Objektum elvű alkalmazások fejlesztése

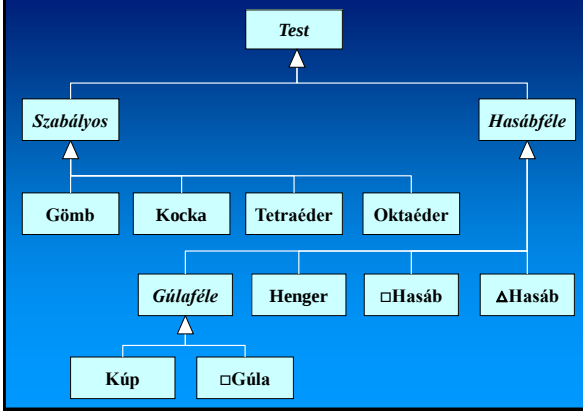
Öröklődés

Készítette: Sike Sándor
Gregorics Tibor

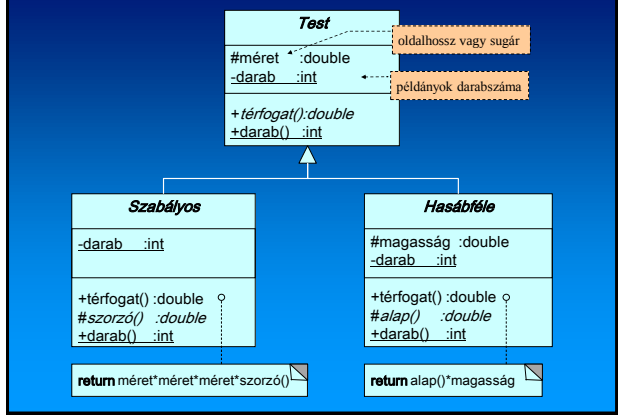
Feladat

Készítsünk programot, amellyel testek térfogatát számolhatjuk ki, illetve megadhatjuk azt is, hogy az egyes testfajtákból hány objektum létezik!
A lehetséges fajták:
- szabályos sokszögek: gömb, kocka, tetraéder, oktaéder;
- hasáb jellegű testek: henger, négyzet alapú és szabályos háromszög alapú hasáb;
- gúla jellegű testek: kúp, négyzetes gúla.

Osztály diagram



Testek



Absztrakt test

```
class Test
{
    public:
        virtual ~Test();
        virtual double terfogat() const = 0;
        static int darab() { return _darab; }
    protected:
        Test(double meret);
        double _meret;
    private:
        static int _darab;
};
```

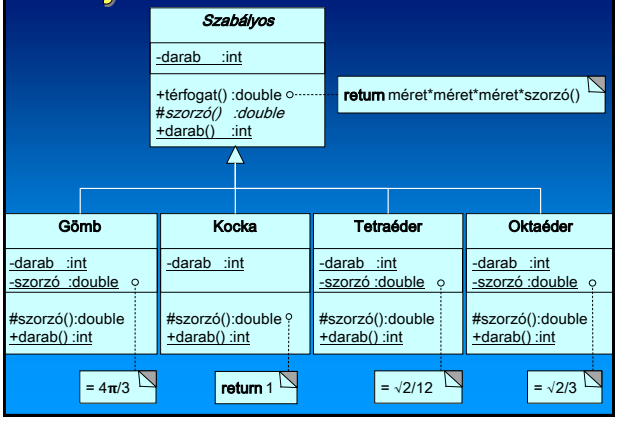
```
int Test::_darab = 0;

Test::Test(double meret){
    _meret = meret;
    ++_darab;
}
Test::~Test(){
    --_darab;
}
```

Szabályos absztrakt test

```
class Szabalyos : public Test{
    public:
        ~Szabalyos();
        double terfogat() const;
        static int darab() { return _darab; }
    protected:
        Szabalyos(double meret);
        virtual double szorzo() const = 0;
    private:
        static int _darab;
};
```

```
int Szabalyos::_darab = 0;

Szabalyos::Szabalyos(double meret) : Test(meret){
    ++_darab;
}
Szabalyos::~Szabalyos(){
    --_darab;
}
double Szabalyos::terfogat() const{
    return _meret * _meret * _meret * szorzo();
}
```

