

Varga Márton

INBPM0315-17

Magas szintű programozási nyelvek 2*

1.hét

OO szemlélet :

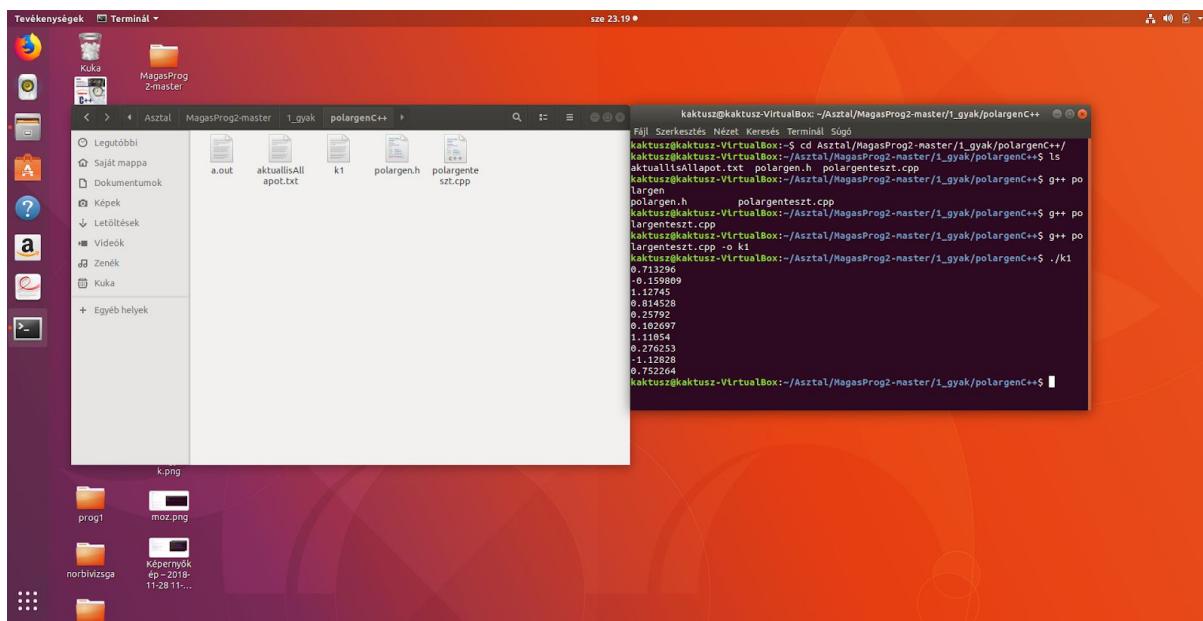
Az OO egy a objektumok fogalmán alapuló programozási paradigma. Erre a céral létrehozunk ebben a feladatban egy osztályt. Egy ciklus 10-szer végig megy a main-en, és minden alkalommal létrahoz egy új polargen objektumot, majd kiírja a következő nevű függvény eredményét: A polargen konstruktora alapján a nincstarolt bool igaz lesz ezért a következő végre lesz hajtva, és addig hajtja végre a do while –on belüli műveleteket, ameddig a w kisebb nem lesz mint 1. aztán visszatér r*v1 értékkel. Vagy ha nincstárolt értéke hamis akkor visszatér a tarolt értékkel.

Java:

The screenshot shows the Eclipse IDE interface with the following details:

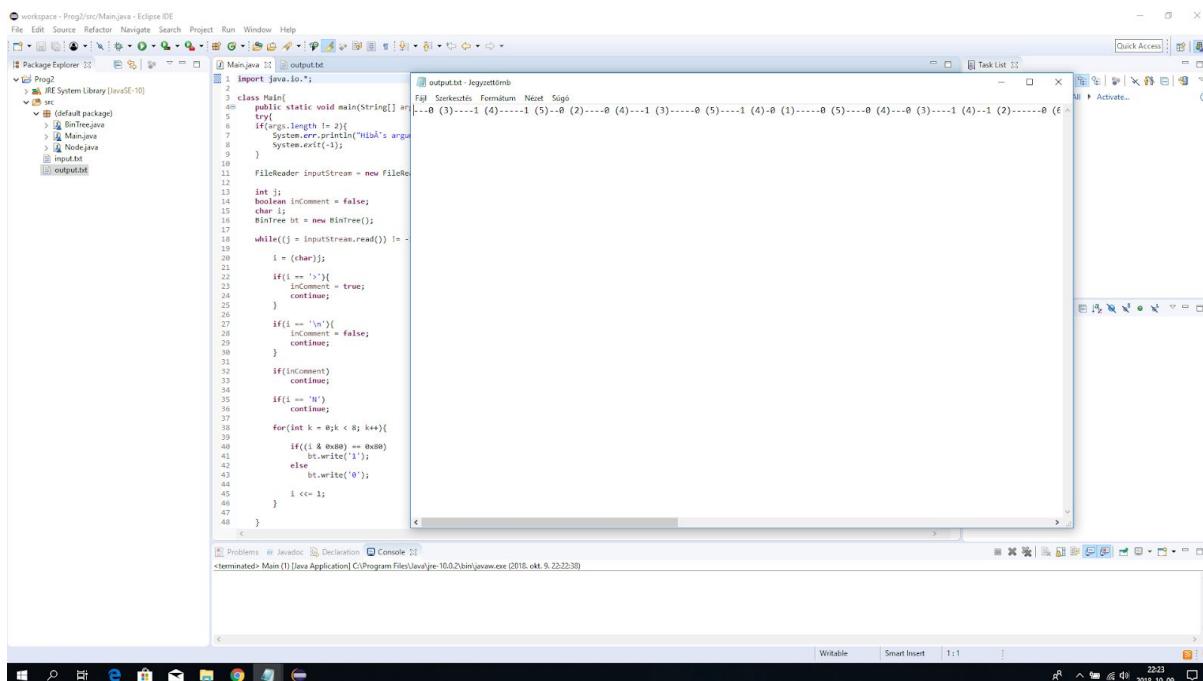
- Title Bar:** eclipse-workspace - Prog2_gyak/src/PolarGen.java - Eclipse IDE
- Left Margin:** 41 lines (33 sloc) 709 Bytes
- Code Editor:** The code for the `PolarGen` class is displayed. It includes a constructor that initializes `nincstarolt` to `true`, a method `kovetkezo()` that generates random values `u1` and `u2`, and a `do-while` loop that continues until `w` is greater than 1. Inside the loop, it calculates `v1` and `v2` using the formula $v = \sqrt{u_1^2 + u_2^2}$. The loop then updates `u1` and `u2` and `w` using the formula $u = \sqrt{u_1^2 + u_2^2}$.
- Project Explorer:** Shows the project structure with packages `Prog2` and `PolarGen`, and source files `PolarGen.java`, `Voda.java`, and `PolarGenTest.java`.
- Console:** Displays the output of the application, showing several random numbers generated by the program.

