Elemi Alkalmazások Fejlesztése II.

Öröklődés

Az osztályokkal és az öröklődéssel kapcsolatos ismereteket egészítjük ki a következő feladat megoldása kapcsán.

1. Feladat

Készítsünk programot, amellyel testek térfogatát határozhatjuk meg, 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 hasáb, szabályos háromszög alapú hasáb;
- gúla jellegű testek: kúp, négyzetes gúla.

2. A megoldás menete

A feladatot az öröklődés felhasználásával oldjuk meg. Az absztrakt `Test` osztály, adja meg a testek közös jellemzőit.

Ebből származtatjuk az absztrakt `Szabalyos` osztályt, amely a szabályos testek tulajdonságait tartalmazza. Ennek konkrét esetei lesznek a `Gomb`, `Kocka`, `Tetraeder`, `Oktaeder` osztályok.

A `Hasab` absztrakt osztály szintén a `Test` osztály leszármazottja, és az alapterület és magasság szorzatával kiszámítható térfogatú testeket írja le. Ennek speciális esete a `Gula`, amikor a szorzatot $\frac{1}{3}$-dal kell megszorozni.

A `Hasab` osztály konkrét esetei: `Henger`, `Negyzetes` (négyzet alapú hasáb), `Haromszoges` (szabályos háromszög alapú hasáb).

A `Gula` osztály konkrét esetei: `Kup` és `NGula` (négyzet alapú gúla).

3. Közös tulajdonságok

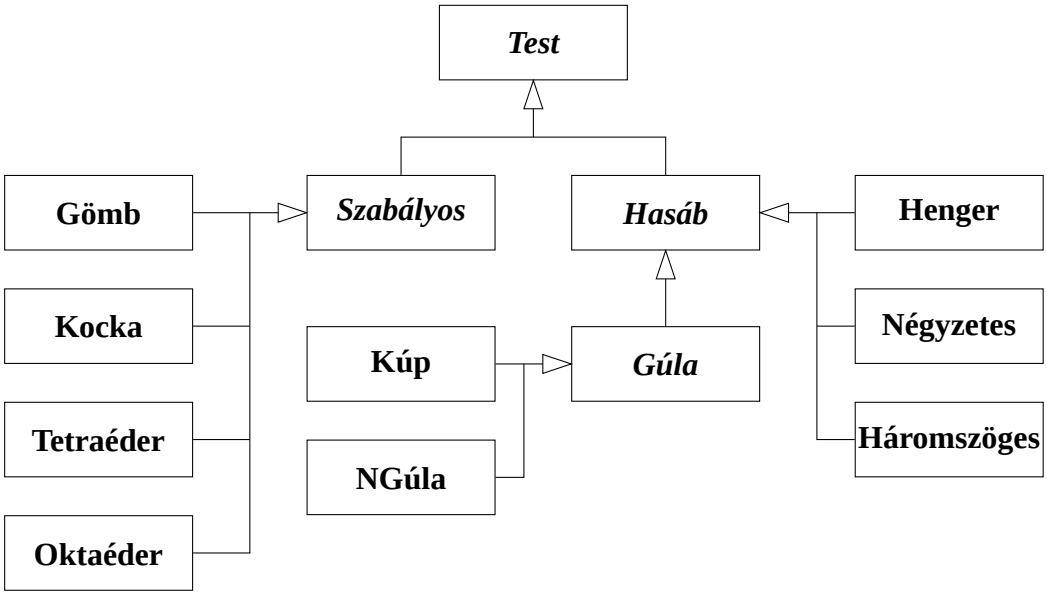
Minden test jellemzője egy *méret*, ami szabályos sokszögek esetén a sugarat, illetve az oldal hosszát jelenti. Egyéb esetben (hasábok) az alapot meghatározó szabályos síkidom meghatározó adata, amit ekkor a *magassággal* kell kiegészítenünk.