Szabályos testek



Gömb

```
class Gomb : public Szabalyos
{
   public:
      Gomb(double meret);
      ~Gomb();
      static int darab() { return _darab; }
   protected:
      double szorzo() const { return _szorzo; }
   private:
      const static double _szorzo = (4.0 * 3.14159) / 3.0;
      static int _darab;
};
```

```
int Gomb::_darab = 0;

Gomb::Gomb(double meret) : Szabalyos(meret){
    ++_darab;
}
Gomb::~Gomb(){
    --_darab;
}
```

Kocka

```
class Kocka : public Szabalyos
{
   public:
      Kocka(double meret);
      ~Kocka();
      static int darab() { return _darab; }
   protected:
      double szorzo() const { return 1.0; }
   private:
      static int _darab;
};
```

```
int Kocka::_darab = 0;

Kocka::Kocka(double meret) : Szabalyos(meret){
    ++_darab;
}
Kocka::~Kocka(){
    --_darab;
}
```

Tetraéder

```
class Tetraeder : public Szabalyos
{
   public:
      Tetraeder(double meret);
      ~Tetraeder();
      static int darab() { return _darab; }
   protected:
      double szorzo() const { return _szorzo; }
   private:
      const static double _szorzo = 1.41421 / 12.0;
      static int _darab;
};
```

```
int Tetraeder::_darab = 0;

Tetraeder::Tetraeder(double meret) : Szabalyos(meret){
    ++_darab;
}
Tetraeder::~Tetraeder(){
    --_darab;
}
```

Oktaéder

```
class Oktaeder : public Szabalyos
{
   public:
      Oktaeder(double meret);
      ~Oktaeder();
      static int darab() { return _darab; }
   protected:
      double szorzo() const { return _szorzo; }
   private:
      const static double _szorzo = 1.41421 / 3.0;
      static int _darab;
};
```

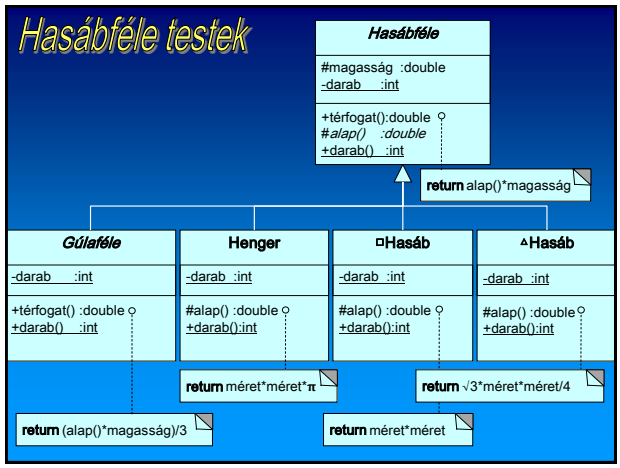
```
int Oktaeder::_darab = 0;

Oktaeder::Oktaeder(double meret) : Szabalyos(meret){
    ++_darab;
}
Oktaeder::~Oktaeder(){
    --_darab;
}
```

Hasábféle absztrakt test

```
class Hasabfele : public Test{
   public:
      ~Hasabfele ();
      double terfogat() const;
      static int darab() { return _darab; }
   protected:
      Hasabfele(double meret, double magassag);
      virtual double alap() const = 0;
      double _magassag;
   private:
      static int _darab;
};
```

```
int Hasabfele::_darab = 0;

Hasabfele::Hasabfele(double meret, double magassag): Test(meret){
    _magassag = magassag; ++_darab;
}
Hasabfele::~Hasabfele (){
    --_darab;
}
double Hasabfele::terfogat() const{
    return alap() * _magassag;
}
```

