# Napredne tehnike programiranja

## Vježba 10

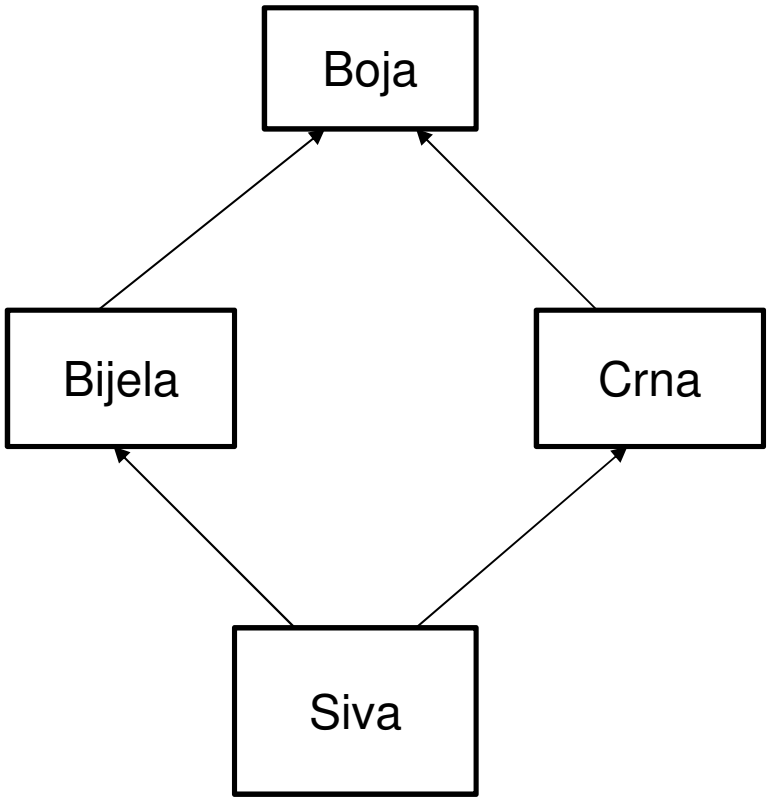
# Teme

- Virtualno nasljeđivanje
- Virtualne funkcije
- Apstraktne klase
- Polimorfizam

# Virtualno nasljeđivanje

- Problem s višestrukim nasljeđivanjem klasa (dijamantna hijerarhijska struktura) možemo rješiti virtualnim nasljeđivanjem
- Kod definiranja nasljeđivanja ispred tipa nasljeđivanja stavimo ključnu riječ `virtual`
- Sada u izvedenoj klasi postoji samo jedna kopija člana klase iz vrha dijamantne strukture

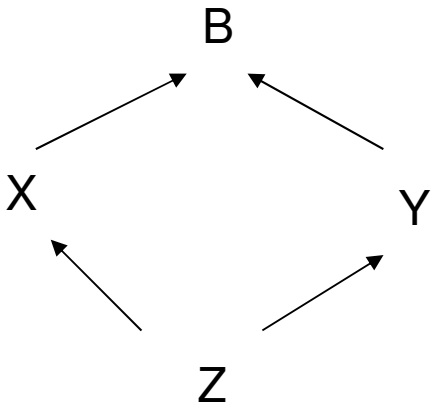
# Višestruki podobjekti




```
class Boja{
  public:
       string naziv;
};
class Bijela : virtual public
Boja{
  public:
       //...
};
class Crna : virtual public
Boja{
  public:
       //...
};
class Siva : public Bijela,
public Crna{
  public:
       //...
};
```

# Virtualne osnovne klase

```
class B {/*...*/};
class X : virtual public B {/*...*/};
class Y : virtual public B {/*...*/};
class Z : public X, public Y {/*...*/};
```




- Klasa Z ima samo jedan skup članova klase B.
- Moraju se i X i Y virtualno izvesti iz B;
- Ako je samo jedna izvedena virtualno ostaju dva skupa članova.
- Konstruktori virtualnih osnovnih klasa se pozivaju prije konstruktora nevirtualnih osnovnih klasa.
- Svi konstruktori osnovnih klasa se pozivaju prije konstruktora članova i konstruktora izvedene klase.

