kolegi Mićoviću što se saglasio da koristimo ovu prezentaciju.