Napredne tehnike programiranja

Vježba 9

Teme

- Konstruktori i destruktor kod nasljeđivanja
- Prava pristupa kod nasljeđivanja
- Višestruko nasljeđivanje

Nasljeđivanje – konstruktor i destruktor

- Konstruktori i destruktor se ne nasljeđuju
- Pri inicijalizaciji se pozivaju svi konstruktori iz hijerarhije nasljeđivanja
- Najprije se moraju izvršiti konstruktori baznih klasa
- Tek nakon njih se izvršava konstruktor izvedene klase

Konstruktori i destruktori

- Pri kreiranju objekta izvedene klase poziva se konstruktor te klase, ali i konstruktor osnovne klase.
- U zaglavlju definicije konstruktora izvedene klase, u listi inicijalizatora, moguće je navesti i inicijalizator osnovne klase (argumente poziva konstruktora osnovne klase).
- To se radi navođenjem imena osnovne klase i argumenata poziva konstruktora osnovne klase

Konstruktori i destruktori

- ```
  class Osnovna {
    int bi;
  public:
    Osnovna(int);  // konstruktor osnovne klase
  };
  Osnovna::Osnovna (int i) : bi(i) {/*...*/}
  ```

- ```
  class Izvedena : public Osnovna {
    int di;
  public:
    Izvedena(int);
  };
  ```

- ```
  Izvedena::Izvedena (int i) : Osnovna(i),di(i+1)
  {/*...*/}
  ```
  ```

Redoslijed poziva

- Pri kreiranju objekta izvedene klase redoslijed poziva konstruktora je sljedeći:
  - inicijalizira se podobjekat osnovne klase, pozivom konstruktora osnovne klase;
  - inicijalizuju se podaci članovi, eventualno pozivom njihovih konstruktora, po redoslijedu deklariranja;
  - izvršava se tijelo konstruktora izvedene klase.
- Pri uništavanju objekta, redoslijed poziva destruktora je uvek obratan.

Redoslijed poziva

```
class Element {
   public:
          Element () {cout<<"Konstruktor klase Element .\n";}
          ~Element () {cout<<"Destruktor klase Element .\n";}
   };
class Osnovna {
   public:
          Osnovna() {cout<<"Konstruktor osnovne klase."<<endl;}
          ~Osnovna() {cout<<"Destruktor osnovne klase."<<endl;}
   };

class Izvedena : public Osnovna {
          Element x;
   public:
          Izvedena() {cout<<"Konstruktor izvedene klase."<<endl;}
          ~Izvedena() {cout<<"Destruktor izvedene klase."<<endl;}
   };

   int main () {
          Izvedena d;
   }

   /* Izlaz programa:
        Konstruktor osnovne klase.
        Konstruktor klase Element.
        Konstruktor izvedene klase.
        Destruktor izvedene klase.
        Destruktor klase Element.
        Destruktor osnovne klase.
   */
```

Nasljeđivanje – konstruktor i destruktor

```
class Osoba{
  public:
        string ime;
        string prezime;
        Osoba(string _ime, string _prezime) : ime(_ime),
        prezime(_prezime){
              //...
        }
};

class Student : public Osoba{
  public:
        // prosljeđivanje parametara baznom konstruktoru
        Student(string ime, string prezime) : Osoba(ime, prezime)
{
              //...
        }
};
```

Iako klasa Student nema vlastite podatkovne članove ime i prezime ona te podatke treba predati konstruktoru klase Osoba

Nasljeđivanje – konstruktor i destruktor

```
class Osoba{
 public:
        string ime;
        string prezime;
        Osoba(string _ime, string _prezime) : ime(_ime), prezime(_prezime){
                //...
        }
};
class Student{
  public:
        string JMBAG;
        Student(string jmbag) : JMBAG(jmbag){
                //...
        }
};
class Akademik : public Osoba, public Student{
  public:
        // prosljeđivanje parametara konstruktorima baznih klasa
        Akademik(string ime, string prez, string jmbg):Osoba(ime, prez),
Student(jmbg) {
                //...
        }
};
```

Klasa Akademik ima dvije temeljne klase, pa njen konstruktor treba pozvati konstruktore obe temeljne klase

Nasljeđivanje - primjer