# Virtualne funkcije

Pokazivač tipa nadređene klase može pokazivati na objekt podređene klase

```
class A{
  public:
        void ispis(){ cout<<"A"<<endl; }
};
class B : public A {
  public:
        int x;
        void ispis(){ cout<<"B"<<endl; }
};
int main(){
        B objekt;
        A *pok;
        pok=&objekt;//pokazivač osnovne klase pokazuje na objekt izvedene
klase
        return 0;
}
```

# Virtualne funkcije

```
class A{
  public:
       void ispis(){ cout<<"A"<<endl; }
};
class B : public A {
  public:
       int x;
       void ispis(){ cout<<"B"<<endl; }
};
int main(){
       B objekt;
       A *pok;
       pok=&objekt;
       objekt.ispis();  // ispisuje B
       pok->ispis();    // ispisuje A
       return 1;
}
```

# Virtualne funkcije

```
class A{
  public:
       virtual void ispis(){ cout<<"A"<<endl; }
};
class B : public A {
  public:
       int x;
       void ispis(){ cout<<"B"<<endl; }
};
int main(){
       B objekt;
       A *pok;
       pok=&objekt;
       objekt.ispis();  // ispisuje B
       pok->ispis();    // ispisuje B
       return 1;
}
```

# Virtualne funkcije

Ako ispred deklaracije metode osnovne klase koja će biti redefinirana u nekoj izvedenoj klasi stavimo ključnu riječ `virtual` tada će prevodilac pozvati istoimenu metodu izvedene klase, čak i kada je pokazivač, koji pokazuje na objekt izvedene klase, iz osnovne klase

```
OsnovnaK *p;
IzvedenaK1 a;
p=&a;      //pokazivač  tipa osnovne klase
pokazuje na objekt izvedene klase
```

# Čiste virtualne funkcije

- C++ podržava kreiranje čistih virtualnih funkcija
- Čista virtualna funkcija se kreira tako da se izjednači sa nulom

```
virtual void funkcija( ) = 0;
```

- U podklasi mora biti kreirana implementacija čiste virtualne funkcije (inače i ona postaje virtualna)

# Virtualne funkcije i apstraktne klase

- Virtualne funkcije ne moraju imati tijelo
- U tom slučaju se one zovu čiste virtualne funkcije
- Čistoj virtualnoj funkciji se na kraju deklaracije dodjeljuje vrijednost 0
- Tada prevoditelj zna da ona neće imati tijelo, a time i klasa u kojoj se ona nalazi automatski postaje apstraktna
- Klasa u kojoj se nalazi barem jedna takva funkcija se naziva apstraktna klasa
- Ne može se kreirati objekt koji pripada apstraktnoj klasi

# Virtualne funkcije i apstraktne klase

```
// Apstraktna klasa
class Osoba{
  public:
      // čista virtualna funkcija
      virtual void opisPosla() = 0;
};

class Vozac : public Osoba{
  public:
      // implementacija metode opisPosla u klasi
      Vozac!
      void opisPosla(){
            cout << "Vozac!" << endl;
      }
};
```

# Virtualne funkcije i apstraktne klase

- Apstraktne klase su klase u kojima je definirana bar jedna čista virtualna funkcija
- Čista virtualna funkcija se označava tako da se iza deklaracije funkcije napiše = 0, tj
  `virtual deklaracija_funkcije() = 0;`
- Klase koje sadrže čiste virtualne funkcije ne mogu se koristiti za deklaraciju objekata, ali se pomoću njih može deklarirati pokazivač na objekte ili argument funkcije koji je referenca objekta.