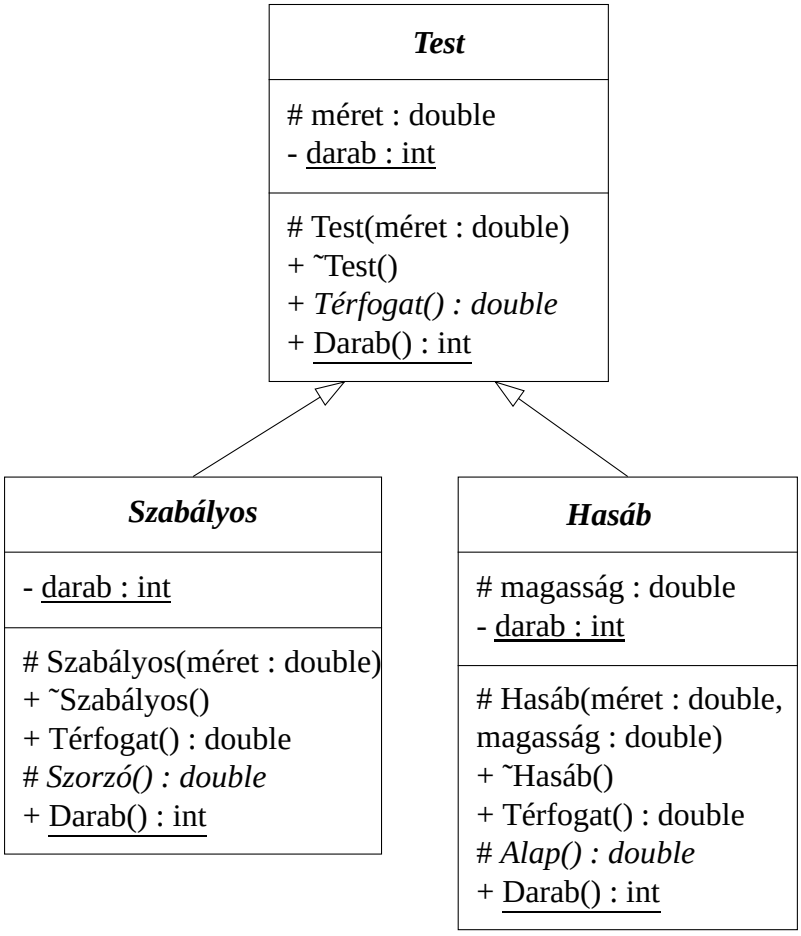
Ezen kívül minden osztályban nyilván kell tartanunk az osztály objektumainak a számát. Ez egy osztályszintű adattag lesz.