C++:



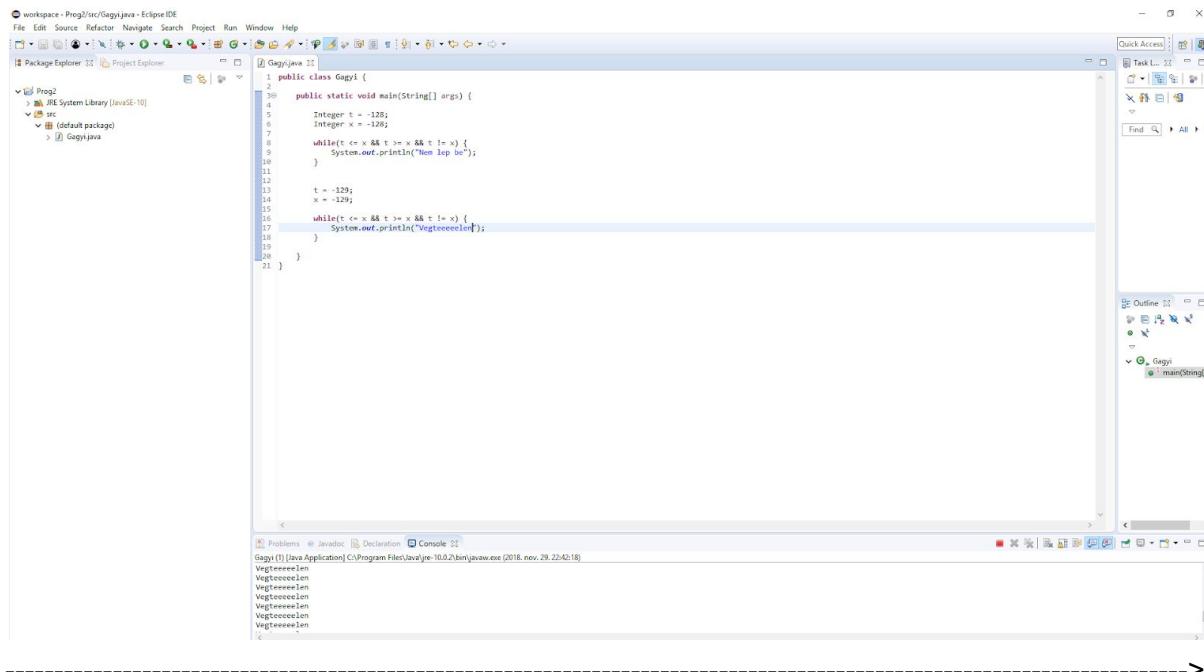
Homokozó:

A feladat megoldásához tutorként Bátfai Norbert tanár úr forráskódját vettet igénybe. igyekeztem a feladathoz hűen rámutatni a pointerekre illetve a referenciaiák. A feladatot értelmeztem és sikeresen fordítottam, futtattam.



“Gagyi”

Hozzunk létre két Integer objektumot, és ha a két szám egyenlő és -128, és 127 között van, akkor a while ciklus egyből végetér mert az Integer objekt előre letárolt értékekkel dolgozik de ha ez az érték mondjuk 129, akkor az érték nincs előre tárolva és ilyenkor egy new int() értékkel tér vissza ekkor nem lesznek ugyan azok az objektumok, csak ugyan azt az értéket veszik fel tehát a while ciklus örökké fog tartani.



```
public class Gagy {
    public static void main(String[] args) {
        Integer t = -128;
        Integer x = -128;
        while(t <= x && t >= x && t != x) {
            System.out.println("Nem lep be!");
        }
        t = -129;
        x = -129;
        while(t <= x && t >= x && t != x) {
            System.out.println("Vegteeeeleen");
        }
    }
}
```

Gagy [Run Application] Declaration Console

Gagy [Run Application] C:\Program Files\Java\jre-10.0.2\bin\javaw.exe (2018. nov. 29. 22:40:18)

Vegteeeeleen
Vegteeeeleen
Vegteeeeleen
Vegteeeeleen
Vegteeeeleen
Vegteeeeleen
Vegteeeeleen

Yoda:

A Yonda condition egy programozási stílus ahol a kifejezéseket fordítva írjuk ki. "ez".equals(myString) esetén a program hamissal tér vissza, míg myString.equals("ez") esetén a program java.lang.NullPointerException hibaüzenetet dob vissza. Ez hasznos lehet, mivel yoda conditionnal elkerülhető hogy hibát dobjon ki a program amikor null objektummal dolgozunk.

2.hét

Ciklomatikus komplexitás:

A ciklomatikus komplexitás azt számmolja, hogy a programban hogy egyéni út járható be. A ciklomatikus komplexitás számolásához az oclint nevű programot használtam, majd az oclint átláthatatlansága miadt inkább a lizaard.ws nevű oldalt használtam. A BBP és a fullscreen egyik eredeti verziójának néztem meg a komplexitását.

Function Name	NLOC	Complexity	Token #	Parameter #
screen:screen	4	1	22	
screen:setFullScreen	10	4	62	
screen:getFullScreenWindow	3	1	12	
screen:restoreScreen	7	2	35	

Szülő-gyerék:

Ebben a feladatban a szülő-gyerek objektum kapcsolatáról írtunk programot. A szülő objektumnak létrehoztam egy alosztályt majd a main-en belül hívtam meg a szülő, illetve a gyerek objektumban lévő függvényeket. Bár a szülő része az alosztály nem lehet a szülőn keresztül az alosztály függvényét meghívni. Akkor se lehet hozzáérni ha például egy gyerek objektumba hozunk létre egy szülő osztályt.

C++:

The screenshot shows the Geany IDE interface. On the left is the code editor with the file `szulo_gyerek.cpp` open. The code defines two classes: `Rectangle` and `Square`, both derived from a base class `Shape`. The `Shape` class has a protected member `v`. The `Rectangle` class has public methods `getWidth()` and `getHeight()` which return `m_width` and `m_height` respectively. The `Square` class has a constructor that initializes `m_width` and `m_height` to the same value `side`. It also overrides the `getWidth()` and `getHeight()` methods to return `side`. The `main` function creates a `Shape` pointer and points it to a `Square` object, then prints its width and height. On the right is a terminal window showing the compilation command `g++ szulo_gyerek.cpp -o szulogyerek` and the resulting error message: `szulo_gyerek.cpp: In function 'int main()': szulo_gyerek.cpp:32:35: error: 'class Rectangle' has no member named 'getArea'`.