# Virtualne funkcije i apstraktne klase

```
class Osoba{// Apstraktna klasa
      public:
            // čista virtualna funkcija
            virtual void opisPosla() = 0;
};
class Vozac : public Osoba{
      public:
      // implementacija opisPosla u klasi Vozac!
      void opisPosla(){
            cout << "Vozac!" << endl;
      }
};
```

# Apstraktna klasa

- Apstraktna klasa je ona koja ima bar jednu virtualnu funkciju
- Ne može se kreirati objekt koji pripada apstraktnoj klasi
- Može se kreirati pokazivač na objekt apstraktne klase

# Polimorfizam

- Polimorfizam je svojstvo promjenjivosti oblika
- Kaže se da je polimorfan onaj program koji je napisan tako da se programski algoritami ili funkcije mogu primijeniti na objekte različitih oblika
- U tu se svrhu koristi mehanizam virtualnih funkcija

# Polimorfizam

- Nasljeđivanjem klasa možemo konstruirati hijerarhijsku strukturu koja će nam poslužiti za lakše oblikovanje programskog koda
- Imamo li više izvedenih klasa, možemo pojednostaviti rukovanje s njima koristeći pokazivač ili referencu na osnovnu klasu
- Pokazivač tipa osnove klase može, osim na objekte osnovne klase, pokazivati i na bilo koji objekt izvedenih klasa
- Ovisno o vrsti objekta na koji pokazuje, prevodilac može različito interpretirati upotrebu takvog pokazivača (polimorfizam)

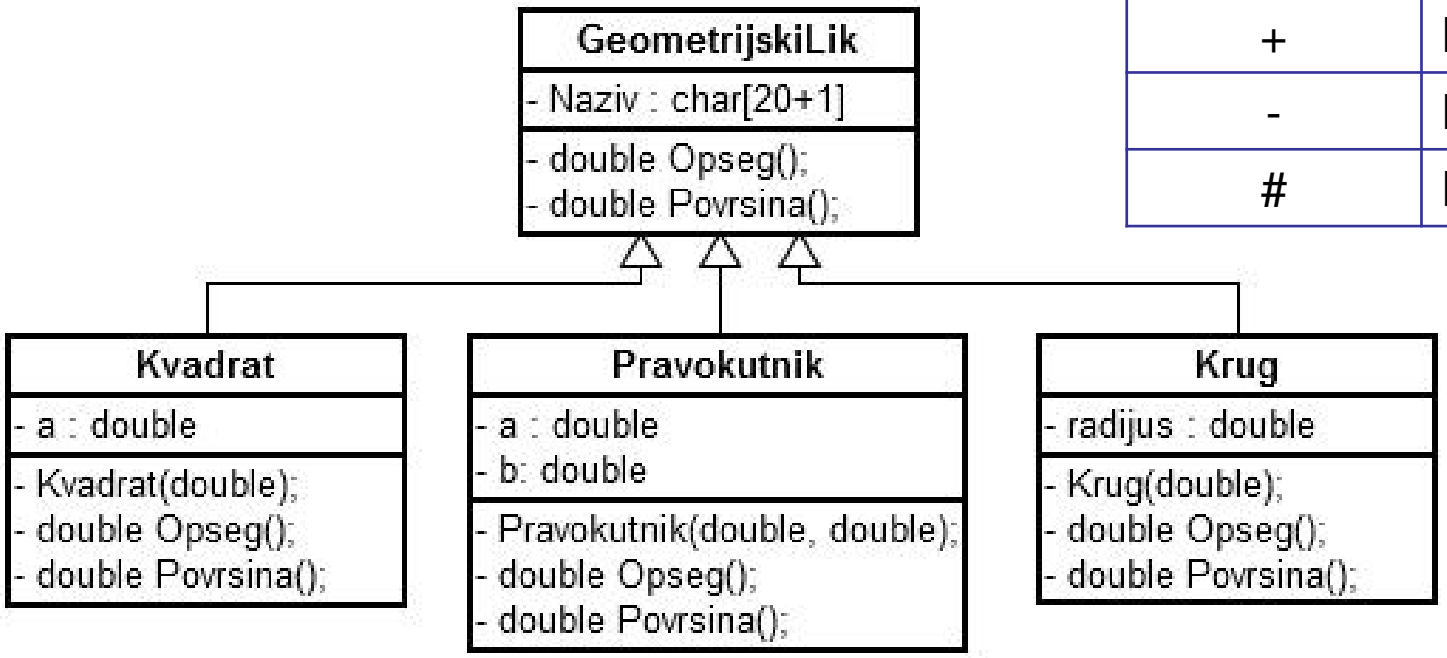
# Polimorfizam

```
class Zivotinja{
  public:
        virtual void glasanje()=0;
};

class Macka : public Zivotinja{
  public:
        void glasanje() {
                cout<<"mijau mijau
                ..."<<endl;
        }
};

class Pas : public Zivotinja{
  public:
        void glasanje() {
                cout<<"vau vau
                ..."<<endl;
        }
};

int main(){
        Zivotinja *pok;
        Macka maca;
        Pas peso;
        pok=&maca;
        pok->glasanje();//mijau mijau
        pok=&peso;
        pok->glasanje(); // vau vau
        system("pause");
        return 0;
}
```