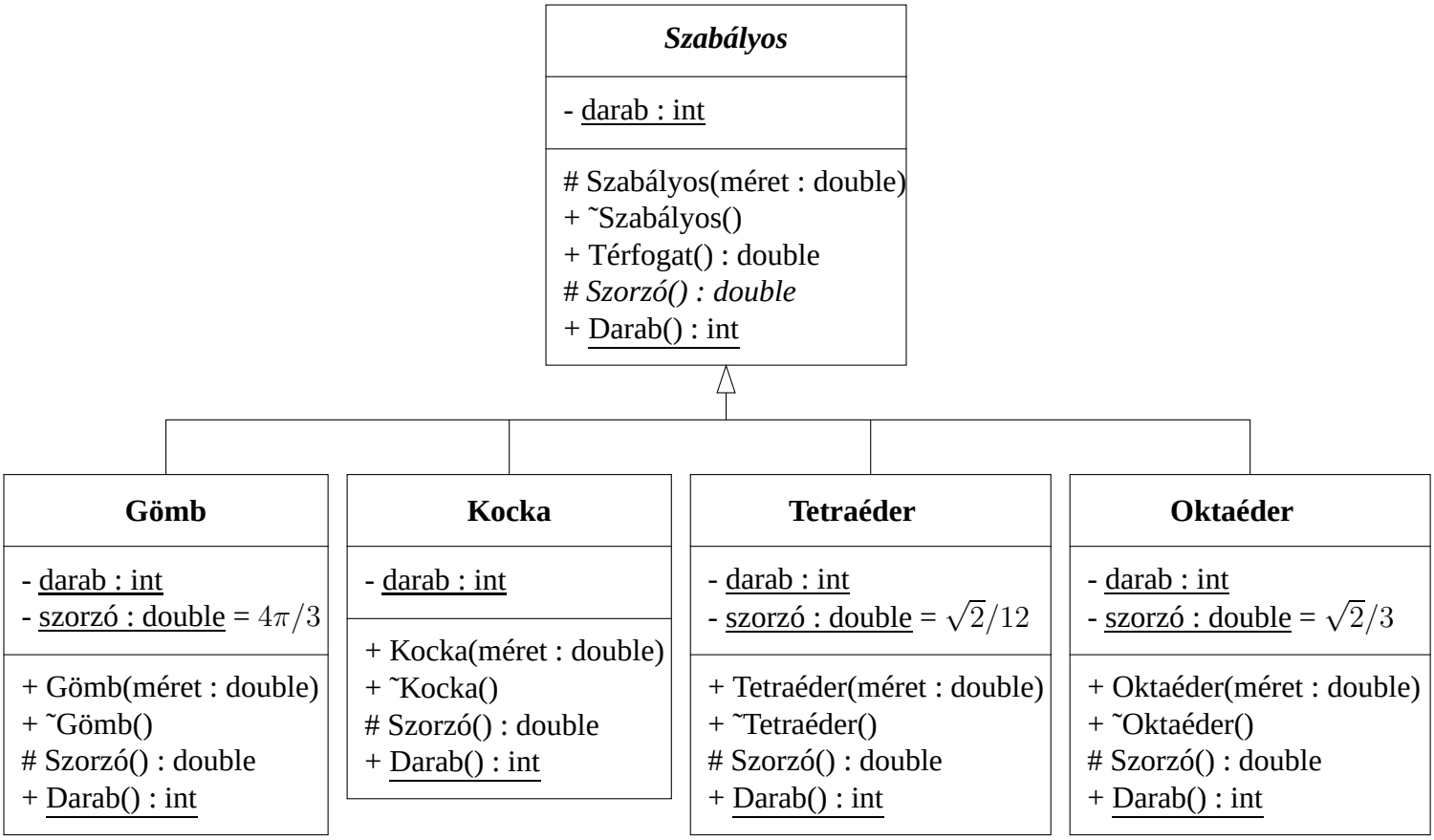
A közös művelet a térfogat kiszámítására szolgáló függvény (`Terfogat`), illetve az objektumok számát lekérdező osztályszintű `Darab` függvény.

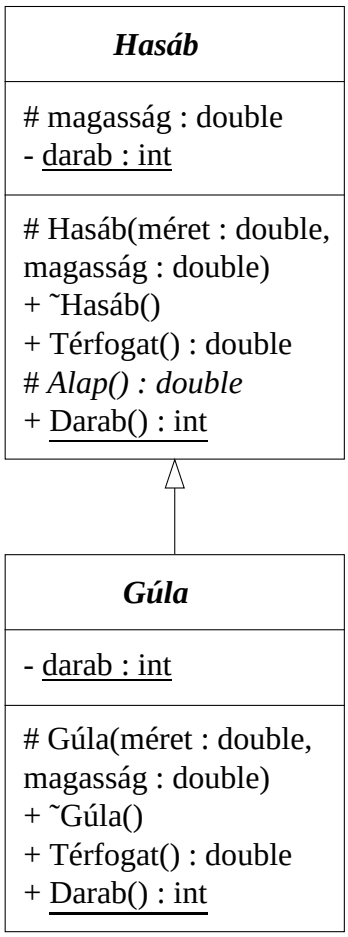
Az objektumok számát az egyes konstruktorokban kell növelni, destruktorokban csökkenteni. Egy származtatott osztályban explicit meg kell hívnunk az ősosztály paraméteres konstruktorát. Ez egyben garantálja, hogy az ősosztály „objektumainak” száma is nő. A származtatott osztály destruktora automatikusan meghívja az ős osztály destruktorát is, erről nem kell külön rendelkeznünk.