Java:

The screenshot shows the Eclipse IDE interface. On the left is the code editor with the file `SzuloGyerek.java` open. The code defines three classes: `Vader`, `SkywalkerLajos`, and `SzuloGyerek`. The `Vader` class has protected fields `v_name` and `v_age`, and methods `setAge` and `setName`. The `SkywalkerLajos` class extends `Vader` and overrides the `getName` method. The `SzuloGyerek` class has a static `main` method that creates a `Vader` object, sets its name to "Vader", and then prints the names of a `SkywalkerLajos` object and the `Vader` object. The right side of the interface shows the `Outline` view with the class hierarchy and the `Problems` view which shows an unresolved compilation problem: `The method getName() is undefined for the type Vader`.

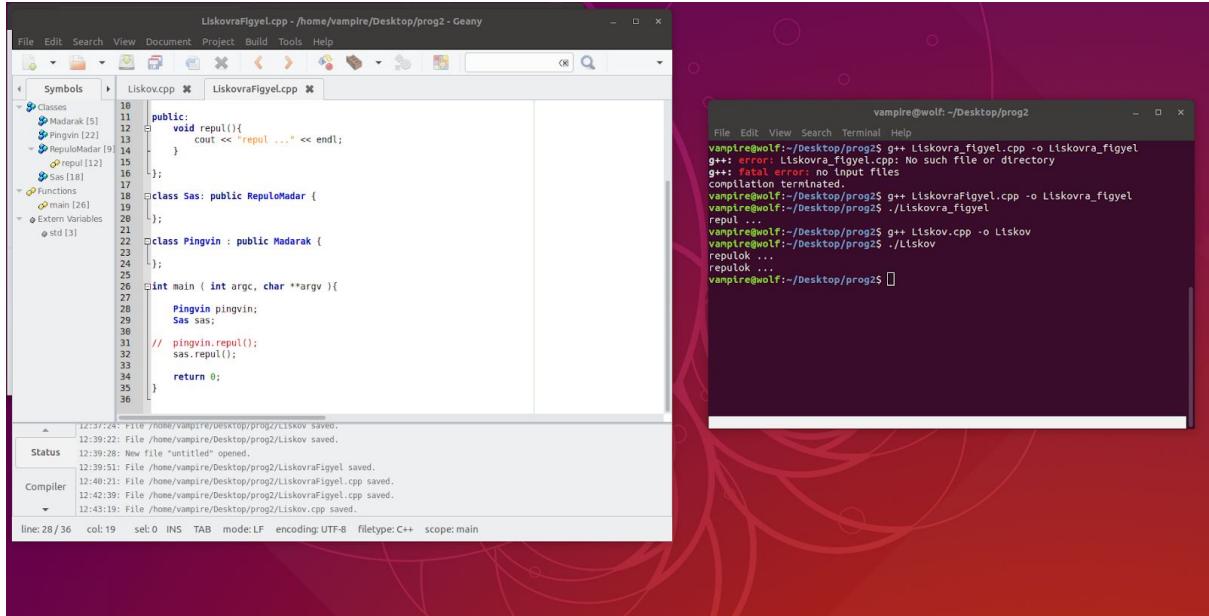
Liskov:

A Liskov elv szerint az osztályokat úgy kell megtervezni, ha az ős osztály objektuma helyet a gyerek

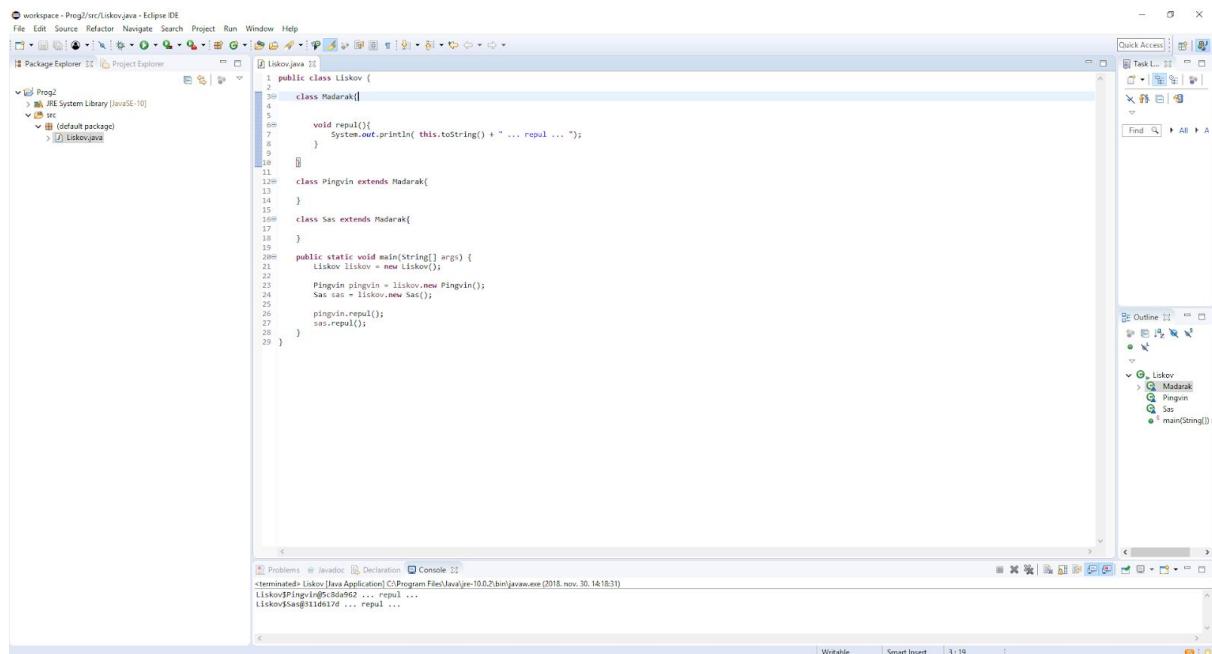
osztály objektumát használjuk az applikációnkban, akkor ettől még nem szabadna, hogy az applikáció elromoljon.

Az első képen a C++ verziót szemlélteti a másik kettő pedig a javat hivatott.

C++:



Java:



```

1 public class Liskova_figylej {
2     class Madarak{
3     }
4     class RepuloMadar extends Madarak{
5     }
6     void repul(){
7         System.out.println( this.toString() + " ... repul ... ");
8     }
9     class Pingvin extends Madarak{
10    }
11    class Sas extends RepuloMadar{
12    }
13    public static void main(String[] args) {
14        Liskova_figylej liskov = new Liskova_figylej();
15        Pingvin pingvin = liskov.new Pingvin();
16        Sas sas = liskov.new Sas();
17        // pingvin.repul();
18        // sas.repul();
19    }
20 }
```

Outline view:

- Liskova_figylej
 - RepuloMadar
 - Pingvin
 - Sas
- main(String[])

Console tab output:

```
terminated: Liskova_figylej [Java Application] C:\Program Files\Java\jre10.0.2\bin\javaw.exe (2018. nov. 30. 12:29:45)
Liskova_figylej$Sas@512ddff17 ... repul ...
```

3.hét

A feladatokat tesomal kozosen sikerult megoldani.