```
class Osnovna {
       int i;              // privatni podatak clan osnovne klase

   public:
      void f(); // javna funkcija clanica osnovne klase
   };

class Izvedena : public Osnovna {
       int j;              // privatni podatak clan izvedene klase

   public:
      void g(); // javna funkcija clanica izvedene klase
   };

void main () {
     Osnovna b;
     Izvedena d;
     b.f();
     b.g();    //GRESKA: g je funkcija izvedene klase, a b je objekat osnovne
     d.f();          // objekat izvedene klase d ima i funkciju f,
     d.g();        // i funkciju g
   }
```

Nasljeđivanje

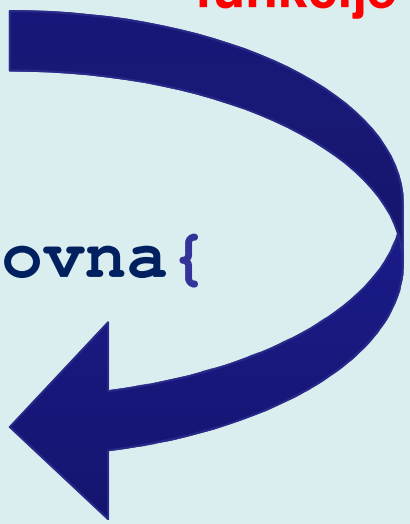
Svojstva izvedene klase:

- Ona nasljeđuje varijable i metode svoje nadređene klase tj. klase koju nasljeđuje
- Ima svoje nove varijable i metode koje ispunjavaju njene specifične zahtjeve
- Izvedena klasa nasljeđuje sva svojstva i mogućnosti nadređene klase

Nasljeđivanje

- Primjer nasljeđivanja:

```
class Osnovna{
  public;
    Osnovna() { a=0; }
    int a;
}
class Izvedena : public Osnovna{
  public:
    Izvedena() {b = 0}
    int b;
}
```



Izvedena klasa dobiva podatke i funkcije iz osnovne klase

Nasljeđivanje

- Iako izvedena klasa nasljeđuje sve elemente osnovne klase to ne znači da su joj svi ti elementi direktno dostupni
- Dostupnost elemenata iz osnovne klase ovisi o pravu pristupa tom elementu u osnovnoj klasi (private, protected, public) i o pravu pristupa kod nasljeđivanja (private, protected, public)

Prava pristupa

```
class Osnovna{
 private;
    int a,b;
    int funkcija1();
  protected;
    int c,d;
    int funkcija2();
 public;
    int e,f;
    int funkcija3();
}
```

Ovim elementima može se pristupiti iz klase, iz izvedenih klasa i izvan klase

Prava pristupa

```
class Osnovna{
 private;
    int a,b;
    int funkcija1();
  protected;
    int c,d;
    int funkcija2();
 public;
    int e,f;
    int funkcija3();
}
```

Ovim elementima može se pristupiti iz klase i iz izvedenih klasa

Prava pristupa

```
class Osnovna{
  private;
     int a,b;
     int funkcija1();
   protected;
     int c,d;
     int funkcija2();
  public;
     int e,f;
     int funkcija3();
}
```

Ovim elementima može se pristupiti samo unutar klase

Nasljeđivanje

- Ključne riječi public, private i protected se koriste i kao specifikatori tipa naslijeđivanja
- Pravo pristupa članu u izvedenoj klasi određeno je pravom pristupa u osnovnoj klasi te vrstom nasljeđivanja.
- Kako postoje tri prava pristupa te tri moguća tipa nasljeđivanja, imamo ukupno devet mogućih kombinacija.

Načini izvođenja nasljeđivanja

- Javno – prava pristupa se ne mijenjaju u odnosu na osnovnu klasu (javni ostaju javni, zaštićeni ostaju zaštićeni)
- Zaštićeno – svi nasljeđeni članovi postaju zaštićeni u izvedenoj klasi
- Privatno (podrazumijevano ako se ne navede drugo) – svi nasljeđeni članovi postaju privatni u izvedenoj klasi

