TULETATUD KLASSID

Olemasolevatest klassidest saab tuletada uusi klasse

Tuletatud klass (*derived class*) sisaldab endas kõiki baasklassi (*base class*) liikmeid – eellane pärandab järglastele kõik oma omadused (*inheritance*)

```
class  UUS  :  BAAS  {  …….  }   // privaatne tuletamine

class  UUS  :  public BAAS  {   …… }  // avalik tuletamine
```

Tuletatud klassi meetoditel on juurdepääs teatud baasklassi liikmetele:

Tuletamise viis	Baasklassi liige	Juurdepääsu viis tuletatud klassis
private	public	private
	private	---
	protected	private
public	public	public
	private	---
	protected	protected

Avalikul tuletamisel muutuvad baasklassi avalikud liikmed tuletatud klassi avalikeks liikmeteks

Näide:

```
# include  <conio.h>

class Aken                          // tekstiaken kuvari ekraanil
{
protected:
        int x1, x2;                 // horisontaalsed koordinaadid
        int y1, y2;                 // vertikaalsed koordinaadid
        int  taust, tekst;          // värvide koodid
        char *alumine;              // viit akna alla jäänud tekstile
        void varvid( void )
        {       textbackground( taust );  textcolor( tekst );
        };
        void snap (void )           // allajääva teksti salvestamine
        {       alumine = new char[(x2-x1+1)*(y2-y1+1)*2 + 1];
                gettext( x1, y1, x2, y2, alumine );
public:
        Aken( void )
        {       x1=y1=1; x2=80; y2=25; taust=BLACK;
                tekst=GREEN; suurus();  snap();
        }
        Aken( int a, int b, int c, int d )
        {       taust=BLACK; tekst=GREEN;
                suurus( a, b, c, d ); snap();
        }
        void suurus( void )
        {       window( x1, y1, x2, y2 );  varvid();
        }
        void suurus( int a, int b, int c, int d )
        {       x1=a; y1=b; x2=c; y2=d;  suurus();
        }
        void taasta( void )
        {       puttext( x1, y1, x2, y2, alumine);
        }
```

```
        void varv( int v, int w )
        {       taust=v; tekst=w; varvid();
        }
        void tyhi( void )
        {       suurus();  clrscr();
        }
}


class Menyy : Aken                  // rippmenüü
{
        int ridu;                   // valikute arv
        char **tekstid              // viit stringide massiivile
        int valikunr;               // valitud rea number
        Aken valik;                 // valitud rida on omaette aken!
        void print ( void );
        void print_valik( void );
public:
        Menyy( int, int, int, int, char**);
        void varv( int v, int w)
        {       Aken::varv( v, w);      // NB!
        }
        int vali( void );
        void snap( void )
        {       gettext( x1, y1, x2, y2, alumine );
        }
}
```

```
Menyy :: Menyy( int a, int b, int c, int d, char **p) : Aken(a,b,c,d)
{
	ridu=y2-y1+1; valikunr=1; tekstid=p;
}


void Menyy :: print( void )
{
	tyhi();
	for( int j=0; j<ridu; j++)
	{
		gotoxy( 1, j+1);
		cprintf( “%s”, *(tekstid+j) );		//  *tekstid[ j ]
	}
	print_valik();
}


void Menyy :: print_valik( void )
{
	int rida = y1 + valikunr – 1;
	valik.suurus( x1, rida, x2, rida );		// üherealine aken
	valik.varvid( tekst, taust );			// vastupidi!
	valik.tyhi();
	goto( 1, 1 )					// aknas valik!
	cprintf( “%s”, *(tekstid+valikunr-1) );
}
```

```
int Menyy :: vali( void )
{
	print();
	do {
		char c = getch();  if( !c ) c = getch();
		switch( c )
		{
		case  0x50:				// alla
			valikunr = (valikunr == ridu) ? 1 : valikunr + 1;
			print();
			break;
		case 0x48:				// üles
			valikunr = (valikunr == 1) ? ridu : valikunr - 1;
			print();
			break;
		case 0x1B:				// ESC
			taasta();
			return 0;
		case 0x0D:				// Enter
			return valikunr;
		default:
			cprintf( "\007" );		// Bell
		}
	} while( 1 );				// igavene tsükkel
}
```

```
int main( void )
{
	Aken ekraan;
	ekraan.varv( CYAN, BLACK );
	ekraan.tyhi();

	char *valikud[ ] = { “Tallinn”, “Tartu”, “Pärnu”, “Türi” };
	Menyy peamenyy( 20, 7, 32, 11, valikud );
	peamenyy.varv( BLUE, YELLOW );

	int v;
	do {
		v = peamenyy.vali();
		ekraan.suurus();
		cprintf( “Valiti %d  %s”, v, valikud[ v-1 ] );
		getch();
	 } while( v );
	getch();
	ekraan.taasta();
}
```