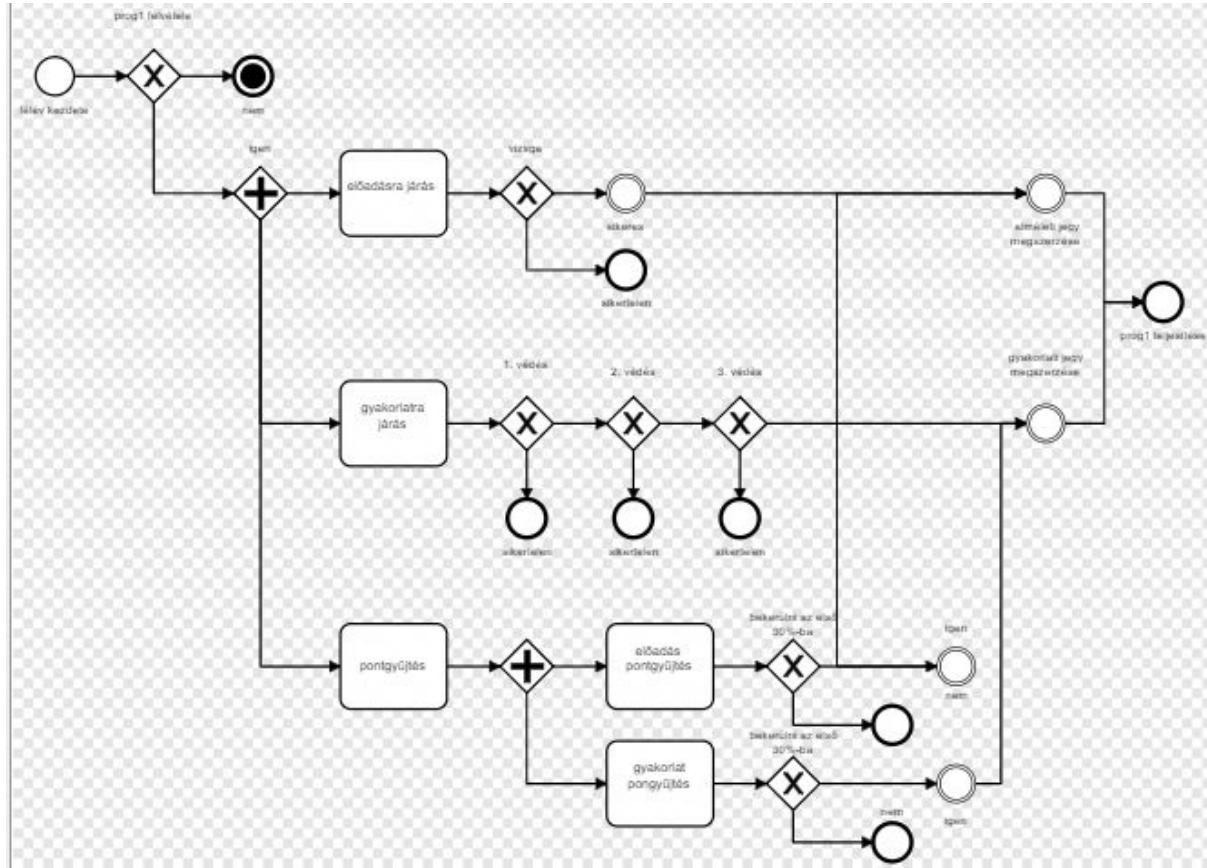
BPMN:

Business Process Model and Notation (BPMN) egy grafikus reprezentációja a speciális üzleti folyamatoknak az üzleti folyamatmodellekben.

Egy folyamatábra amin könnyen el lehet igazodni.

A feladat megoldásához az egyik plugint használtam.

A feladat szerint rajzoltam egy tevékenységet, ami az átlagos prog1 tárgyat jellemzi.



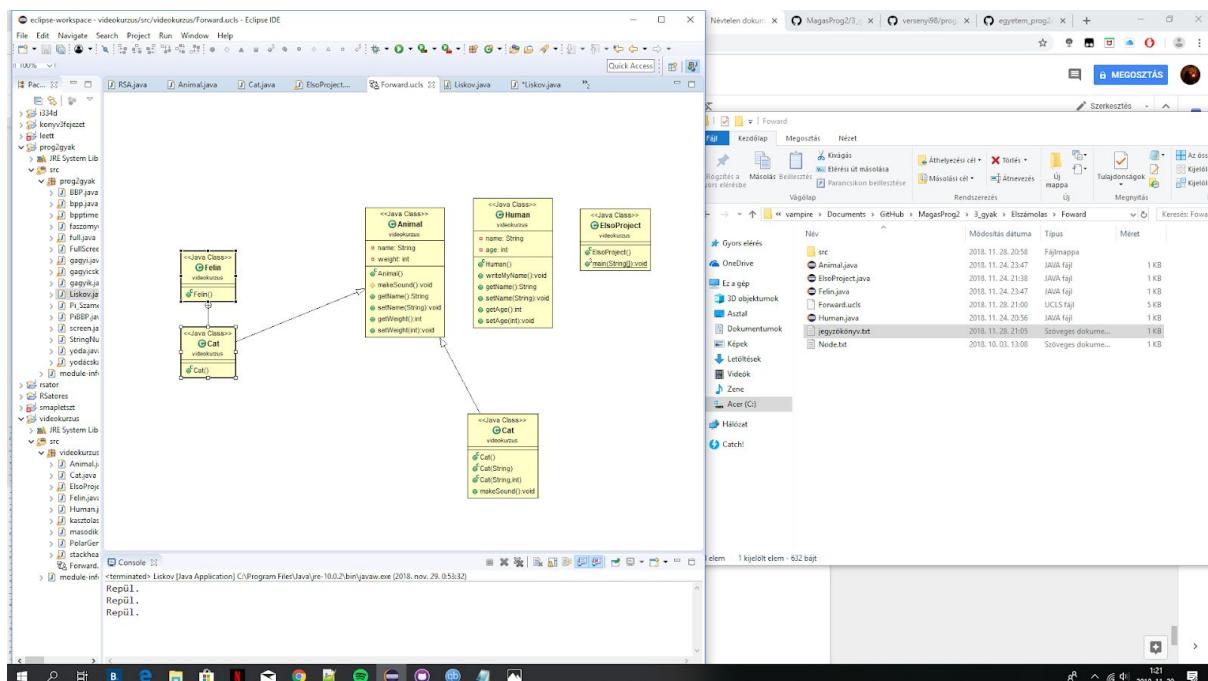
Forward engineering UML osztálydiagram :

Ehez a feladathoz a eclipse plugint használtam.

Forward engineering a Reverse engineering ellentéte, szóval a helyet, hogy egy már előre megírt

kód ból generálnánk egy diagrammot, itt diagrammot kell alkotnunk, majd abból kell kódot generáltnunk.

Itt egy egyszerű osztályt készítettem a példámhoz.



