5. Nasljeđivanje

Osnovna ideja nasljeđivanja je da se prilikom razvoja identificiraju razredi koji imaju sličnu funkcionalnost, te da se u izvedenim razredima samo redefiniraju specifična svojstva, dok se preostala svojstva nasljeđuju u nepromijenjenom obliku.

Uzmimo kao primjer jednostavan program za vektorsko crtanje koji može na radnoj plohi imati različite grafičke elemente. Ograničimo skup objekata na linije, elipsine isječke, krugove, poligone i pravokutnike. Ako detaljnije razmotrimo dani skup objekata, ustanovit ćemo da svi oni imaju niz zajedničkih svojstava: svaki objekt se može nacrtati na željenom području ekrana, može ga se translatirati za neki vektor te rotirati oko neke točke za željeni kut. Također, svaki objekt može biti nacrtan u određenoj boji koja se po želji može mijenjati.

No svaki od tih objekata ima i svoja specifična svojstva koja u potpunosti opisuju upravo njega. Linija ima početnu i završnu točku. Elipsin isječak ima središte, veliku i malu poluos te početni i završni kut isječka. Poligon ima niz točaka koje određuju njegove vrhove. Operacija crtanja će za svaki od tih objekata biti obavljena na različit način što vrijedi i za operacije translacije i rotacije.

Navedene objekte implementirat ćemo pomoću hijerarhijskog stabla. Kao osnovni razred definirat ćemo razred `GraphObject` u kojem će biti definirana zajednička svojstva svih objekata bez navođenja kako se pojedina operacija mora obaviti.Takav razred koji postoji samo zato da bi ga se moglo naslijediti zove se `apstraktni razred`. Ostali razredi će naslijediti taj razred i uvesti nova svojstva potrebna za opisivanje objekata i redefinirati postojeća u skladu sa samim objektom.

```
class GraphObject
{
private:
        int colour;
public:
        void SetColour(int newcolour) { colour = newcolour; }
        int GetColour() { return colour; }
        void Draw() {}
        void Translate(int, int) {}
        void Rotate(int, int, int) {}
};
```

Funkcijski članovi `Translate()` i `Rotate()` ne mogu biti definirani te njihovo tijelo nije specificirano. Nasljeđivanje je postupak pri kojem se na osnovu već postojećeg osnovnog razreda definira novi izvedeni razred. Objekti izvedenog razreda sadrže sve funkcijske i podatkovne članove osnovnog razreda a mogu im se dodati i nova svojstva. Nasljeđivanje se specificira tako da se u deklaraciji razreda iza naziva razreda navede lista osnovnih razreda, kao u slijedećem primjeru:

```
class Line : public GraphObject {};
```

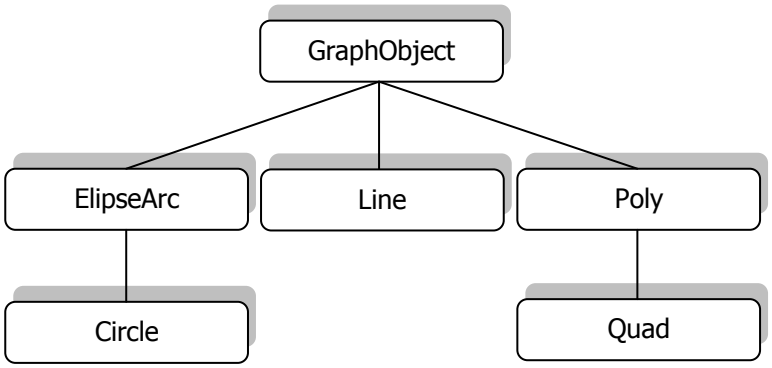
Lista osnovnih razreda navodi se tako da se iza naziva razreda stavi dvotočka ta se navedu nazivi osnovnih razreda razdvojeni zarezom. Ispred svakog od naziva moguće je

staviti jednu od ključnih riječi `public`, `private` ili `protected` čime se navodi tip nasljeđivanja.

Podatkovni i funkcijski članovi osnovnog razreda u slučaju javnog nasljeđivanja (kada se koristi ključna riječ `public` u listi nasljeđivanje) zadržavaju svoje pravo pristupa. To znači da podatkovni član `colour` ostaje privatan, pa nije javno dostupan niti u naslijeđenom razredu `Line`, dok članovi `SetColour()` i `GetColour()` imaju javni pristup te im se može pristupiti izvan razreda. U izvedenom razredu moguće je zamijeniti željene funkcijske članove osnovnog razreda novim članovima. Taj postupak se zove `zaobilaženje`.