Virtuaalsed meetodid

Tuletatud klassis võib olla sama nimega meetodeid, mis baasklassiski.

Kui neil meetoditel on kas

- erinev arv parameetreid

või

- parameetrite tüübid erinevad

siis loetakse neid erinevateks meetoditeks.

Baasklassis kirjeldatud meetoditele saab tuletatud klassis anda uue tähenduse (*overridding*), kui baasklassis funktsiooni prototüübile eelneb võtmesõna *virtual.*

Näide 1.

```
class A
{  public:
        virtual int tegevus( void ) { ….. }
}

class B : public A
{  public:
        int tegevus( void )              // uus tähendus!
        {    ……
             int n = A :: tegevus();    // kui vaja
        }
}

class C : public A
{  public:
        int tegevus( int );
}

int main( void )
{
        B asi;
        int k = asi.tegevus();           // B :: tegevus()
        C tykk;
        k = tykk.tegevus();              // A :: tegevus()
        k = tykk.tegevus( 3 );           // C :: tegevus()
}
```

Näide 2.

Klass *List* realiseerib nimistusse suvalise andmeelemendi lisamise. Kuna elementide printimine sõltub elemendi tüübist, on vastav meetod kuulutatud virtuaalseks.

Klassist *List* on tuletud klassid *IntList* (täisarvude nimistu) ja *StrList* (stringide nimistu), milles printimine on üledefineeritud.

```
#include <iostream.h>
#define  NULL  0

struct Elem
{
        void  *data;                        // viit ükskõik millele
        Elem *next;
}

 class List
{
protected:
        Elem  *top;
public:
        List( void ) { top = NULL; }
        int  add( void * );
        virtual void print( void );
}

int List :: add( void *p )
{
        Elem uus = new Elem;
        if( !uus ) return 0;            // FALSE, kui pole mälu
        uus->data = p;
        uus->next = top;  top = uus;
        return 1;                       // TRUE, kui lisatud
}
```

```
void List :: print( void )
{       Elem *jooksev = top;
        while( jooksev )
        {
                cout << “Aadress = “ << jooksev->data << ‘\n’;
                jooksev = jooksev->next;
        }
}

class StrList : public List
{
 public:
        void print( void )
        {
                Elem *jooksev = top;
                while( jooksev )
                {
                        cout <<  (char *) jooksev->data << ‘\n’;
                        jooksev = jooksev->next;
                }
        }
}

class IntList : public List
{
 public:
        void print( void )
        {       Elem *jooksev = top;
                while( jooksev )
                {
                        cout <<  *(int *) jooksev->data << ‘\n’;
                        jooksev = jooksev->next;
                }
        }
}
```

```
int main( void )
{
        StrList  s;
        s.add( “Kollane” );  s.add( “Sinine” ); s.add( “Punane” );
        s.print();

        IntList  j;
        int k=17, m=-4, n=6;
        j.add( &k ); j.add( &m ); j.add( &n );
        j.print();

        List *w;
        w = &s;   w->print();           // prindib teist korda!
        w = &j;    w->print();

 // Ebameeldivus: stringide nimistusse saab toppida arve ja
 // vastupidi!

        s.add( &k );  j.add( “Must” );
        s.print();    j.print();                          // pudru!

// võib proovida aga nii:

        s.List::print();  j.List::print();
}
```