Reverse engineering UML osztálydiagram

Az UML egy gyakorlati, objektum orientált modellező megoldás, nagy méretű programrendszerek

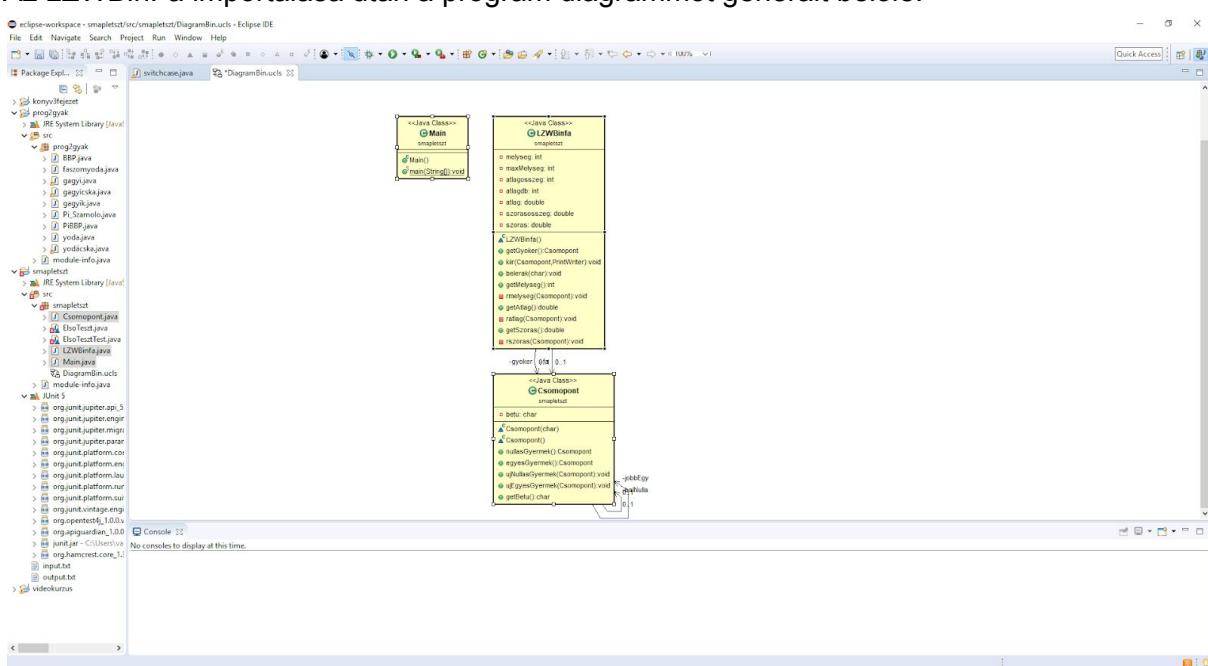
modelljeinek vizuális dokumentálására alkalmas eszköz.

A reverse engineering segít vizualizálni a nagyobb, bonyolultabb programokat.

Ehez a feladathoz a eclipse plugint használtam.

A feladat szerint előbb kellett megcsinálni a programot, majd UML diagrammot készíteni.

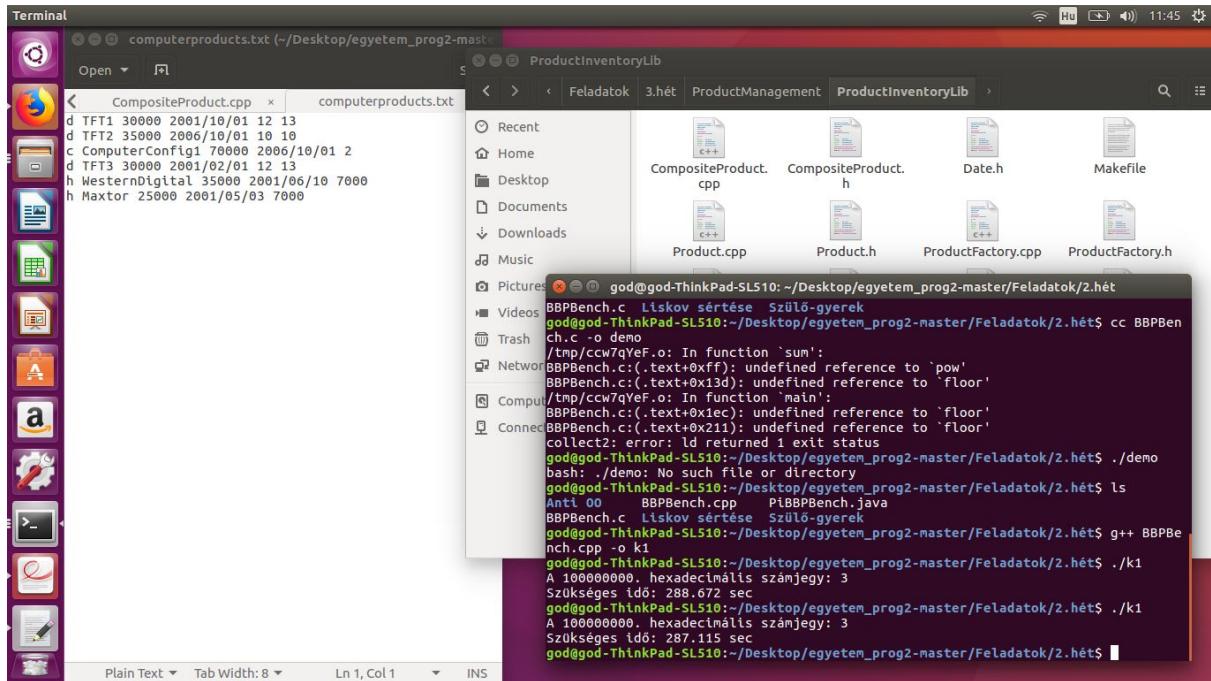
Az LZWBinFa importálása után a program diagrammot generált belőle.



“Egy esettan” :

Igyekeztem értelmezni a megadott feladatokat és meglévő kódokat. Lefuttattam.

A BME-s C++ tankönyv 14. fejezetét (427-444 elmélet, 445-469 az esettan) dolgoztam fel.