Hasábféle testek



Henger

```
class Henger : public Hasabfele
{
   public:
      Henger(double meret, double magassag);
      ~Henger();
      static int darab() { return _darab; }
   protected:
      double alap() const;
   private:
      static int _darab;
};
```

```
int Henger::_darab = 0;

Henger::Henger(double meret, double magassag)
: Hasabfele(meret, magassag){
    ++_darab;
}
Henger::~Henger(){
    --_darab;
}
double Henger::alap() const{
    return 3.14159 * _meret * _meret;
}
```

Hasáb

```
class NegyzetesHasab : public Hasabfele
{
   public:
      NegyzetesHasab (double meret, double magassag);
      ~NegyzetesHasab ();
      static int darab() { return _darab; }
   protected:
      double alap() const;
   private:
      static int _darab;
};
```

```
int NegyzetesHasab::_darab = 0;

NegyzetesHasab::NegyzetesHasab (double meret, double magassag)
: Hasabfele(meret, magassag){
    ++_darab;
}
NegyzetesHasab::~NegyzetesHasab (){
    --_darab;
}
double NegyzetesHasab::alap() const{
    return _meret * _meret;
}
```

Hasáb

```
class HaromszogesHasab : public Hasabfele
{
   public:
      HaromszogesHasab (double meret, double magassag);
      ~HaromszogesHasab ();
      static int darab() { return _darab; }
   protected:
      double alap() const;
   private:
      static int _darab;
};
```

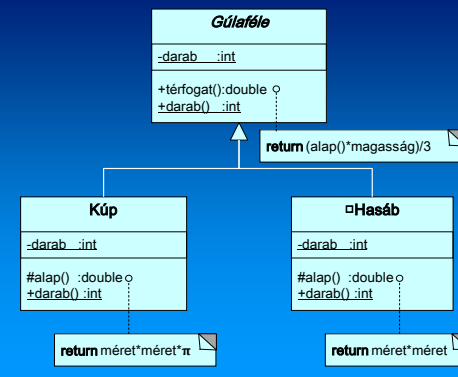
```
int HaromszogesHasab::_darab = 0;

HaromszogesHasab::HaromszogesHasab (double meret, double magassag)
: Hasabfele(meret, magassag){
    ++_darab;
}
HaromszogesHasab::~HaromszogesHasab (){
    --darab;
}
double HaromszogesHasab::alap() const{
    return 1.73205 * _meret * _meret / 4.0;
}
```

Gúlaféle absztrakt testek

```
class Gulafele : public Hasabfele
{
   public:
      ~Gulafele();
      double terfogat() const;
      static int darab() { return _darab; }
   protected:
      Gulafele(double meret, double magassag);
   private:
      static int _darab;
};
```

```
int Gulafele::_darab = 0;

Gulafele::Gulafele(double meret, double magassag)
: Hasabfele(meret, magassag){
    ++_darab;
}
Gulafele::~Gulafele(){
    --_darab;
}
double Gulafele::terfogat() const{
    return (alap() * _magassag) / 3.0;
}
```

Gúlaféle testek



Kúp

```
class Kup : public Gulafele
{
   public:
      Kup(double meret, double magassag);
      ~Kup();
      static int darab() { return _darab; }
   protected:
      double alap() const;
   private:
      static int _darab;
};
```

```
int Kup::_darab = 0;

Kup::Kup(double meret, double magassag)
: Gulafele(meret, magassag){
   ++_darab;
}
Kup::~Kup(){
   --_darab;
}
double Kup::alap() const{
   return 3.14159 * _meret * _meret;
}
```

Gúla

```
class NegyzetesGula : public Gulafele
{
   public:
      NegyzetesGula(double meret, double magassag);
      ~ NegyzetesGula();
      static int darab() { return _darab; }
   protected:
      double alap() const;
   private:
      static int _darab;
};
```

```
int NegyzetesGula::_darab = 0;

NegyzetesGula::NegyzetesGula(double meret, double magassag)
: Gulafele(meret, magassag){
   ++_darab;
}
NegyzetesGula::~NegyzetesGula(){
   --_darab;
}
double NegyzetesGula::alap() const{
   return _meret * _meret;
}
```

