Szabó Mária

Programozás – Objektum-orientált programozás a gyakorlatban

NSZFI
NEMZETI SZAKKÉPZÉSI ÉS FELNŐTTKÉPZÉSI INTÉZET

A követelménymodul megnevezése:
Informatikai ismeretek

A követelménymodul száma: 1155-06 A tartalomelem azonosító száma és célcsoportja: SzT-019-50

Új Magyarország FEJLESZTÉSI TERV

PROGRAMOZÁS – OBJEKTUM ORIENTÁLT PROGRAMOZÁS A GYAKORLATBAN

ESETFELVETÉS – MUNKAHELYZET

Egy feltörekvő ékszerbolt korszerűsíti az eladás folyamatát. Az a feladata, hogy objektumorientált módon tervezzen meg egy programot, ami segíti a bolt tevékenységét! Milyen lépésekben tervezné meg a programot?

SZAKMAI INFORMÁCIÓTARTALOM

OBJEKTUM ORIENTÁLT PROGRAMOZÁS ALAPFOGALMAI

1. Fogalma:

Olyan programozási technika, amely a programokat objektumokból építi fel. A program működése tulajdonképpen objektumok (*minden objektumnak megvan a jól meghatározott feladata*) kommunikációját jelenti. Legfontosabb alapelvei: egységbezárás, öröklődés, polimorfizmus.

A strukturált programozást felváltja az objektum orientált programozás, hiszen a strukturált módszer már nem képes a megváltozott igényeknek megfelelő, szoftver megalkotására.

Az elkészített szoftverrel szemben támasztott követelmények:

- helyesség
- karbantarthatóság és bővíthetőség
- újrafelhasználhatóság
- felhasználó barátság
- hordozhatóság
- hatékonyságellenőrizhetőség
- kompatibilitás
- hibatűrés
- integritás (sérthetetlenség)
- szabványosság

MUNKAANYAG

2. Objektumorientált nyelvek osztályozása

- tiszta objektumorientált nyelvek
- objektum alapú nyelvek
- hibrid nyelvek

3. Az objektumorientált program jellemzői, alapfogalmak

Osztály (Class): Az osztály egy felhasználói típus, amelynek alapján példányok (objektumok) hozhatók létre. Az osztály alapvetően adat és metódus (művelet) definíciókat tartalmaz.

Objektum (példány): Információt (adatokat) tárol és kérésre műveleteket végez. Van állapota, viselkedése és futásidőben azonosítható.

Felelősség: Minden objektumnak megvan a jól meghatározott feladata, amelynek elvégzéséért felelős.

Osztályozás: Az objektum osztályokat viselkedésük szerint osztályokba soroljuk.

Üzenet, kérelem: Ezen keresztül kérjük meg az objektumokat különböző feladatok lvégzésére. Tulajdonképpen objektumhoz továbbított kérés. Válaszként az objektum végrehajtja a kért műveletet.

Bezárás, információ elrejtése: A feladatok elvégzésének módja az objektum belügye. Az objektum belseje sérthetetlen. Az objektummal csak az interfészen keresztül lehet kommunikálni. Tulajdonképpen bezárás alatt az adatok és metódusok egybezárását értjük. Az információ elrejtése azt jelenti, hogy az adatok és metódusok közül csak az érhető el kívülről, amelynek külső elérését az interfészen keresztül engedélyezzük.

Polimorfizmus (többalakúság): Ugyanarra az üzenetre különböző objektumok különbözőképpen reagálhatnak. Egy típuselméleti fogalom, amely szerint egy ősosztály típusú változó hivatkozhat ugyanazon közös ősosztályból származó (vagy ugyanazon interfészt megvalósító) osztályok példányaira. A polimorfizmus lehet statikus és dinamikus.

- Statikus polimorfizmus: metódusok túlterhelése, függvénysablonok, osztálysablonok. Satikus, fordításidejű kötés.
- Dinamikus polimorfizmus: metódusok felülírása. Dinamikus, futásidejű kötés.

Öröklődés: Egy osztály örökölhet tulajdonságokat és viselkedésformákat egy másik osztálytól. Az utód osztályban csak az ős osztálytól való eltéréseket kell megadni.

OBJEKTUMORIENTÁLT RENDSZERFEJLESZTÉS

Lépései:

- Analízis (rendszerelemzés)
- tervezés

MUNKAANYAG

- kódolás és
- tesztelés.