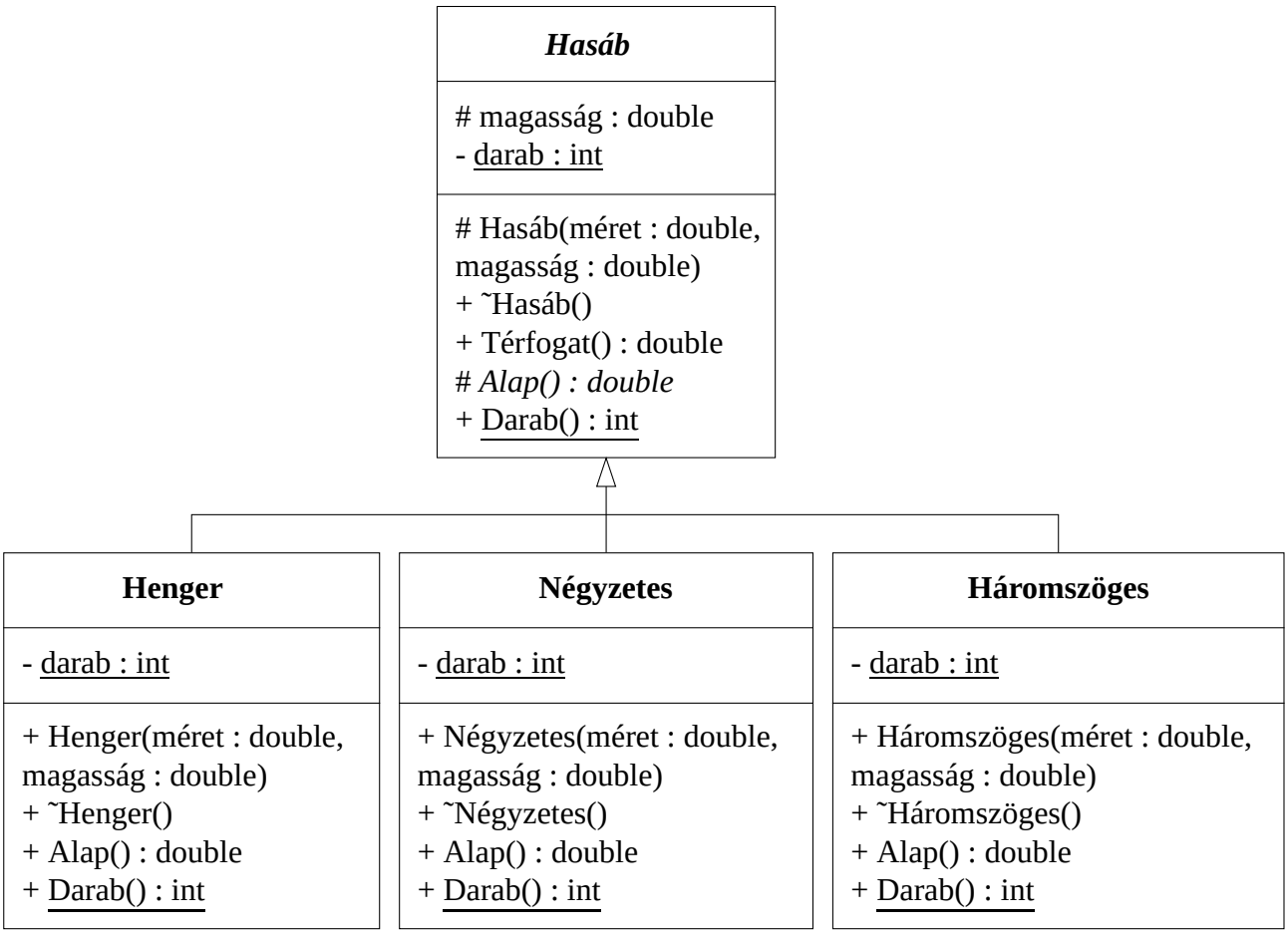
4. Osztálydiagram

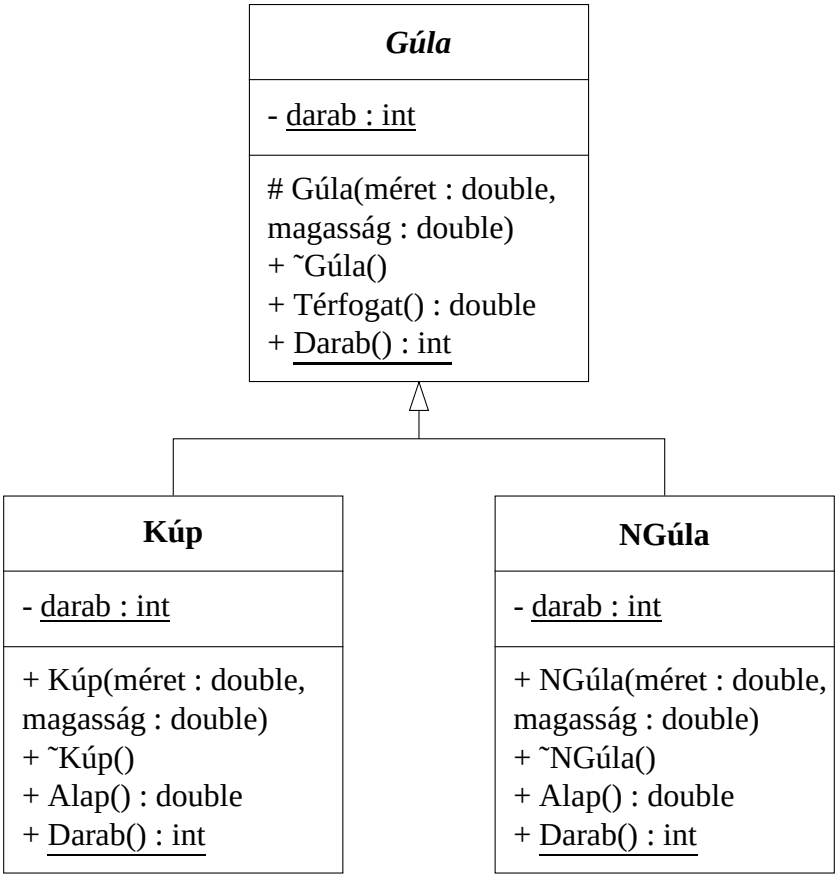



Test
méret : double
- darab : int
Test(méret : double)
+ ˜Test()
+ Térfogat() : double
+ Darab() : int
Szabályos
- darab : int
Szabályos(méret : double)
+ ˜Szabályos()
+ Térfogat() : double
# Szorzó() : double
+ Darab() : int
Hasáb
magasság : double
- darab : int
Hasáb(méret : double,
magasság : double)
+ ˜Hasáb()
+ Térfogat() : double
# Alap() : double
+ Darab() : int


Szabályos
- darab : int
Szabályos(méret : double)
+ ~Szabályos()
+ Térfogat() : double
# Szorzó() : double
+ Darab() : int
Gömb
- darab : int
- szorzó : double = 4π/3
+ Gömb(méret : double)
+ ~Gömb()
# Szorzó() : double
+ Darab() : int
Kocka
- darab : int
+ Kocka(méret : double)
+ ~Kocka()
# Szorzó() : double
+ Darab() : int
Tetraéder
- darab : int
- szorzó : double = √2/12
+ Tetraéder(méret : double)
+ ~Tetraéder()
# Szorzó() : double
+ Darab() : int
Oktaéder
- darab : int
- szorzó : double = √2/3
+ Oktaéder(méret : double)
+ ~Oktaéder()
# Szorzó() : double
+ Darab() : int


Hasáb
magasság : double
- darab : int
Hasáb(méret : double,
magasság : double)
+ ˜Hasáb()
+ Térfogat() : double
# Alap() : double
+ Darab() : int
Gúla
- darab : int
Gúla(méret : double,
magasság : double)
+ ˜Gúla()
+ Térfogat() : double
+ Darab() : int