Főprogram - beolvasás

```
#include <cstdlib>
#include <iostream>
#include <fstream>
#include "test.h"

using namespace std;

int main()
{
   ifstream inp("testek.txt");

   int testszam;
   inp >> testszam;
   Test **testek = new Test *[testszam];

   for ( int i = 0; i < testszam; ++i ){
      string tipus;
      double meret, magassag;
      inp >> tipus;
      …   // különféle testek létrehozása a fájlbeli adatok alapján
   }
   inp.close();
   …
```

testek.txt

```
8
Kocka 5.0
Henger 3.0 8.0
Henger 1.0 10.0
Tetraeder 4.0
NegyzetesGula 3.0 10.0
Oktaeder 1.0
Kocka 2.0
NegyzetesGula 2.0 10.0
```

Főprogram – testek létrehozása

```
      inp >> tipus;
      if ( tipus == "Kocka" ){
         inp >> meret;
         testek[i] = new Kocka(meret);
      }
      else if ( tipus == "Gomb" ){
         inp >> meret;
         testek[i] = new Gomb(meret);
      }
      else if ( tipus == "Tetraeder" ){
         inp >> meret;
         testek[i] = new Tetraeder(meret);
      }
      else if ( tipus == "Oktaeder" ){
         inp >> meret;
         testek[i] = new Oktaeder(meret);
      }
      else if ( tipus == "Henger" ){
         inp >> meret;
         inp >> magassag;
         testek[i] = new Henger(meret, magassag);
      }
      …
```

A származtatás miatt lehet értékül adni egy Test* típusú változónak egy Kocka* pointert

Főprogram – testek létrehozása

```
      …
      else if ( tipus == "NegyzetesHasab" ){
         inp >> meret;
         inp >> magassag;
         testek[i] = new NegyzetesHasab(meret, magassag);
      }
      else if ( tipus == "HaromszogesHasab" ){
         inp >> meret;
         inp >> magassag;
         testek[i] = new HaromszogesHasab(meret, magassag);
      }
      else if ( tipus == "Kup" ){
         inp >> meret;
         inp >> magassag;
         testek[i] = new Kup(meret, magassag);
      }
      else if ( tipus == "NegyzetesGula" ){
         inp >> meret;
         inp >> magassag;
         testek[i] = new NegyzetesGula(meret, magassag);
      }
      else{
         cout << "Ismeretlen idom" << endl;
      }
```

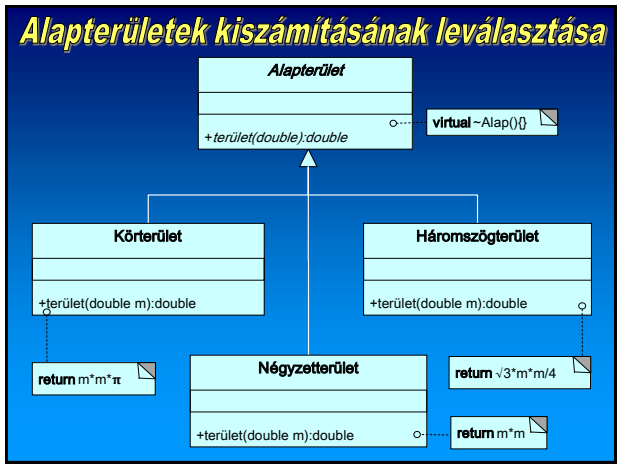
Főprogram

```
   …
   for ( int i = 0; i < testszam; ++i ){
      cout << testek[i]->terfogat() << endl;
   }

   cout << Test::darab()       << " " << Szabalyos::darab()       << " "
        << Hasabfele::darab() << " " << Gulafele::darab()       << " "
        << Gomb::darab()       << " " << Kocka::darab()          << " "
        << Tetraeder::darab() << " " << Oktaeder::darab()       << " "
        << Henger::darab()     << " " << NegyzetesHasab::darab() << " "
        << HaromszogesHasab::darab() << " "
        << Kup::darab()        << " " << NegyzetesGula::darab()  << endl;

   for ( int i = 0; i < testszam; ++i ) delete testek[i];
   delete [] testek;

   cout << Test::darab()       << " " << Szabalyos::darab()       << " "
        << Hasabfele::darab() << " " << Gulafele::darab()       << " "
        << Gomb::darab()       << " " << Kocka::darab()          << " "
        << Tetraeder::darab() << " " << Oktaeder::darab()       << " "
        << Henger::darab()     << " " << NegyzetesHasab::darab() << " "
        << HaromszogesHasab::darab() << " "
        << Kup::darab()        << " " << NegyzetesGula::darab()  << endl;
   return 0;
}
```