Módszerei:

- OMT (Object Modelling Technique) – 1991
  - erős az analízis fázis során, népszerű az adat-intenzív alkalmazásoknál.
- Booch – 1991
  - erős a tervezés fázisában, népszerű az engineering-intenzív alkalmazásoknál.
- OOSE – 1992
  - kiváló támogatást ad a "business engineering"-hez, és igazan csak ez támogatja a követelmény analízist.
- UML – 1997.
  - Az UML (Unified Modeling Language) egy szabványos, általános célú modellező nyelv, melynek segítségével szöveges és grafikus modelleket készíthetünk többek közt:
    - Rendszerekről, Szervezetekről: viselkedésükről, külső és belső kölcsönhatásaikról, stb.
    - Szereplőkről: viselkedésük egy rendszerben,kapcsolatuk más rendszerekkel, stb.
    - Üzleti tevékenységről, folyamatokról:
    - Logikai összetevőkről:azok viselkedéséről, feladataikról, kommunikációjáról egy rendszeren belül, vagy rendszerek között, stb.
    - Szoftverekről, programokról: Az UML az objektumorientált programozás szabványos specifikációs nyelve.
    - Adatbázisokról: elméleti, logikai és fizikai modellekről egyaránt.
  - Az UML grafikus jelöléseket használ rendszerek absztrakt modelljének leírására. Az UML modellek szabvány UML jelölést használó diagramokból állnak.

OBJEKTUMOK

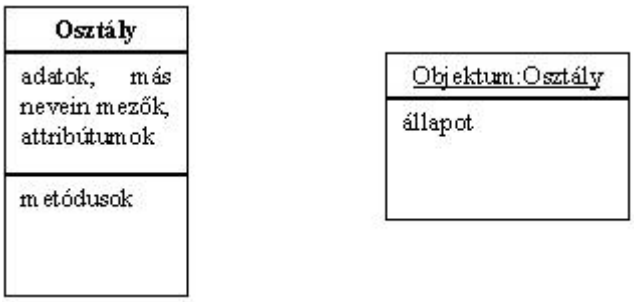
Objektum: Információkat tárol és kérésre feladatokat hajt végre. Adatok és metódusok összessége, mely felelős feladatai elvégzéséért.

Objektum állapota: Adatainak pillanatnyi értékei.

Objektum azonosítása: Egyértelműen azonosítható, azonossága független az állapotától.

Objektum osztály és objektum példány: Az osztály egy típus, amely mintául szolgál az ezen típushoz tartozó példányok (objektumok) létrehozásához. Jelölésük UML-ben:

MUNKAANYAG



1. ábra. Objektum – osztály

Példányváltozó, példánymetódus: A példányváltozók az objektum példányok változói, azaz az állapotukat rögzítő mezők. A példánymetódusok az osztálybeli operációk, azaz metódusok.

Üzenetküldés: Ha egy objektum üzenetet (kérelmet) küld egy másik objektumnak, akkor az üzenetküldő a kliens, a fogadó pedig a szerver objektum. Az üzenetküldés nem más, mint a szerver objektum egy metódusának hívása. Formája: Objektum.Üzenet(Paraméterek)

Interfész objektum: A felhasználóval kommunikáló objektum (tipikusan valamilyen ablak).

Objektum inicializálása: Az objektum kezdeti adatainak beállítása, valamint a működéséhez szükséges kezdeti tevékenységek végrehajtása.

Bezárás, információ elrejtése: A bezárás alatt az adatok és metódusok egybezárását értjük. Az információ elrejtése azt jelenti, hogy az adatok és metódusok közül csak az érhető el kívülről, amelynek külső elérését az interfészen keresztül engedélyezzük.

Láthatóság: Minden mező vagy metódus háromféle lehet elrejtés szempontjából: publikus (az interfészen keresztül tetszőlegesen elérhető), privát (az osztályon kívülről nem érhető el), védett (csak bizonyos helyekről elérhető).

Kód újrafelhasználása: Az egyes osztályokból újabb és újabb objektumok létrehozása kód újrafelhasználásnak minősül, mivel a metódus csak egyetlen helyen, az osztályban létezik fizikailag.

Self paraméter: Egy láthatatlan paraméter, mely minden metódus paraméterlistájában implicit megtalálható. Erről ismeri fel a metódus, hogy mely példány adatain kell végrehajtani a kérelmet.

Mivel a metódus mindig a hívó objektum adatain dolgozik, illetve fordításkor a metódus nem ismerheti az objektum osztályban deklarált adatok címeit, minden metódusnak van egy utolsó, rejtett paramétere, a SELF vagyis saját maga .