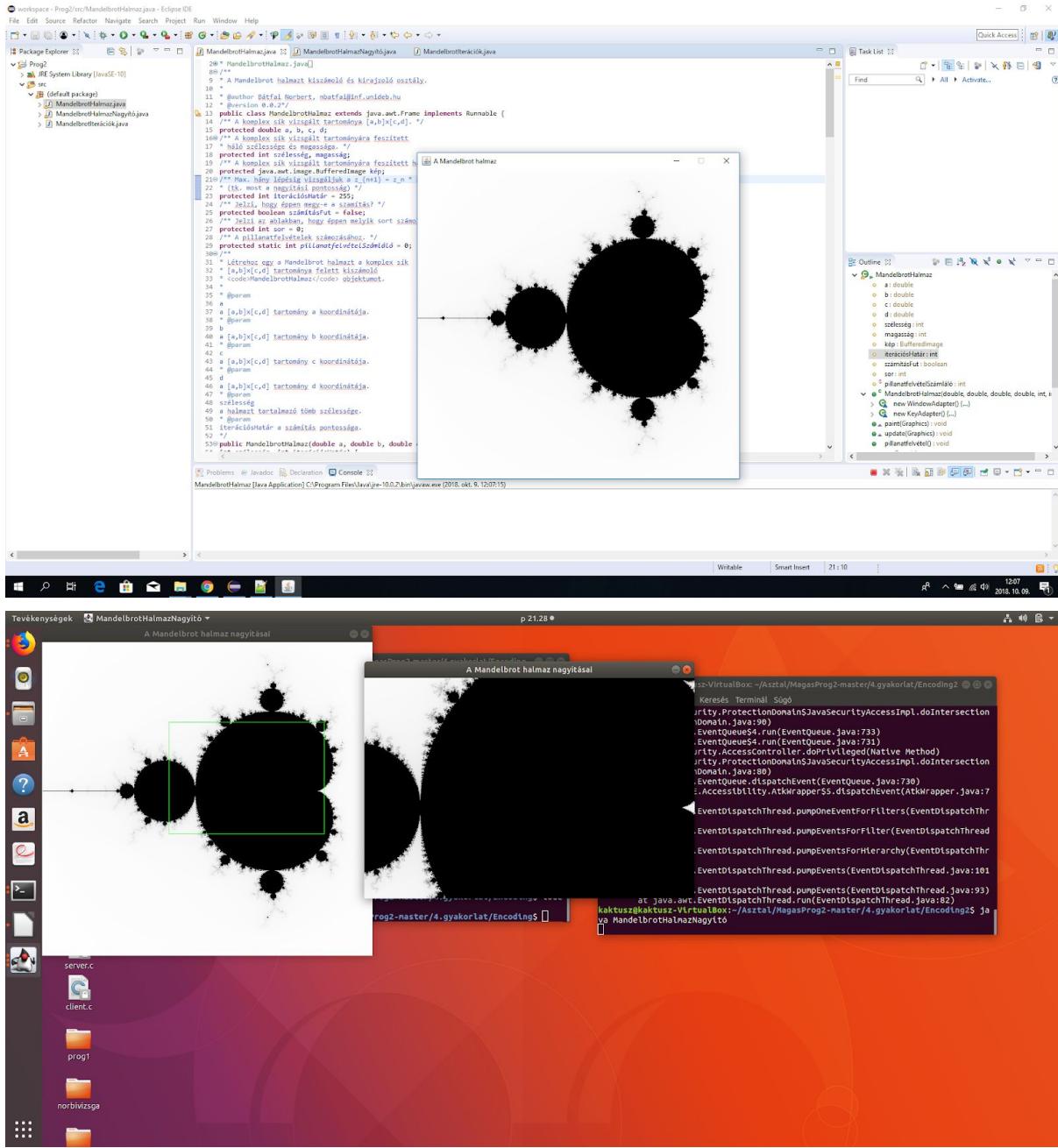
The screenshot shows a Linux desktop environment with a terminal window open in the foreground. The terminal window has tabs for 'computerproducts.txt' and 'ProductInventoryLib'. The current tab displays the contents of 'ProductInventoryLib' which includes files like 'CompositeProduct.cpp', 'CompositeProduct.h', 'Date.h', 'Makefile', 'Product.cpp', 'Product.h', 'ProductFactory.cpp', and 'ProductFactory.h'. Below the terminal, a file manager window is visible, showing a file tree with categories like Recent, Home, Desktop, Documents, Downloads, Music, Pictures, Videos, and Trash. The terminal output is as follows:

```
BBPBench.c Liskov sértése Szülő-gyerek
god@god-ThinkPad-SL510:~/Desktop/egyetem_prog2-master/Feladatok/2.hét$ cc BBPBench.c -o demo
/tmp/crw7qYeF.o: In function 'sum':
BBPBench.c:(.text+0x1d): undefined reference to `pow'
BBPBench.c:(.text+0x1ec): undefined reference to `floor'
BBPBench.c:(.text+0x1ec): undefined reference to `floor'
BBPBench.c:(.text+0x211): undefined reference to `floor'
collect2: error: ld returned 1 exit status
god@god-ThinkPad-SL510:~/Desktop/egyetem_prog2-master/Feladatok/2.hét$ ./demo
bash: ./demo: No such file or directory
god@god-ThinkPad-SL510:~/Desktop/egyetem_prog2-master/Feladatok/2.hét$ ls
Anti OO BBPBench.cpp PiBBPBench.java
BBPBench.c Liskov sértése Szülő-gyerek
god@god-ThinkPad-SL510:~/Desktop/egyetem_prog2-master/Feladatok/2.hét$ g++ BBPBench.cpp -o k1
god@god-ThinkPad-SL510:~/Desktop/egyetem_prog2-master/Feladatok/2.hét$ ./k1
A 100000000. hexadecimális számjegy: 3
Szükséges idő: 288.672 sec
god@god-ThinkPad-SL510:~/Desktop/egyetem_prog2-master/Feladatok/2.hét$ ./k1
A 100000000. hexadecimális számjegy: 3
Szükséges idő: 287.115 sec
god@god-ThinkPad-SL510:~/Desktop/egyetem_prog2-master/Feladatok/2.hét$
```