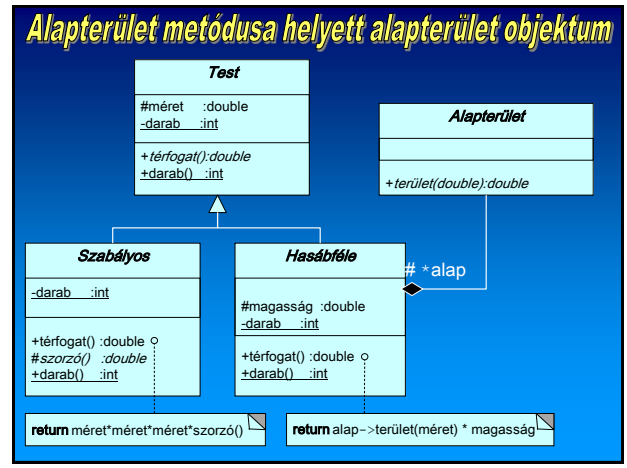
polimorfizmus: többalakúság

dinamikus kötés: futás közben dől el, hogy ez milyen típusú, és ettől függ, hogy melyik terfogat() metódus hívódjon meg.

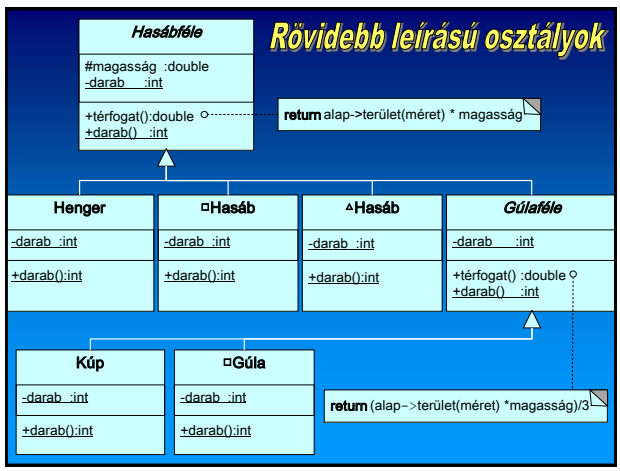
Alapterületek kiszámításának leválasztása



Alapterület metódusa helyett alapterület objektum



Rövidebb leírású osztályok




```
NegyzetesHasab::NegyzetesHasab(…) : Hasabfele(…)
{
    ++_darab; _alap = new Negyzet();
}
NegyzetesHasab::~NegyzetesHasab()
{
    --_darab; delete _alap;
}
```