A SELF paraméter a hívó objektum címe, 4 byte. Ha a metódus hívása példányból történik, akkor a SELF paraméter a példány címe, ha a metódus hívása metódusból történik, akkor a SELF paraméter továbbadódik.

MUNKAANYAG

Objektum definiálása:

Memória foglalással jár együtt. Egyazon osztály objektumaival egymás értékadása és inicializálása elvégezhető. Legegyszerűbb osztálytörzsön belül definiálni.

```
Pl.: class Kor{
        protected;
        double rx, ry;
        double Kr;
        public:
        void init(double x, double y, double r;
        double Kerulet();
        double Terulet();
        }
```

A függvények definícióját osztálytörzsön kívül kell elvégezni.

```
Pl.: void Kor:: Init(double x, double y, double r){
                                rx=x; ry=y; kr=r;
                                }
```

A Terület függvény definiálása:

```
double Kör:: Terület( ){
                return kr*kr*3.14;
                }
```

A definícióban szereplő Kör az osztályérvényességi kör.

A különböző osztályoknak lehet ugyanolyan nevű tagfüggvénye:

```
Pl.: class Kör{ ...
        double Terület( );
        ...
        };
```

MUNKAANYAG

```
class Ellipszis{
        ...
        doubleTerület( );
        ...
         };
```

```
Double Kör::Terület( ){...};

Double Ellipszis:: Terület( ){...};
```

Osztály tagfüggvényei:

Fajtái:

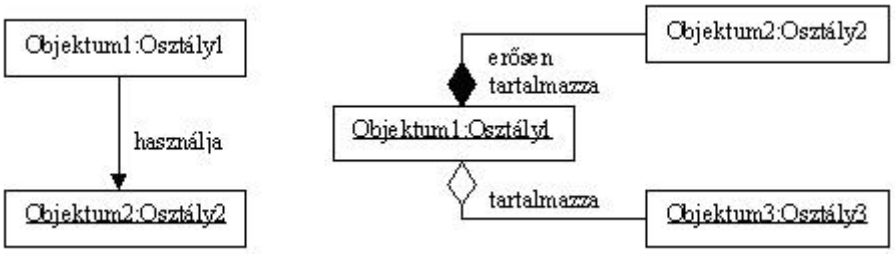
- Kezelő függvény
- Munkavégző függvény
- Segítő függvény
- Elérési függvény
- Konstans függvény

OBJEKTUM ORIENTÁLT TERVEZÉS

1. Objektumok közötti kapcsolatok

Ismeretségi kapcsolat: létük egymástól független, és legalább az egyik használja a másikat.

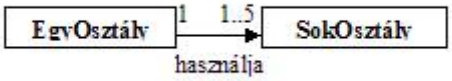
Tartalmazási kapcsolat: az egyik fizikailag tartalmazza a másikat. Ha ez erős tartalmazás, akkor a kapcsolat neve **kompozíció**.



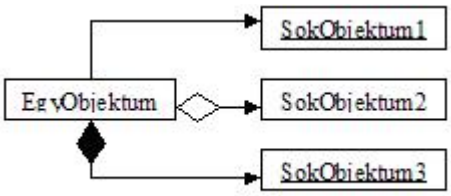
2. ábra. Objektumok közötti kapcsolat

Osztályok közötti kapcsolatok: egy-egy, egy-sok, sok-sok kapcsolat. Osztályok számának (multiplicitás) feltüntetése felsorolással (* jelenti a tetszőleges számút). Ha 0 is lehet, akkor az előfordulás opcionális, egyébként kötelező.

NYÍLT INKAANYAG



3. ábra. Osztálydiagram



4. ábra. Példánydiagram

Szerepnév: Egy adott osztály szerepe egy kapcsolatban.

Együttműködési diagram: Objektumdiagram, amely az egyes operációk működését is bemutatja.

Vezérlő objektum: Kliens objektum, amely a teljes feladat elvégzéséért felelős.

Osztályleírás: Az osztálydiagramon szereplő osztályok bővebb leírása.

Részei:

- osztály neve
- feladatleírás
- közvetlen ős
- objektumok száma
- kapcsolatok
- adatok (neve, típusa, értékhatárai, beviteli formátum és korlátozások)
- metódusok (szöveges vagy/és pszeudokód)).