4.hét

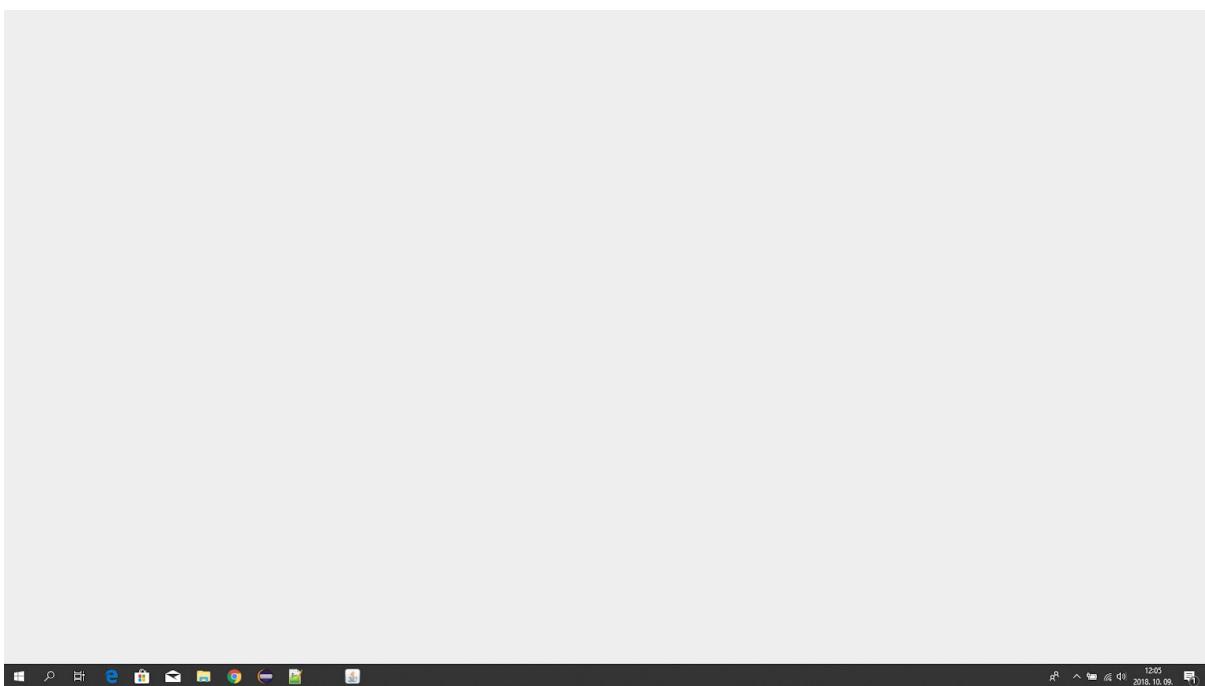
Encoding :

A feladatban adott link letöltése után egyszerűen futtatni kell a `MandelbrotHalmazNagyító.java` nevű fájlt. A fordítás `MandelbrotHalmazNagyító.java` -val az eclipsbe történik. Futtatni pedig `java MandelbrotNagyító` –val. A `MandelbrotHalmazNagyító`ban rá tudunk közelíteni a mandelbrot halmazra.



Full screen:

Egy programot kellett készíteni, ami teljes képernyőt befedi.



i334d1c48:

A leet vagy leetspeak egy, az interneten létrejött jelrendszer, ami a hétköznapi karakterek helyett más karaktereket használ.

Az interneten gyakran használt a leet, de töbnyire az emberek csak az egyszerűbb változatát

használják (e->3;t->7 stb.).

A program beolvas egy fájlt, majd annak a leet változatát adja vissza.

A screenshot of the Eclipse IDE showing the code editor and terminal panes for the i334d1c48 Java program.
Code Editor:

```
1 package i334d;
2
3 import java.io.BufferedReader;
4
5 public class i334d1c4 {
6     private static String[] leetAbcs = new String[36];
7     private static Random rand = new Random();
8
9     public static void main(String[] args) throws Exception{
10         BufferedReader br = new BufferedReader(new FileReader("input2.txt"));
11         for(int i=0;i<leetAbc.length();i++){
12             leetAbc[i]=br.readLine();
13             //System.out.println(leetAbc[i]);
14         }
15         br.close();
16         System.out.println("Irja be amit az akar fordítani");
17         Scanner scanner = new Scanner(System.in);
18         String fromSt= scanner.nextLine();
19         FromSt = FromSt.toUpperCase();
20         String Res="";
21         for(int i = 0; i < FromSt.length() ; i++) {
22             if(FromSt.charAt(i)==' ') Res+= ' ';
23             else Res+= leetAbc[i];
24         }
25         System.out.println("EREDMENY:-----");
26         System.out.println(Res);
27     }
28
29     private static String changeLetter( char Tochange ) {
30         int i=0;
31         String[] chosefrom;
32         while(i<36){
33             if(leetAbc.length() == i){
34                 String OtherCharacter = "" + Tochange;
35                 return OtherCharacter;
36             }
37         }
38         //Tocheck if character is == Tochange
39     }
40 }
```

Terminal:

```
Irja be amit az akar fordítani
stöt akkarok proghol
EREDMENY:-----
S+Ö57 ^1<X/-\1'()l< l>l'[],§ö1
```

Leettranslator:

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows the project structure with a file named `Leettranslator.java` under the `src` folder.
- Code Editor:** Displays the Java code for `Leettranslator.java`. The code reads from a file named `input.txt` and writes to a file named `output.txt`. It contains logic to replace specific characters based on their ASCII values, such as replacing 'a' with '4' and 'b' with '6'. It also handles punctuation and whitespace.
- Output View:** Shows the contents of the `output.txt` file, which contains the processed text from `input.txt`.
- Console View:** Shows the terminal output of the application running. It includes the command used to run the program, the input file path, and the output file path.

```
workspace - Prog2/src/Leettranslator.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
[...] Leettranslator.java [...] input.txt
Project System Library [JavaSE-1.8]
src (default package)
> Leettranslator.java
input.txt
Leettranslator.jar
37     if (text.contains("a")) {
38         text = text.replace('a', '4');
39     }
40     /*if (text.contains("z")) {
41         text = text.replace('z', '5');
42     }*/
43     if (text.contains("g")) {
44         text = text.replace('g', '6');
45     }
46     if (text.contains("t")) {
47         text = text.replace('t', '7');
48     }
49     if (text.contains("b")) {
50         text = text.replace('b', '6');
51     }
52     //English
53     if (text.contains("c")) {
54         text = text.replace('c', '(');
55     }
56     if (text.contains("s")) {
57         text = text.replace('s', '$');
58     }
59     if (text.contains("l")) {
60         text = text.replace('l', '1');
61     }
62     if (text.contains("i")) {
63         text = text.replace('i', '!');
64     }
65     return text;
66 }
66@ public static void main(String[] args) throws IOException {
67     Scanner scanner = new Scanner(System.in);
68     System.out.print("Kérlek a fájl nevét: ");
69     String filename = scanner.nextLine();
70     File file = new File(filename);
71     Scanner scanner2 = new Scanner(file, StandardCharsets.UTF_8);
72     System.out.println("Fájl beolvása.");
73     fullFileContents = fullFileContents.toLowerCase();
74     fullFileContents += scanner2.nextLine();
75     System.out.println("Fájl olvasása.");
76
77     PrintWriter writer = new PrintWriter("output.txt", "UTF-8");
78     writer.print(fullFileContents);
79     writer.close();
80     System.out.println("A fordított szöveg az output.txt-ben megtalálható.");
81 }
82 }
83 }
```

```
Fajl: Szerkesztés Formátum: Nézet: Súgó
32 36y 1337 7r4n$1470r 4160r!7mu$ 177 42 Inpu7 $20v36 h06y 73$2731n! 13h35$3n.

Problems Declaration Console
<Terminal> Leettranslator [Java Application] C:\Program Files\Java\jre10.0.2\bin\javaw.exe (2016. okt. 9. 12:22:18)
Kérlek a fájl nevét: C:\Users\vampire\workspace\Prog2\src\input.txt
Fájl beolvása.
Fájl olvasása.
A fordított szöveg az output.txt-ben megtalálható.
```

5.hét

JDK osztályok:

Ezt terjedelmesebben az összefoglalóban igyekeztem bemutatni. De léynegében Boost C++ programot kell hazsnálni és ott egyik részlét beimpelmentéálni .hogy ki tudjuk listázni a JDK osztályokat.