# Zadatak 1 – apstraktna klasa

- Napišite klase prema slijedećem dijagramu klasa
- Napomena: u zadatku sve funkcije trebaju imati public pravo pristupa (a ne private kako je prikazano na dijagramu)

| Pravo pristupa | |
|---|---|
| + | Public |
| - | Private |
| # | Protected |



# Zadatak 2 – polimorfizam

Neka su dane 3 klase koje opisuju geometrijski `lik`, `trokut`, `krug` i `pravilni poligon` kako je prikazano u kodu. Za svaku klasu implementirajte sljedeće:

**`Klasa Lik`:**

a)Implementirajte funkciju status() koja će ispisivati informacije o objektu koji je poziva u smislu radijusa upisane kružnice, površine i opsega. To napravite tako da unutar metode pozivate odgovarajuće virtualne funkcije.

**`Klasa Trokut:`**

a)Definirajte konstruktor koji prima referencu na 3 točke A,B,C i postavlja ih kao vrhove trokuta.

b)Definirajte virtualnu funkciju radiusUpKruz() koja vraća duljinu radijusa upisane kružnice trokuta.

c)Implementirajte virtualne funkcije za računanje opsega i površine trokuta.

**`Klasa Krug:`**

a)Definirajte konstruktor koji prima referencu na točku središta S i duljinu radijusa r.

b)Definirajte virtualnu funkciju radiusUpKruz() koja vraća radijus kruga.

c)Implementirajte virtualne funkcije za računanje opsega i površine kruga.

**`Klasa PravilniPoligon:`**

a)Definirajte konstruktor koji prima pokazivač na polje od N podataka klase Tocka i postavlja vrhove mnogokuta koristeći točke iz polja. Točke moraju tvoriti pravilni N-terokut.

b)Definirajte virtualnu funkciju radiusUpKruz() koja vraća radijus upisane kružnice pravilnog N-terokuta.

c)Implementirajte virtualne funkcije za računanje opsega i površine pravilnog mnogokuta.

**`Glavni program:`**

a)Program testirajte tako da napravite barem jedan objekt svake klase.

b)Pokažite primjenu pozivanja virtualnih funkcija iz klasa koristeći pokazivač tipa bazne klase.

# Zadatak 2 – polimorfizam (kod)

```
class Tocka{
  public:
    float x,y;
    Tocka();
    Tocka(float x, float y);
};
class Lik{
  public:
    void status();
    virtual float povrsina() = 0;
    virtual float opseg() = 0;
    virtual float radiusUpKruz() = 0;
};
class Trokut : public Lik{
  private:
    Tocka A,B,C;
  public:
    Trokut();
    Trokut(Tocka &A, Tocka &B, Tocka &C);
};
class Krug : public Lik{
  private:
    Tocka S;
    float radius;
  public:
    Krug();
    Krug(Tocka &S,float r);
};
class PravilniPoligon : public Lik{
  private:
    int N;
    Tocka *vrhovi;
  public:
    PravilniPoligon();
    PravilniPoligon(Tocka *vrhovi, int N);
};
```