CRC kártya:

5. ábra. CRC kártya

A CRC kártyákat index kártyákból készítik, amelyekre leírják:

MUNKAANYAG

- Az osztály nevét
- A főosztályait és alosztályait (ha lehetséges)
- Az osztály felelősségét
- Azon osztályok neveit, amelyekkel az osztály együttműködni fog, hogy megvalósíthassa saját felelősségeit
- Szerző

FEJLESZTÉSI LÉPÉSEK

1. Feladatspecifikáció
2. Programterv
3. Objektum- vagy együttműködési diagram
4. Osztálydiagram
5. Osztályleírás (osztály neve, feladatleírás, közvetlen ős, objektumok száma, kapcsolatok, adatok, metódusok)
6. Forráskód
7. Osztályok tesztelése

Használati eset: A rendszer egy tranzakciójának általános leírása.

Forgatókönyv: A használati eset forgatókönyve egy konkrét esetet ír le.

Specializálás: Az a folyamat, melyben egy objektum leírásához egyedi jellemzőket adunk hozzá. (Pl. A téglalap „az egy" paralelogramma (vagy: „olyan" paralelogramma), amelynek szögei derékszögek.)

Általánosítás: Az a folyamat, melyben több objektum leírásából kiemeljük a közös jellemzőket. (Pl. Paralelogramma a téglalap és a rombusz is, mert két-két párhuzamos oldalpárjuk van.)

Öröklődés: Egy meglévő osztály továbbfejlesztése. A meglévő osztály neve ős osztály, a továbbfejlesztetté pedig utód osztály. Az öröklés tranzitív tulajdonság. Egy öröklési lánc legfelső osztálya az alaposztály. Az adatok csak örökölhetők, a metódusok felül is írhatók. Egyszeres öröklés: Egy osztálynak legfeljebb egy őse lehet. Többszörös öröklés: Egy osztálynak több őse is lehet. Öröklés jelölése (osztályhierarchia diagram):

6. ábra. Öröklődés

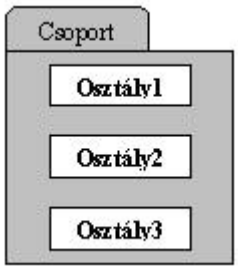
Statikus metódus: A fordító rögzíti a metódus címét.

MUNKAANYAG

Statikus metódus hívása:

- Kívülről (üzenetküldés): Objektum.Metódus.
- Metódusból: [Osztály.]Metódus (Osztály a hívó saját osztálya vagy annak valamelyik őse) vagy [Inherited]Metódus (csak valamelyik ős azonos nevű metódusa aktiválható így).

Csomag: Osztályok elvi csoportosítása. Megvalósítása általában fizikailag is megtörténik (pl. azonos modul). Jelölés:



7. ábra. Csoport

Hozzáférési jogok:

- nyilvános: public
- saját: private.
- védett: protected.
- publikált: published.

Public: Mindenhonnan látható.

Protected: Csak saját objektumból és az utódokból látszik.

Private: Csak saját objektumon belül látszik.

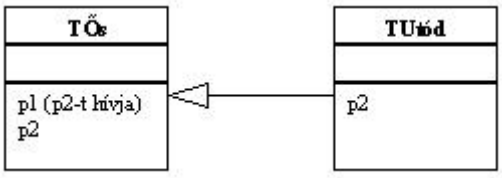
Barát: Egy osztálynak barátja (friend) egy olyan metódus vagy akár teljes osztály, mely hozzáférhet az adott osztály minden mezőjéhez és metódusához. (C++-ban explicit létezik, Object Pascalban az azonos modulban levő osztályok mindig kölcsönösen egymás barátai.)

Polimorfizmus / Sokalakúság: Különböző osztályok rendelkezhetnek azonos nevű metódusokkal. Az üzenetküldő nem köteles ismerni a szerver osztályát (csak azt kell tudja, hogy az ismeri az adott nevű üzenetet).

Futás alatti kötés

Probléma: az ős és az utód példányai p1 hívásakor egyaránt az ős p2 metódusát hívják

MUNKAANYAG



8. ábra. Futás alatti kötés

Megoldás:

Futás alatti kötés: (késői vagy dinamikus kötés) A metódus futási időben kötődik a programhoz.

Virtuális metódus: Olyan metódus, amelynek címét az üzenetküldő objektumtól függően futási időben oldja fel a program. Ha egy osztálynak van virtuális metódusa, kötelező azt inicializálni a virtuális metódus futtatását megelőzően.

Virtuális Metódusok Táblázata: Adott osztály virtuális metódusai belépési címeinek (az örökölt virtuális metódusok címeinek is) táblázata. A VMT címét az osztályhoz tartozó objektum – direkt nem elérhető – mezőként tartalmazza.