Hasáb
magasság : double
- darab : int
Hasáb(méret : double, magasság : double)
+ ˜Hasáb()
+ Térfogat() : double
# Alap() : double
+ Darab() : int
Henger
- darab : int
+ Henger(méret : double, magasság : double)
+ ˜Henger()
+ Alap() : double
+ Darab() : int
Négyzetes
- darab : int
+ Négyzetes(méret : double, magasság : double)
+ ˜Négyzetes()
+ Alap() : double
+ Darab() : int
Háromszöges
- darab : int
+ Háromszöges(méret : double, magasság : double)
+ ˜Háromszöges()
+ Alap() : double
+ Darab() : int


Gúla
- darab : int
Gúla(méret : double, magasság : double)
+ ˜Gúla()
+ Térfogat() : double
+ Darab() : int
Kúp
- darab : int
+ Kúp(méret : double, magasság : double)
+ ˜Kúp()
+ Alap() : double
+ Darab() : int
NGúla
- darab : int
+ NGúla(méret : double, magasság : double)
+ ˜NGúla()
+ Alap() : double
+ Darab() : int

5. A `Test` osztály

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

```
int Test::darab = 0;

Test::Test(double meret)
{
    this->meret = meret;   darab++;
}

Test::~Test()
{
    darab--;
}
```

6. A `Szabalyos` osztály

```
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--;
}
```

7. A `Hasab` osztály

```
class Hasab : public Test
{
public:
    ~Hasab();
    double Terfogat() const;
    static int Darab()  { return darab; }
protected:
    Hasab(double meret, double magassag);
    virtual double Alap() const = 0;
protected:
    double   magassag;
private:
    static int darab;
};
```

```
int Hasab::darab = 0;

Hasab::Hasab(double meret, double magassag) : Test(meret)
{
    this->magassag = magassag;   darab++;
}

Hasab::~Hasab()
{
   darab--;
}

double Hasab::Terfogat() const
{
    return Alap() * magassag;
}
```

8. A `Gula` osztály

```
class Gula : public Hasab
{
public:
    ~Gula();
    double Terfogat() const;
    static int Darab()  { return darab; }
protected:
    Gula(double meret, double magassag);
private:
    static int darab;
};
```

```
int Gula::darab = 0;

Gula::Gula(double meret, double magassag) :
            Hasab(meret, magassag)
{
    darab++;
}

Gula::~Gula()
{
    darab--;
}

double Gula::Terfogat() const
{
    return (Alap() * magassag) / 3.0;
}
```

9. A `Gomb` osztály

```
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 * 3.141592) / 3.0;
    static int darab;
};
```

```
int Gomb::darab = 0;

Gomb::Gomb(double meret) : Szabalyos(meret)
{
    darab++;
}

Gomb::~Gomb()
{
    darab--;
}
```

10. A Kocka osztály

```
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--;
}
```

11. A `Tetraeder` osztály

```
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 = 1.4142135 / 12.0;
    static int darab;
};
```

```
int Tetraeder::darab = 0;

Tetraeder::Tetraeder(double meret) : Szabalyos(meret)
{
    darab++;
}

Tetraeder::~Tetraeder()
{
    darab--;
}
```

12. A `Oktaeder` osztály

```
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 = 1.4142135 / 3.0;
    static int darab;
};
```

```
int Oktaeder::darab = 0;

Oktaeder::Oktaeder(double meret) : Szabalyos(meret)
{
    darab++;
}

Oktaeder::~Oktaeder()
{
    darab--;
}
```

13. A `Henger` osztály

```
class Henger : public Hasab
{
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) :
                  Hasab(meret, magassag)
{
    darab++;
}

Henger::~Henger()
{
    darab--;
}

double Henger::Alap() const
{
    return 3.141592 * meret * meret;
}
```

