6. Principi polimorfizma

Na prethodnim vježbama obrađene su dvije značajke koje svrstavaju C++ jezik u objektno orijentiranu domenu:

- enkapsulacija – objedinjavanje podataka i funkcija koje manipuliraju tim podacima
- nasljeđivanje – stvaranje hijerarhije razreda

Polimorfizam je treće važno svojstvo koje svaki ozbiljni objektno orijentirani jezik mora podržavati. Ono omogućava definiranje operacija koje su ovisne o tipu. Svaki objekt izvedenog razreda može se promatrati i kao objekt bilo kojeg javnog osnovnog razreda. Time se ne narušava integritet objekta jer su svi članovi osnovnog razreda prisutni i u objektima izvedenog razreda pa im se bez problema može i pristupiti.

```
class GraphObject
{
public:
      //..
      void Draw() {}
};

class Line : public GraphObject
{
public:
      //..
      void Draw();
};

Line *pointLine = new Line;
GraphObject *pointGraph = pointLine;

pointLine->Draw();            // poziva se Line::Draw()
pointGraph->Draw();           // poziva se GraphObject::Draw()
```

Ovako napisan program narušava objektno orijentirani pristup programiranju. Operacija crtanja postaje ovisna o načinu poziva.

Zadatak za vježbu:

1. Prepravite program iz vježbe 5 tako da napravite polje pokazivača na osnovni objekt (`GraphObject`) koje će po potrebi pokazivati na objekte izvedenih razreda.

```
GraphObject *qObject[???];
//…
qObject[0] = new Line;
qObject[1] = new ElipseArc;

for(i = 0; i < ???; i++)
```

```
qObject[i]->Draw();
```

Objasnite zašto program ne radi kako treba? Koja metoda se poziva pri crtanju objekta: npr. Line::Draw() ili GraphObject::Draw() ?

6.1. Virtualni funkcijski članovi

Osnovna ideja virtualnih funkcijskih članova je da se funkcijski članovi za koje želimo dinamičko povezivanje označe prilikom deklaracije razreda. To se čini navođenjem ključne riječi `virtual`, a takvi članovi se nazivaju `virtualnim funkcijskim članovima`. Prevoditelj automatski održava tablicu virtualnih članova koja se pohranjuje u memoriju zajedno sa samim objektom. Prilikom poziva člana, prevoditelj će potražiti adresu člana u tablici koja je privezana uz objekt te na taj način pozvati član.

Zadatak za vježbu:

2. Modificirajte razred `GraphObject` i izvedene razrede tako da funkcijske članove `Draw()`, `Erase()`, `Translate()` i `Rotate()` označite virtualnima.

Napomena: Ključnu riječ `virtual` ne treba pisati kada se metoda piše izvan razreda, npr.:

```
void Line::Draw()
{
      //..
}
```

Prevoditelj će sve virtualne članove pohraniti u tablicu virtualnih članova. Ona se u literaturi često označava skraćenicom `vtable`. Toj tablici se ne može direktno pristupiti.. Svaki objekt sadržavat će skriveni pokazivač `vptr` na virtualnu tablicu. Poziv preko virtualne tablice se zove `dinamički` ili `virtualni poziv`. Za nevirtualne članove se kaže da se pozivaju `statički`.

Zadatak za vježbu:

3. Napišite slijedeći C++ program i pokrenite ga:

```
class A
{
public:
      void func() { cout << "A::func()" << endl; }
};

class B : public A
{
      virtual void func(){ cout << "B::func()" << endl; }
};

class C : public B
{
      virtual void func(){ cout << "C::func()" << endl; }
};

main()
{
      C objc;
      A *pA = &objc;
      B *pB = &objc;
```

```
        pA->func();
        pB->func();
    }
```

Objasnite što se događa pri prvom i drugom pozivu funkcije func().

Funkcijski članovi `Draw()`, `Translate()` i `Rotate()` u razredu `GraphObject` nemaju neko stvarno značenje. Njihova je jedina zadaća definiranje javnog sučelja koje će biti detaljno specificirano tek u izvedenim razredima. Također nema smisla stvarati objekt razreda `GraphObject` jer taj razred služi samo kao temelj za nasljeđivanje. Takvi razredi se u C++ terminologiji zovu `apstraktni razredi`, dok se funkcijski članovi koji služe samo za definiciju javnog sučelja nazivaju `čisti virtualni funkcijski članovi`.

Zadatak za vježbu:

4. Napravite redeklaraciju razreda GraphObject tako da postane apstraktan razred.

```
class GraphObject
{
private:
      int colour;
public:
      void SetColour(int newc) { colour = newc; }
      int GetColour() { return colour; }
      virtual void Draw() = 0;
      virtual void Erase() = 0;
      virtual void Translate(int, int) = 0;
      void Move(int px, inr py);
};
```

Probajte deklarirati objekt tipa `GraphObject`.

Zadatak za vježbu:

5. Ponovo iskoristite primjer iz zadatka 1. i redefiniran osnovni razred `GraphObject` i napišite C++ program. Ubacite u razred `GraphObject` virtualni destruktor. Isto to napravite u izvedenim razredima.