Konstruktor: Olyan metódus, amely inicializálja az osztályt és létrehozza a VMT-t. (Delphiben kötelező, ez foglal tárhelyet a példány számára.)

Destruktor: Olyan metódus, amely felszabadítja egy példány által foglalt tárhelyet és elvégzi az objektum megszüntetésével kapcsolatos feladatokat.

Absztrakt metódus: Üres virtuális metódus, amely csak örökítési célokat szolgál. Jelölés: UML-ben dőlt betűvel írjuk a nevét.

Absztrakt osztály: Absztrakt metódust tartalmazó osztály. Nem hozható belőle létre példány! Jelölés: UML-ben dőlt betűvel írjuk a nevét.

Dinamikus metódus (Delphi): A dynamic direktívával megadott metódusok úgy viselkednek, mint a virtuális metódusok, de VMT helyett egy láncolt listában csak az adott osztály saját dinamikus metódusainak címe szerepel. Kevesebb helyet foglalnak, mint a virtuális metódusok, de lassabban fejtik vissza az örökölt dinamikus metódusok belépési címét.

Dinamikus objektum: Olyan objektum, mely futási időben jön létre (ekkor történik a tárfoglalás). (Delphiben minden objektum ilyen.)

MUNKAANYAG

TANULÁSIRÁNYÍTÓ

1. Olvasson be két értéket a billentyűzetről, majd képezze ezek

   - összegét

- különbségét
- szorzatát,

majd írassa ezeket ki a képernyőre! Használja segítségül a képernyőtervet!

9. ábra. 1. feladat képernyőterve

2. Olvassa be egy kör sugarát, majd határozza meg a kör kerületét és területét!

Az előző feladat segítségével tervezze meg a képernyőképet!

3. Elemezze ki a következő programot! Mit csinál a program, ehhez milyen eszközöket használ (osztály, változó…)? (A megoldás C#-ban készült.)

```
using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace ConsoleApplication1

{

   class Program

   {

      static void Main(string[] args)

      {

         uint szám, i=2;

         Boolean prim;

         szám = uint.Parse(Console.ReadLine());
```

MUNKAANYAG

```
            if (szám <= 1)
                Console.WriteLine("\nNem értelmezzük");
            else
            {
                prim = true;
                while ((prim) || (i <= szám / 2))
                {
                    if ((szám % i) == 0)
                        prim = false;
                    else
                        i++;
                }
                if (prim)
                    Console.WriteLine( "pim");
                else
                    Console.WriteLine(" nem prim");
            }
            Console.WriteLine("%d dddd",szám );
            Console.ReadLine();
        }
    }
}
```

4. Írjon programot, mely egy háromszög oldalainak (a, b, c) hosszát olvassa be a billentyűzetről, majd megmondja, hogy a háromszög szerkeszthető-e! (A háromszög szerkeszthető, ha az (a+b>c) és (a+c>b) és (b+c>a) feltétel teljesül.)

Az első feladat segítségével készítsen képernyőtervet!

MUNKAANYAG

5. Írjon programot, mely beolvas egy számpárt a billentyűzetről, majd kiírja a két szám számtani közepét! (Két szám számtani közepe az összegük fele.)

6. Írjon programot egy kocka felszínének és térfogatának meghatározására!

Képernyőtervek:

10. ábra. Képernyőterv az eredmény megjelenítésére

11. ábra. Képernyőterv az adat beírására

Használja segítségül a következő programrészletet! (A megoldás C#-ban készült.)

```
using System;
```

```
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace Kockás
{
    public  class Kocka_szamitas
    {
        double Katlo;
        public double Atlo
        {
            get { return Katlo; }
            set { Katlo = value; }
        }
        double Kfelszin;
        public double Felszin
        {
            get { return Kfelszin; }
            set { Kfelszin = value; }
        }
        double Kterfogat;
        public double Terfogat
        {
            get { return Kterfogat; }
            set { Kterfogat = value; }
        }
        public double Szamol_testatlo(double a)
```

MUNKAANYAG

```
        {
            Katlo = Math.Sqrt(3) * a;
            return Katlo;
        }
        public double Szamol_felszin(double a)
        {
            Kfelszin = 6 * a * a;
            return Kfelszin;
        }
        public double Szamol_terfogat(double a)
        {
            Kterfogat =a*a*a;
            return Kterfogat;
        }
    }
}
```

7. Az előző feladatot megoldva, megértve, elemezze végig a következő programrészletet! (A megoldás C#-ban készült.) Készítsen képernyőtervet (az előző feladathoz hasonlóan)!