```
class A{};
class B: public A {};
class C: protected A {};
class D: private A{};
```

Načini izvođenja nasljeđivanja

```
class A{};
class B: public A {};
class C: protected A {};
class D: private A{};
```

Pristup u osnovnoj klasi	Pristupačnost u javno izvedenoj klasi	Pristupačnost u zaštićeno izvedenoj klasi	Pristupačnost u privatno izvedenoj klasi
Nepristupačan	Nepristupačan	Nepristupačan	Nepristupačan
Privatni	Nepristupačan	Nepristupačan	Nepristupačan
Zaštićeni	Zaštićeni	Zaštićeni	Privatni
Javni	Javni	Zaštićeni	Privatni

Primjer

```
class Glavna {
     public:
          int a;
     protected:
          int b;
     private:
          int c;
}
class A: public Glavna{}
class B: protected Glavna{}
class C: private Glavna{}
```

Prikrivanje i prava pristupa

```
class Osnovna {
     int p;
   protected:
     int z;
   public:
     int j;
   };

class Izvedena : public Osnovna {
   public:
     void write(int x) {
        j=z=x;   // moze pristupiti javnom i zasticenom clanu,
        p=x;     // ! GRESKA: privatnom clanu se ne moze pristupiti
      }
   };

void f() {
     Osnovna b;
     b.z=5; //odavde se ne moze pristupiti zasticenom clanu
   }
```

Višestruko nasljeđivanje

- Višestruko nasljeđivanje (engl. multiple inheritance) omogućava nasljeđivanje osobina više osnovnih klasa
- Klasa se deklarirane kao nasljednik više klasa tako što se u zaglavlju deklaracije navode osnovne klase.
- Ispred svake osnovne klase treba staviti oznaku public, da bi izvedena klasa nasljeđivala prava pristupa članovima.