Zadatak za vježbu:

1. Napišite C++ program koji će imati deklaracije razreda (osnovnog i izvedenih) na osnovu slijedećeg stabla:



5.1. Prava pristupa pri nasljeđivanju

5.1.1. Javni osnovni razred

Određeni razred je `javni osnovni razred` ako je u listi prilikom nasljeđivanja navedena ključna riječ `public`. Svi elementi osnovnog razreda bit će uključeni u izvedeni razred u kojem će zadržati svoje originalno pravo pristupa. Privatni članovi neće biti dostupni iz izvedenih razreda niti iz preostalog programa. Zaštićenim članovima će se moći pristupiti iz izvedenog razreda, ali ne iz glavnog programa. Javni članovi ostat će dostupni i iz glavnog programa i iz izvedenog razreda.

5.1.2. Privatni osnovni razred

Prilikom privatnog nasljeđivanja, privatni članoviosnovnog razreda nisu dostupni izvedenom razredu, dok javni i zaštićeni članovi osnovnog razreda postaju privatni članovi izvedenog razreda.

```
class Vector
{
private:
        float ax, ay;
public:
        Vector(float a = 0, float b = 0) : ax(a), ay(b) {}
        void Setxy(float x, float y) { ax = x; ay = y; }
        float Getx() { return ax; }
        float Gety() { return ay; }
        float MultiplyCross(Vector &v);
};
```

```
class Cplx : public Vector
{
public:
        Cplx(float a = 0, float b = 0) : Vector(a, b) {}
        void Multiply(Cplx &c);
};
```

Nedostatak ovakve hijerarhije razreda je u tome što je cjelokupno javno sučelje razreda `Vector` uključeno u javno sučelje razreda `Cplx`. To znači da će i funkcijski član `MultiplyCross` (vektorski produkt) biti dostupan u razredu `Cplx`. Time je omogućeno vektorsko množenje kompleksnih brojeva koje nema smisla i ne smije biti dozvoljeno.

Bolje rješenje prilikom izvođenja razreda `Cplx` koristi privatno nasljeđivanje. Time svi javni i zaštićeni članovi osnovnog razreda postaju privatni članovi izvedenog razreda. Javno sučelje osnovnog razreda ne uključuje se u javno sučelje izvedenog razreda.

```
class Cplx : private Vector
{
public:
        Cplx(float a = 0, float b = 0) : Vector(a, b) {}
        void Multiply(Cplx &c);
};
```

5.1.3. Zaštićeni osnovni razred

Zaštićeno nasljeđivanje se specificira tako da se prije naziva osnovnog razreda navede ključna riječ `protected`. Time se svi javni i zaštićeni članovi osnovnog razreda prenose kao zaštićeni u izvedeni razred. Cjelokupno javno sučelje osnovnog razreda se prenosi kao sučelje za nasljeđivanje u izvedeni razred. Svi izvedeni razredi mogu imati pristup članovima osnovnog razreda a ostatak programa tim članovima ne može pristupiti.

Pregled tipova nasljeđivanja i prava pristupa

Tip nasljeđivanja	Pravo pristupa u osnovnom razredu		
	public	protected	private
public	public	protected	private
protected	protected	protected	private
private	private	private	private

5.1.4. Izuzeće članova

Prilikom privatnog ili zaštićenog nasljeđivanja ponekad može biti prikladno ostaviti originalno pravo pristupa pojedinom članu osnovnog razreda. Na primjer, prilikom privatnog izvođenja razreda `Cplx` iz razreda `Vector`, javni članovi `Getx()` i `Gety()` automatski postaju skriveni. No kompleksni brojevi imaju svoj realni i imaginarni dio kojima bi se također moglo pristupiti pomoću tih članova. U jeziku C++ moguće je izuzeti pojedini član iz privatnog nasljeđivanja te mu ostaviti njegovo osnovno pravo pristupa:

```
class Vector
{
private:
        float ax, ay;
public:
        Vector(float a = 0, float b = 0) : ax(a), ay(b) {}
        void Setxy(float x, float y) { ax = x; ay = y; }
        float Getx() { return ax; }
        float Gety() { return ay; }
        float MultiplyCross(Vector &v);
};
```

```
class Cplx : private Vektor
{
public:
	Cplx(float a = 0, float b = 0) : Vector(a, b) {}
	void Multiply(Cplx &c);
	using Vector::Getx;
	using Vector::Gety;
};
```

Zadatak za vježbu:

1. Koristeći rješenje iz prethodnog zadatka, napišite deklaraciju razreda (osnovnog i izvedenih) koristeći različite vrste nasljeđivanja (javno, privatno, zaštićeno). Usporedite rezultate.