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace WindowsFormsApplication1
{
    public  class Koros
    {
```

MUNKAANYAG

```
double Ksugar;
public double Sugar
{
    get { return Ksugar; }
    set { Ksugar = value; }
}
double KKerulet;
public double Kerulet
{
    get { return KKerulet; }
    set { KKerulet = value; }
}

public double Szamol_ker(double ujsugar)
{
    Kerulet = 2 * ujsugar * Math.PI;
    return Kerulet;
}
double KTerulet;
public double Terulet
{
    get { return KTerulet; }
    set { KTerulet = value; }
}
public double Szamol_ter(double ujsugar)
{
```

MUNKAANYAG

```
            Terulet = ujsugar  * ujsugar * Math.PI;

            return Terulet;

        }

    }

}
```

8. A téglatest oldalainak ismeretében számítsa ki a lapátlókat, majd határozza meg a testátlót, a felszínt és a térfogatot! Használja az előző feladatok részletesen ismertetett megoldását!

9. A téglatest oldalainak ismeretében határozza meg az oldalélek összegét, majd írassa ki a legrövidebb oldalél összeggel rendelkező test adatait, felszínét és térfogatát. Használja az előző feladatok részletesen ismertetett megoldását!

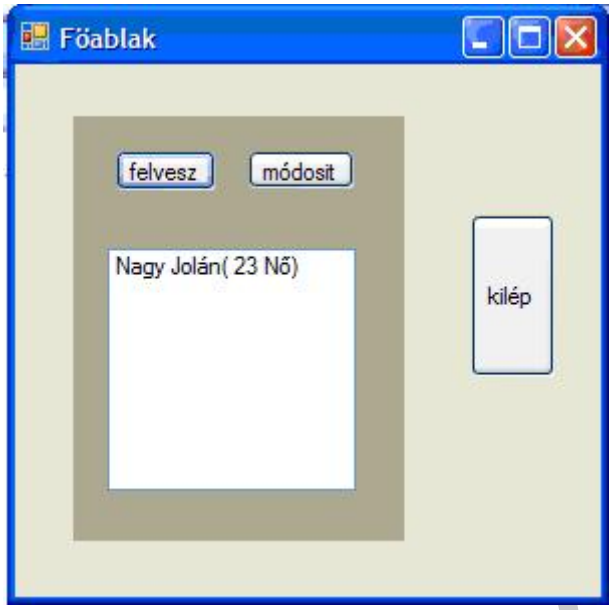
10. Deciliterben megadott súlyt bontsunk hektoliterre, literre és deciliterre!

11. Beolvasott egész számról döntsük el, hogy osztható-e hárommal, a vizsgálatot a számjegyek összegének 3-cel való oszthatóságával végezzük, majd ellenőrizzük le maradékos osztással is.

12. Készítsen programot, mely egy ListBox-ba felveszi egy személy nevét, korát és nemét! Használja segítségül a következő képernyőterveket! Segítségül használja az elkészített programrészletet, majd egészítse azt ki!

12. ábra. 12. feladat képernyőterve

MUNKAANYAG



13. ábra. 12. feladat képernyőterve 2.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace Paros
{
    public enum nem_tipus
    {
        Nő,
        Férfi
    }
    public class Szemely
    {
        string Fnev;
        public string Nev
        {
```

MUNKAANYAG

```
        get { return Fnev; }
        set { Fnev = value; }
    }
    int Fkor;
    public int Kor
    {
        get { return Fkor; }
        set { Fkor = value; }
    }
    nem_tipus Fnem;
    public nem_tipus Nem
    {
        get { return Fnem; }
        set { Fnem = value; }
    }
//  Szemely Fadat;
 //public Szemely Adat
 //{
   //  get { return Fadat; }
    //set { Fadat = value; }
 //}
    public Szemely(string ujnev, int ujkor, nem_tipus ujnem)
    {
        Fnev = ujnev;
        Fkor = ujkor;
        Fnem = ujnem;
```

MUNKAANYAG

```
        }
        public override string ToString()
        {
            return Fnev +"("+ "+Fkor +" "+Fnem +")";
        }
    }
}
```

MUNKAANYAG

ÖNELLENŐRZŐ FELADATOK

1. feladat

Melyik fogalom jellemzőit soroltuk fel?

- Információt tárol, kérésre feladatokat hajt végre
- Belső állapota van, üzeneten keresztül lehet megszólítani
- adattagokból és műveletekből áll (algoritmusok, metódusok)
- Felelős feladatainak korrekt elvégzéséért

2. feladat

Melyik fogalom jellemzőit soroltuk fel?