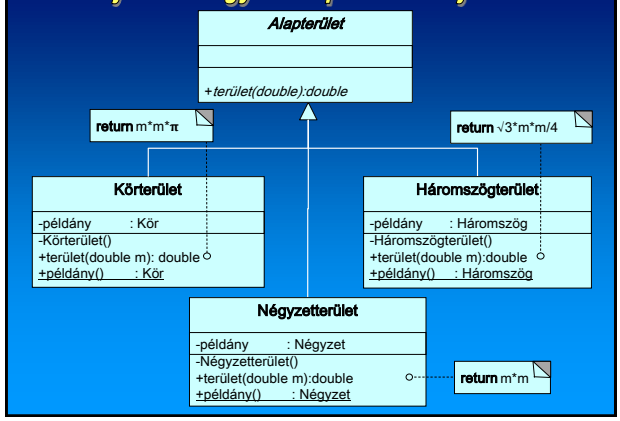
```
NegyzetesGula::NegyzetesGula(…) : Gulafele(…)
{
    ++_darab; _alap = new Negyzet();
}
NegyzetesGula::~NegyzetesGula()
{
    --_darab; delete _alap;
}
```

```
Henger::Henger(…) : Hasabfele(…)
{
    ++_darab; _alap = new Kor();
}
Henger::~Henger()
{
    --_darab; delete _alap;
}
```

```
HaromszogesHasab::HaromszogesHasab(…) : Hasabfele(…)
{
    ++_darab; _alap = new Haromszog();
}
HaromszogesHasab::~HaromszogesHasab()
{
    --_darab; delete _alap;
}
```

```
Kup::Kup(…) : Gulafele(…)
{
    ++_darab; _alap = new Kor();
}
Kup::~Kup()
{
    --_darab; delete _alap;
}
```

További javítás: egyke alapterület-objektumok



Négyzet

```
class Negyzetterulet : public Alapterulet
{
   public:
      double terulet(double m) const {
         return m * m;
      }
      static Negyzetterulet *peldany();
   private:
      static Negyzetterulet *_peldany;
      Negyzetterulet () {}
};
```

```
Negyzetterulet *Negyzetterulet::_peldany = 0;

Negyzetterulet *Negyzetterulet::peldany()
{
   if ( _peldany == 0 ) _peldany = new Negyzetterulet();
   return _peldany;
}
```

Kör

```
class Korterulet : public Alapterulet
{
   public:
       double terulet(double m) const {
           return 3.14159 * m * m;
       }
       static Korterulet *peldany();
   private:
       static Korterulet *_peldany;
       Korterulet () {}
};
```

```
Korterulet *Korterulet::_peldany = 0;

Korterulet *Korterulet::peldany()
{
    if ( _peldany == 0 ) _peldany = new Korterulet();
    return _peldany;
}
```

Háromszög

```
class Haromszogterulet : public Alapterulet
{
   public:
       double terulet(double m) const {
           return 1.73205 * m * m / 4.0;
       }
       static Haromszogterulet *peldany();
   private:
       static Haromszogterulet *_peldany;
       Haromszogterulet() {}
};
```

```
Haromszogterulet *Haromszogterulet::_peldany = 0;

Haromszogterulet *Haromszogterulet::peldany()
{
    if ( _peldany == 0 ) _peldany = new Haromszogterulet();
    return _peldany;
}
```

```
NegyzetesHasab::NegyzetesHasab(…) : Hasabfele(…){
    ++_darab; _alap = Negyzetterulet::peldany();
}
NegyzetesHasab::~NegyzetesHasab(){
    --_darab;
}
```

```
Henger::Henger(…) : Hasabfele(…){
    ++_darab;
    _alap = Korterulet::peldany();
}
Henger::~Henger(){
    --_darab;
}
```

```
NegyzetesGula::NegyzetesGula(…) : Gulafele(…){
    ++_darab; _alap = Negyzetterulet::peldany();
}
NegyzetesGula::~NegyzetesGula(){
    --_darab;
}
```

```
Kup::Kup(…) : Gulafele(…){
    ++_darab;
    _alap = Korterulet::peldany();
}
Kup::~Kup(){
    --_darab;
}
```

```
Haromszoges::HaromszogesHasab(…) : Hasabfele(…){
    ++_darab; _alap = Haromszogterulet::peldany();
}
HaromszogesHasab::~HaromszogesHasab(){
    --_darab;
}
```