14. A `Negyzetes` osztály

```
class Negyzetes : public Hasab
{
public:
    Negyzetes(double meret, double magassag);
    ~Negyzetes();
    static int Darab()  { return darab; }
protected:
    double Alap() const;
private:
    static int darab;
};
```

```
int Negyzetes::darab = 0;

Negyzetes::Negyzetes(double meret, double magassag) :
                      Hasab(meret, magassag)
{
    darab++;
}

Negyzetes::~Negyzetes()
{
    darab--;
}

double Negyzetes::Alap() const
{
    return meret * meret;
}
```

15. A `Haromszoges` osztály

```
class Haromszoges : public Hasab
{
public:
    Haromszoges(double meret, double magassag);
    ~Haromszoges();
    static int Darab()  { return darab; }
protected:
    double Alap() const;
private:
    static int darab;
};
```

```
int Haromszoges::darab = 0;

Haromszoges::Haromszoges(double meret, double magassag) :
                          Hasab(meret, magassag)
{
    darab++;
}

Haromszoges::~Haromszoges()
{
    darab--;
}

double Haromszoges::Alap() const
{
    return  1.73205 * meret * meret / 4.0;
}
```

16. A Kup osztály

```
class Kup : public Gula
{
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) : Gula(meret, magassag)
{
    darab++;
}

Kup::~Kup()
{
    darab--;
}

double Kup::Alap() const
{
    return  3.141592 * meret * meret;
}
```

17. A `NGula` osztály

```
class NGula : public Gula
{
public:
    NGula(double meret, double magassag);
    ~NGula();
    static int Darab()  { return darab; }
protected:
    double Alap() const;
private:
    static int darab;
};
```

```
int NGula::darab = 0;

NGula::NGula(double meret, double magassag) :
             Gula(meret, magassag)
{
    darab++;
}

NGula::~NGula()
{
    darab--;
}

double NGula::Alap() const
{
    return  meret * meret;
}
```

18. Főprogram

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

using namespace std;

int main(int argc, char *argv[])
{
    ifstream inp("testek.txt");
    int     testszam;
    Test    **testek;
    string  tipus;
    double  meret;
    double  magassag;

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

    for ( int i = 0; i < testszam; i++ )
    {
        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);
}
```

```
else if ( tipus == "Negyzetes" )
{
    inp >> meret;
    inp >> magassag;
    testek[i] = new Negyzetes(meret, magassag);
}
else if ( tipus == "Haromszoges" )
{
    inp >> meret;
    inp >> magassag;
    testek[i] = new Haromszoges(meret, magassag);
}
else if ( tipus == "Kup" )
{
    inp >> meret;
    inp >> magassag;
    testek[i] = new Kup(meret, magassag);
}
else if ( tipus == "NGula" )
{
    inp >> meret;
    inp >> magassag;
    testek[i] = new NGula(meret, magassag);
}
else
```

```
    {
        cout << "Ismeretlen idom" << endl;
    }
}
inp.close();

for ( int i = 0; i < testszam; i++ )
    cout << testek[i]->Terfogat() << endl;

cout << Test::Darab() << " " << Szabalyos::Darab() << " "
    << Hasab::Darab() << " " << Gula::Darab() << " " <<
    Gomb::Darab() << " " << Kocka::Darab() << " " <<
    Tetraeder::Darab() << " " << Oktaeder::Darab() << " "
    << Henger::Darab() << " " << Negyzetes::Darab()
    << " " << Haromszoges::Darab() << " " << Kup::Darab()
    << " " << NGula::Darab() << endl;

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

cout << Test::Darab() << " " << Szabalyos::Darab() << " "
    << Hasab::Darab() << " " << Gula::Darab() << " " <<
    Gomb::Darab() << " " << Kocka::Darab() << " " <<
    Tetraeder::Darab() << " " << Oktaeder::Darab() << " "
```

```
        << Henger::Darab() << " " << Negyzetes::Darab()
        << " " << Haromszoges::Darab() << " " << Kup::Darab()
        << " " << NGula::Darab() << endl;

    system("PAUSE");
    return EXIT_SUCCESS;
}
```