- Olyan objektum objektumminta vagy típus, mely alapján példányokat (objektumokat) hozhatunk létre

3.feladat

Két programozási szemlélet lényeges jellemzőit látja. Az egyik a strukturált, a másik pedig az objektum orientált szemlélet jellemzője. Határozza meg melyik az objektum orientált és melyik a strukturált módszer!

I. módszer

- Objektumok üzenetei alkotják a programot
- Nincs igazi id˝obeliség
- Lentr˝ol felfelé építkezik
- Az adatokhoz kapcsoljuk az ˝oket kezel˝o programrészeket
- Legkisebb modulja az objektum

II. módszer

- A teljes feladat egy absztrakt utasítás

MUNKAANYAG

- Időbeli sorrendiség alapján történik a részekre bontás
- Felülről lefelé építkezik
- A szorosabban összetartozó adatelemek a folyamattól függetlenül csoportosíthatóak.
- Az adatcsoportok kezelése a programban szétszórva találhatóak.
- Legkisebb modulja az eljárás, melynek adatai elvesznek. Általában globális változókat kell használni.

I. módszer:_____

II. módszer: _____

4. feladat

Írja egymás mellé a megfelelő fogalmakat!

object class	–	üzenet
message	–	objektum osztály
inheritance	–	öröklődés

object class: _____

message: _____

inheritance: _____

5. feladat

Állítsa sorrendbe az objektív orientált tervezés fő lépéseit!

- Készítünk egy példány diagrammot
- Elkészítünk egy optimális osztály hierarchiát, melyből majd az objektumokat létrehozhatjuk
- Megkeressük a tárgykör objektumai és feladatokat rendelünk hozzájuk
  - potenciális objektum jelöltek – főnevek
  - potenciális üzenet jelöltek – igék

6. feladat

Adja meg a következő angol szavak magyar jelentéseit! A feladat megoldásához szükség esetén használja az internet lehetőségeit!

MUNKAANYAG

- object instance
- associations
- information hiding
- runtime binding
- garbage collection

object instance:_____

associations: _____

information hiding: _____

runtime binding: _____

garbage collection: _____

7. feladat

Írjon programot az Ön által tanult programnyelven, egy henger felszínének és térfogatának meghatározására!

MUNKAANYAG

8. feladat

Készítsen programot az Ön által tanult programnyelven egy téglatest felszínének és térfogatának meghatározására!

9. feladat

Írjon programot az Ön által tanult programnyelven, amely bekér egy N, pozitív egész számot, amely legfeljebb 10, és kiírja az Fibonacci sorozat első N elemét!

MUNKAANYAG

MEGOLDÁSOK

1. feladat

Objektum

2. feladat

Osztály

3. feladat

I. módszer: objektum orientált

II. módszer: strukturált

4. feladat

object class	–	objektum osztály
message	–	üzenet
inheritance	–	öröklődés

5. feladat

- Megkeressük a tárgykör objektumai és feladatokat rendelünk hozzájuk
  - potenciális objektum jelöltek – főnevek
  - potenciális üzenet jelöltek – igék
- Készítünk egy példány diagrammot
- Elkészítünk egy optimális osztály hierarchiát, melyből majd az objektumokat létrehozhatjuk

6. feladat

object instance	–	objektum példány/előfordulás
associations	–	objektumok közötti kapcsolatok
information hiding	–	információ elrejtése
runtime binding	–	futásidejű kapcsolatteremtés
garbage collection	–	szemétgyűjtés

MUNKAANYAG

7. feladat

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace bbbbHenger
{
    public  class Henger
    {
        public  double Hterfogat;
        public double Terfogat
        {
            get { return Hterfogat; }
            set { Hterfogat = value; }
        }
        public  double Hfelszin;
        public double Felszin
        {
            get { return Hfelszin; }
            set { Hfelszin = value; }
        }
        public double Szamol_Felszin(double m, double r)
        {
            Hfelszin = 2 * r * Math.PI*(r+m);
            return Hfelszin;
```

MUNKAANYAG

```
        }
        public double Szamol_terfogat(double m, double r)
        {
            Hterfogat = r * r * m * Math.PI;
            return Hterfogat;
        }

    }
}
```

8. feladat

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace Teglatestes
{
    public  class Teglatest
    {
        int Tter;
        public int Ter
        {
            get { return Tter; }
            set { Tter = value; }
        }
        int Tker;
```

