



Napredne tehnike programiranja

[Studij informatike](#)

# Napredne tehnike programiranja

## Vježba 5



# Teme



---

- Prava pristupa
- Set i get funkcije
- This pokazivač

# Prava pristupa članovima klase

---



- `public` – članovima klase koji imaju ovo pravo pristupa može se pristupiti izvan klase
- `protected` - članovima klase koji imaju ovo pravo pristupa može se pristupiti samo iz klase i podklasa (klasa koji je nasljeđuju)
- `private` - članovima klase koji imaju ovo pravo pristupa može se pristupiti samo unutar klase (preko funkcija članova)
- Ako se pravo pristupa ne navede eksplicitno podrazumijeva se privatni pristup

# Uobičajena građa klase

---

```
class primjer {  
    private:  
        //ovdje dolaze podatkovni  
        članovi  
  
    public:  
        //ovdje dolaze funkcije  
  
};
```

# Kako oblikujemo klase?

---



- Sakrivamo informacije o internoj reprezentaciji i implementaciji u privatni dio klase (enkapsulacija)
- javno prikazujemo sučelje klase (skup metoda/operacija koje će se primjenjivati na instancama klase)
- Metode koje čine sučelje štite attribute od neželjnih promjena

# Klasa

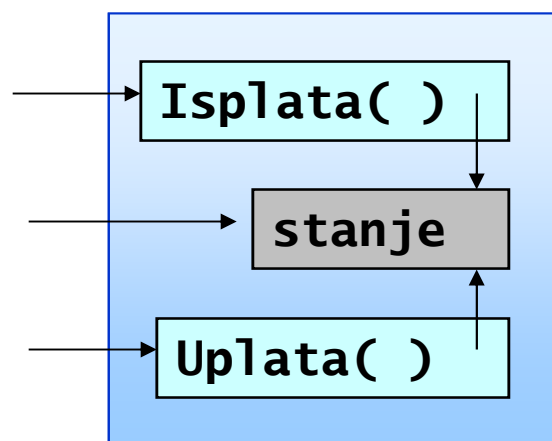
```
class Osoba {  
    private:  
        char JMBG[13+1];  
        char Prezime[40+1];  
        char Ime[40+1];  
        float Placa;  
    public:  
        string GetJMBG();  
        string GetPrezime();  
        string GetIme();  
        float  GetPlaca();  
        void PromijeniPlacu(int posto);  
        ...  
};
```

Stanje

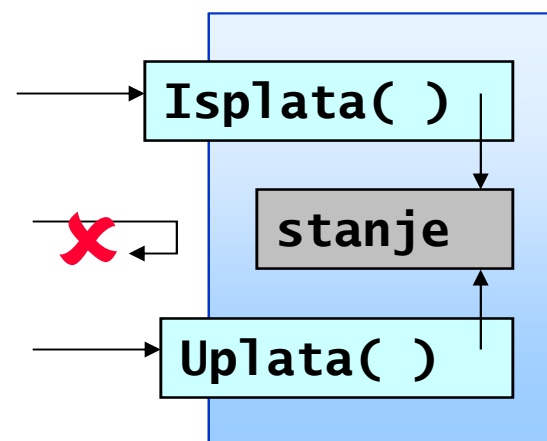
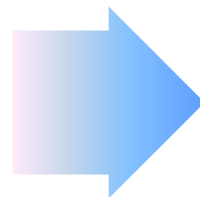
Ponašanje

# Kontrola pristupa

- Metode (funkcije) su obično javne (public), dostupne izvana
- Podaci su obično privatni (private), nedostupni izvana



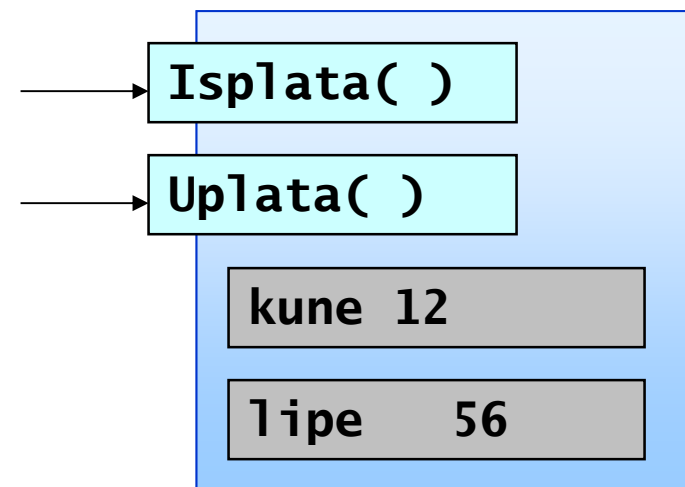
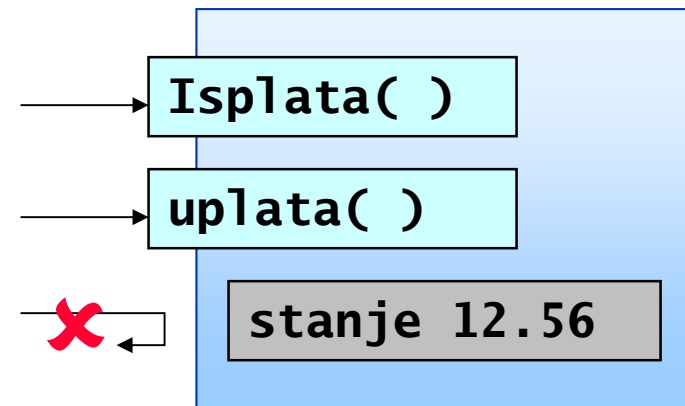
Račun u banci