```
class Izvedena: public Osnovna1, public Osnovna2, private
Osnovna3 {/* ... */};.
```

Konstruktori i destruktori kod višestrukog nasljeđivanja

- Sva pravila o nasljeđenim članovima vrijede i kod višestrukog nasljeđivanja.
- Konstruktori osnovnih klasa se pozivaju prije konstruktora članova izvedene klase i konstruktora izvedene klase.
- Konstruktori osnovnih klasa se pozivaju po redoslijedu deklariranja.
- Destruktori osnovnih klasa se izvršavaju na kraju, poslije destruktora izvedene klase i destruktora članova.

Zadatak

- Zadatak: Neka je zadana slijedeća hijerarhija klasa (pri čemu svaka klasa u hijerarhiji ima definiran defaultni konstruktor):

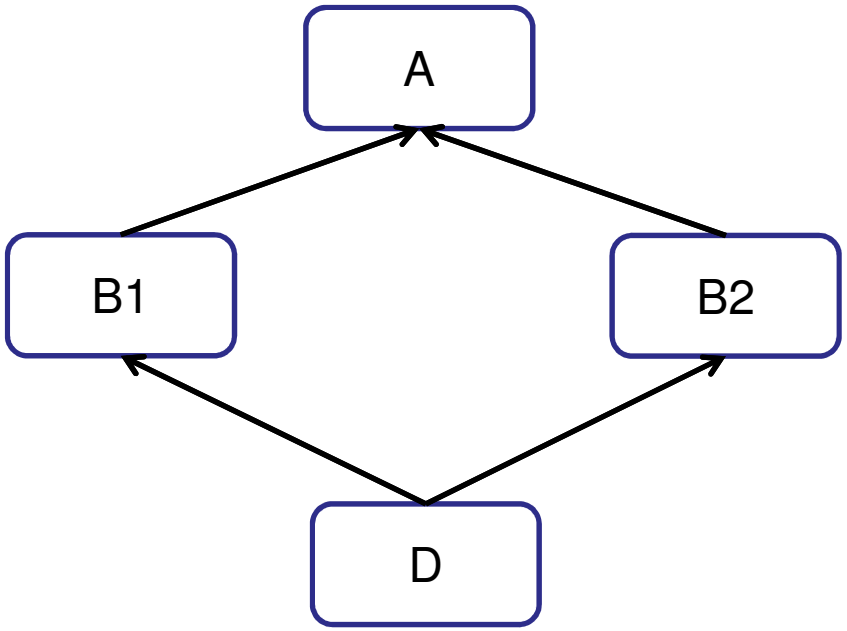
```
class A { ... };
class B : public A { ... };
class C : public B { ... };
class X { ... };
class Y { ... };
class Z : public X, public Y { ... };
class M : public C, public Z { ... };
```

- Odredite redoslijed pozivanja konstruktora prilikom definicije slijedećeg objekta:

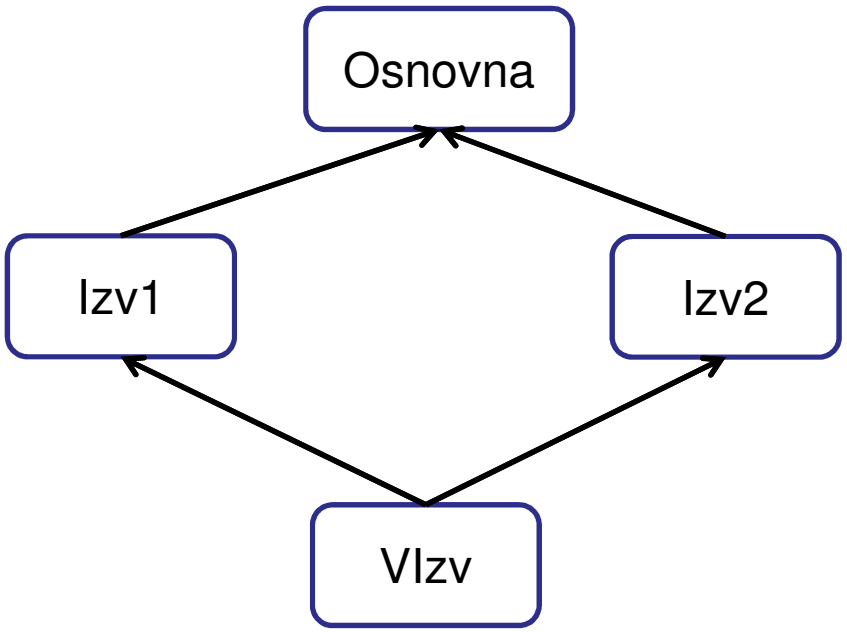
```
M m;
```

Problem višestrukog nasljeđivanja

- Kod višestrukog nasljeđivanja može doći do konflikta
- Klasa D nasljeđuje klase B1 i B1, koje imaju istu temeljnu klasu A



Problem višestrukog nasljeđivanja




```
class Osnovna {
      public: int x;
};
class Izv1 : public Osnovna{
/*...*/};

class Izv2 : public Osnovna{
/*...*/};

class VIzv : public Izv1,
public Izv2 {
      public: int fun();
};
```

Problem višestrukog nasljeđivanja

```
class Osnovna {
      public: int x;
};
class Izv1 : public Osnovna{ /*...*/};

class Izv2 : public Osnovna{ /*...*/};

class VIzv : public Izv1,
public Izv2 {
      public: int fun();
};
void VIzv :: fun() {
x=2; //dvosmislenost,  Izv1::X  ili Izv2::X , greška !!!
Izv1::x = Izv2::x + 1;  // ispravno
}
```

Višestruki podobjekti

- Ako klasa ima višestruku nevirtuelnu osnovnu klasu **X** onda će objekti te klase imati više podobjekata tipa **X**
- Članovima osnovne klase **X** može se pristupiti nedvosmislenim navođenjem njihove pripadnosti, korišćenjem operatora **::**

```
VIzv :: Izv1 :: Osnovna :: X
```

- Konverzija pokazivača tj reference na izvedenu klasu u pokazivač/referencu na višestruku osnovnu klasu može se izvršiti samo ako je nedvosmislena
- Nedvosmislenost znači da ne postoje dva ili više entiteta koja odgovaraju navedenom imenu

```
void VIzv :: fun()
{
     x=2;                    // dvosmislenost,  Izv1::X
ili Izv2::X  ,  greška !!!
     Izv1::x = Izv2::x + 1;  // ispravno
}
```