MUNKAANYAG

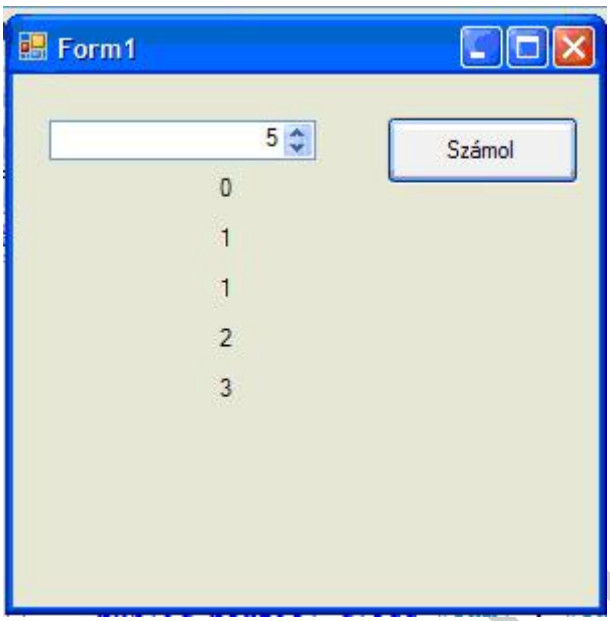
```
        public int Ker
        {
            get { return Tker; }
            set { Tker = value; }
        }

        public int Szamol_Terulet(int uja,int  ujb,int  ujc)
        {
            Ter=uja*ujb *ujc ;
            return Ter;
        }
        public int Szamol_Kerulet(int uja, int ujb, int ujc)
        {
            Ker = 2*(uja * ujb +uja * ujc+ujb *ujc );
            return Ker;
        }


    }
}
```

MUNKAANYAG

9. feladat



14. ábra. 9. feladat képernyőterve

```
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
namespace WindowsFormsApplication1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
```

MUNKAANYAG

```
        }
        private void button1_Click(object sender, EventArgs e)
        {
            int eddig = (int )numericUpDown1.Value;
            Label [] mylabels=new Label [eddig];
            mylabels[0] = new Label();
            mylabels[0].Text = "0";
            mylabels[0].Location = new Point(100,50);
            mylabels[0].Enabled = true;
            this.Controls.Add(mylabels [0]);
            int seged = 1, seged0 = 0, seged1 = 1;
            for (int i = 1; i < eddig; i++)
            {
                mylabels[i] = new Label();
                mylabels[i].Location = new Point(100,mylabels [0].Location .Y+i
                    *(mylabels [0].Height +2) );
                mylabels[i].Enabled = true;
                mylabels[i].Text = seged.ToString();
                this.Controls.Add(mylabels[i]);
                seged = seged0 + seged1;
                seged0 = seged1;
                seged1 = seged;
            }
        }
    }
}
```

MUNKAANYAG

IRODALOMJEGYZÉK

FELHASZNÁLT IRODALOM

Illés Zoltán: Programozás C# nyelven; Jedlik Oktatási Stúdió, Budapest, 2008

Trey Nash: C# 2008; Panem Kft., Budapest, 2008

Angster Erzsébet: Objektum orientált tervezés és programozás; Akadémia Nyomda, Martonvásár, 2002

MUNKAANYAG

A(z) 1155-06 modul 019-es szakmai tankönyvi tartalomeleme felhasználható az alábbi szakképesítésekhez:

A szakképesítés OKJ azonosító száma:	A szakképesítés megnevezése
54 481 01 1000 00 00	CAD-CAM informatikus
54 481 04 0010 54 01	Gazdasági informatikus
54 481 04 0010 54 02	Infostruktúra menedzser
54 481 04 0010 54 03	Ipari informatikai technikus
54 481 04 0010 54 04	Műszaki informatikus
54 481 04 0010 54 05	Távközlési informatikus
54 481 04 0010 54 06	Telekommunikációs informatikus
54 481 04 0010 54 07	Térinformatikus

A szakmai tankönyvi tartalomelem feldolgozásához ajánlott óraszám:
20 óra

MUNKAANYAG

MUNKAANYAG

A kiadvány az Új Magyarország Fejlesztési Terv
TÁMOP 2.2.1 08/1-2008-0002 „A képzés minőségének és tartalmának fejlesztése” keretében készült.
A projekt az Európai Unió támogatásával, az Európai Szociális Alap társfinanszírozásával valósul meg.

Kiadja a Nemzeti Szakképzési és Felnőttképzési Intézet
1085 Budapest, Baross u. 52.
Telefon: (1) 210-1065, Fax: (1) 210-1063

Felelős kiadó:
Nagy László főigazgató