Račun u banci

# Zašto koristiti enkapsulaciju?

- Omogućava kontrolu
  - Objektom upravljamo samo preko metoda
- Omogućava izmjene
  - U slučaju izmjena u klasi upravljanje objektom ostaje nepromijenjeno





# Definicija klase



- Deklaracija funkcija članica u klasi

```
class Screen {  
    public:  
        void home();  
        void move( int, int );  
        char get();  
        char get( int, int );  
        bool checkRange( int, int );  
};
```

- Funkcije članice klase razlikuju se od “običnih” funkcija po sljedećim svojstvima:
  - Imaju potpun pristup privatnom i javnom dijelu klase
  - Pripadaju dosegu klase

# Definicija klase



- U tijelu klase mogu se nalaziti i definicije funkcija članica

```
class Screen {  
    public:  
        void home() { _cursor = 0; }  
        char get() { return _screen[_cursor]; }  
};
```

- Funkcije članice klase mogu biti i preopterećene
- Preopterećene funkcije su funkcije koje imaju isto ime i istu povratnu vrijednost, ali različitu listu parametara

```
class Screen {  
    public:  
        char get() { return _screen[_cursor]; }  
        char get( int, int );  
};
```

# Definicija klase



---

- Funkcije članice koje su definirane unutar klase, automatski su označene kao `inline`
- Kompajler će pokušati cijelo tijelo funkcije zalijepiti na mjesto poziva
- Tijelo inline funkcije mora biti vidljivo u svakoj datoteci koja koristi inline funkciju

# Set i get funkcije



---

- Pošto su podaci u klasi zaštićeni pristupamo im isključivo preko metoda (funkcija) iz javnog dijela klase
- Uobičajeno je u javni dio klase umetnuti funkcije koje postavljaju vrijednosti atributa u klasi (**set metode**) i funkcije koje vraćaju vrijednost atributa u klasi (**get metode**)

# Funkcije pristupa (set i get)

---

```
class Tocka{
private:
    float dimx;
    float dimy;
public:
    //Setters
    void postavi_x(float x){dimx=x;}
    void postavi_y(float y){dimy=y;}
    //Getters
    float citaj_x(){return dimx;}
    float citaj_y(){return dimy;}
};
```

# Funkcije pristupa (set i get)



---

- Ako klasa ima detaljno razrađeno sučelje (način rada sa objektima) ona ne mora nužno imati set i get funkcije
- Set funkcije mogu potavljati vrijednost jednog ili više atributa
- Get funkcije obično dohvaćaju pojedinačmi atribut
- Set funkcije štite podatke i dozvoljavaju samo one promjene koje su regularne

# Implicitni this pokazivač

---




- Funkcije članice klase imaju implicitno jedan dodatni “skriveni” parametar: `this` pokazivač
- `this` pokazivač je pokazivač na objekt koji je pozvao funkciju članicu klase
- Ne moramo (ne smijemo) sami definirati `this` pokazivač, kompajler ga implicitno automatski dodaje u svaku funkciju članicu

# Implicitni this pokazivač

- U tijelu funkcija članica se ne moramo uvijek pozivati na this pokazivač

```
class Screen {  
    public:  
        short _height, _width;  
        void f() {  
            this->_height = 5; // _height =  
5;  
        }  
}
```



Nije nužno



# Kako rade funkcije članovi



Ovako pišemo funkciju: **A ovako ona stvarno radi:**

```
void Date::set(int d, int  
m, int y) {  
    day = d;  
    month = m;  
    year = y;  
}
```

```
Date d;  
d.set(21, 3, 2007);
```

```
void set(Date* const this,  
int d, int m, int y) {  
    this->day = d;  
    this->month = m;  
    this->year = y;  
}
```

```
Date d;  
Date::set(&d, 2, 30, 2007);
```