6.2. Virtualni osnovni razred

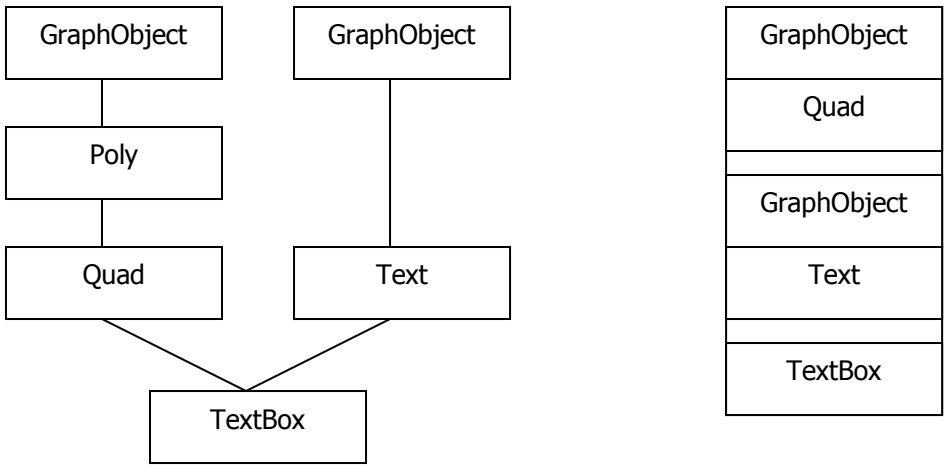
Postoje slučajevi kada se neki osnovni razred može proslijediti izvedenom razredu više no jednom. Kao primjer možemo uzeti grafičke programe u kojima se može ispisati tekst na ekran. U hijerarhiju grafičkih objekata dodat ćemo novi razred `Text` koji će opisivati takst na bilo kojem dijelu ekrana.

```
class Text : public GrafObject
{
private:
      string tmsg;
public:
      virtual void SetText(string newt);
      virtual void Draw();
      virtual void Erase();
      virtual void Translate(int, int);
      virtual ~Text();
};
```

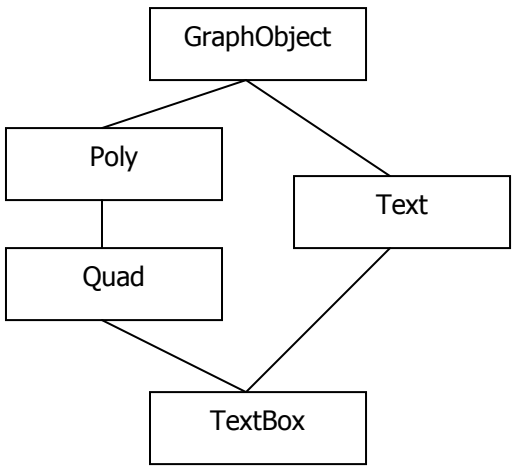
Ponekada je potrebno tekst uokviriti pravokutnikom. Bilo bi praktično stvoriti novi razred `TextBox` koji bi definirao takav objekt. Takav objekt je neka vrsta križanca između objekata `Text` i `Quad`. Jedno od mogućih rješenja je iskoristiti mehanizme višestrukog nasljeđivanja:

```
class TextBox : public Quad, public Text
{
      //…
};
```

Ovo rješenje je elegantno ali pogledajmo što smo zapravo dobili. Razredi `Quad` i `Text` nasljeđuju od razreda `GraphObject`, što znači da sadrže po jedan podobjekt `GrafObject`.



U nekim primjenama pojavljivanje osnovnog razreda više puta u toku nasljeđivanja je poželjno. U ovom slučaju je to pogrešno. Razred `GraphObject` predstavlja osnovni razred za sve grafičke objekte u hijerarhiji. Kako je `TextBox` u cjelini samo grafički objekt, bilo bi logično da taj razred ima razred `GraphObject` proslijeđen samo jednom. Tako oblikovano hijerarhijsko stablo bi izgledalo ovako:



Mehanizam virtualnih osnovnih razreda omogućava definiranje osnovnih razreda koji se dijele između više izvedenih razreda te se prilikom nasljeđivanja izvedenom razredu prosljeđuju samo jednom.

Neki osnovni razred može se učiniti osnovnim virtualnim razredom tako da se prilikom nasljeđivanja u listi nasljeđivanja ispred naziva razreda umetne ključna riječ `virtual`. Redoslijed ključnih riječi `public`, `private`, `protected` i `virtual` pritom nije bitan. Razred se ne definira virtualnim prilikom njegove deklaracije, nego prilikom nasljeđivanja. To znači da se deklaracija razreda `GraphObject` ne mijenja.

```
class Poly : public virtual GraphObject
{
      //…
};

class Quad : public Poly
{
      //…
};

class Text : public virtual GraphObject
{
      //…
};

class TextBox : public Quad, public Text
{
      //…
};
```

Kako su razredi `Poly` (a preko njega i razred `Quad`) i `Text` definirali razred `GraphObject` kao virtualni osnovni razred, razred `TextBox` će sada imati samo jedan podobjekt razreda `GraphObject`.

Zadatak za vježbu:

6. Prepravite vaše programe tako da imaju pravilno napisano virtualno nasljeđivanje iz razreda `GraphObject`. Napišite program koji će imati slijedeće razrede: `GraphObject`, `Poly`, `Quad`, `Text`, `TextBox`.