Source Code:

https://github.com/martva383/prog2/tree/master/5_gyak/JDK%20oszt%C3%A1lyok

Másoló-mozgató szemantika :

copy assignment - le másolja az objektum tartalmát eredményként megadott helyre.

move assignment - kicseréli az objektum tartalmát.

copy constructor - konstruktor ami az objektum konstruktorába megadott értéket másolja.

move constructor – copy konstruktorral ellentétben itt nem átmásoljuk az értéket hanem átmozgatjuk(a régi objektum tartalmát mind itt, mind a move assignment esetén kitöröljük). A move assignmentet, illetve a move konstruktor általában az std::move()-val hívjuk meg.

The screenshot shows a web-based IDE interface for OnlineGDB beta. The code in main.cpp implements copy and move assignment operators for a Java class. The output window shows the program's execution and its results.

```

1 #include <iostream>
2 using namespace std;
3
4 class Java {
5 {
6     public:
7         const char* majom;
8         Java(const char* majom){this->majom=majom;}
9         // copy ctor
10        Java(const Java &A2){
11            majom=A2.majom;
12            cout << "copy" << endl;
13        }
14
15        // move assign ctor
16        Java& operator=(Java&& A2){
17            majom = A2.majom;
18            A2.majom = nullptr;
19            cout << "move" << endl;
20            return *this;
21        }
22    };
23
24 int main(){
25     Java a("Java Java Java Java ");
26     cout << a.majom << endl;
27     Java a2(a);
28     cout << a2.majom << endl;
29     a = move(a2);
30     cout << a.majom << endl;
31     return 0;
32 }

```

Java Java Java
copy
Java Java Java Java
move

...Program finished with exit code 0
Press ENTER to exit console.

The screenshot shows a desktop environment with a terminal window and a code editor window. The terminal window displays a command-line session with the user navigating through files and running programs. The code editor window shows the same C++ code as the OnlineGDB screenshot.

```

Földi Szerkesztés Nézet Keresés Terminál Súgó
asd.cpp gagyl.java kodolas.txt 'prog2 (1).pdf'
haligato@172:~/Asztal/sokszar$ touch masolomozgato.cpp
haligato@172:~/Asztal/sokszar$ ls
Anttoo bittfa.java hvp madar.java
a.out bittfa.cpp kbjoda.java masolomozgato.cpp 'p
asd.cpp gagyl.java kodolas.txt poligen.java
haligato@172:~/Asztal/sokszar$ g++ masolomozgato.cpp -o k1
haligato@172:~/Asztal/sokszar$ ./k1
korte korte korte korte korte
copy
korte korte korte korte korte
move
haligato@172:~/Asztal/sokszar$ ./k1
korte korte korte korte korte
copy
korte korte korte korte korte
move
haligato@172:~/Asztal/sokszar$ g++ masolomozgato.cpp -o k1
haligato@172:~/Asztal/sokszar$ ./k1
Java Java Java Java
copy
Java Java Java Java
move
haligato@172:~/Asztal/sokszar$ 

```

Hibásan implementált RSA törése:

Itt lényegében hibásat /tört kódóból kell implementálni titkosított szöveget.

Az előadás foliája alapján igyekeztem létrehozni akodomat meklyben array ben tároltam a tiszta szöveget

melyet a length függvény összeszámoltam és azt bevittem egy for függvény a java.math.BitEngert

Majd a kódolásnál pedig for függvénnnyel teszteltem a java.modPow használatát.

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows the project structure with files like RSA.java and RSAPedja.java.
- Code Editor:** Displays the RSA.java code, which contains Java code for generating RSA keys and performing encryption/decryption.
- Console Tab:** Shows the output of the RSA application, which includes a long string of characters representing the generated RSA keys and a message.

Változó argumentumszámú ctor :

Source code:

https://github.com/martva383/prog2/tree/master/5_gyak/V%C3%A1ltozoargumentumsz%C3%A1m

Összefoglaló:

Boost a C++ Szabványügyi Szervezet programkönyvtárakon dolgozó csoportja által kezdeményezett szervezet, amely új C++ könyvtárak kifejlesztését tüzte ki célul. Több könyvtár ebben a Boost forrás osztályozásban található.

A Boost sok multiplatform könyvtár halmaza, ami olyan feladatokat és struktúrákat támogat, mint a lineáris algebra, pszeudó-random szám generálás, többszálú programozás, metaprogramozás, lambdák, tetszőleges méretű számok stb. A munkahelyek jelentős részén használnak Boostot, így ismerete jelentősen növelheti az esélyeinket. C++11-es szabvány sok minden átvett a Boostból (pl.: Többszálú programozás, Smart pointerek, Lamdbák).

A Boost C++ könyvtárak a modern könyvtárak gyűjteménye a C++ szabvány alapján. A forráskód a Boost Software License alatt kerül kiadásra, amely lehetővé teszi, hogy bárki ingyen használja, módosítsa és terjessze a könyvtárakat. A könyvtárak platformfüggetlenek és támogatják a legnépszerűbb fordítóprogramokat, valamint sokat, amelyek kevésbé ismertek. A Boost közössége felelős a Boost könyvtárak fejlesztéséért és közzétételéért. A közösség küldetésnyilatkozata a minőségi könyvtárak fejlesztése és gyűjtése, amelyek kiegészítik a normál könyvtárat. Azok a könyvtárak, amelyek értékesnek bizonyulnak, és fontosak a C++ alkalmazások fejlesztéséhez, jó eséllyel

rendelkeznek abban, hogy egy bizonyos pontban bekerüljenek a standard könyvtárba.

A Boost közösség 1998 körül alakult, amikor a szabvány első változata megjelent. Azóta folyamatosan nőtt, és most nagy szerepet játszik a C ++ szabványosításában. Bár a Boost közösség és a szabványosítási bizottság között nincs formális kapcsolat, a fejlesztők egy része mindenkorban aktív. A 2011-ben jóváhagyott C ++ szabvány jelenlegi verziója olyan könyvtákat tartalmaz, amelyek gyökerei a Boost közösségen találhatóak.

A Boost könyvtárak jó választást jelentenek a C ++ projektek termelékenységének növelésére, ha a követelmények meghaladják a szabványos könyvtárban rendelkezésre álló adatokat. Mivel a Boost könyvtárak gyorsabban fejlődnek, mint a normál könyvtár, korábban hozzáérhettek az új fejlesztésekhez, és nem kell várnia, amíg ezeket a fejlesztések hozzáadják a szabványos könyvtár új verziójához. Így a Boost könyvtárnak köszönhetően élvezheti a C ++ gyorsabb fejlődését.

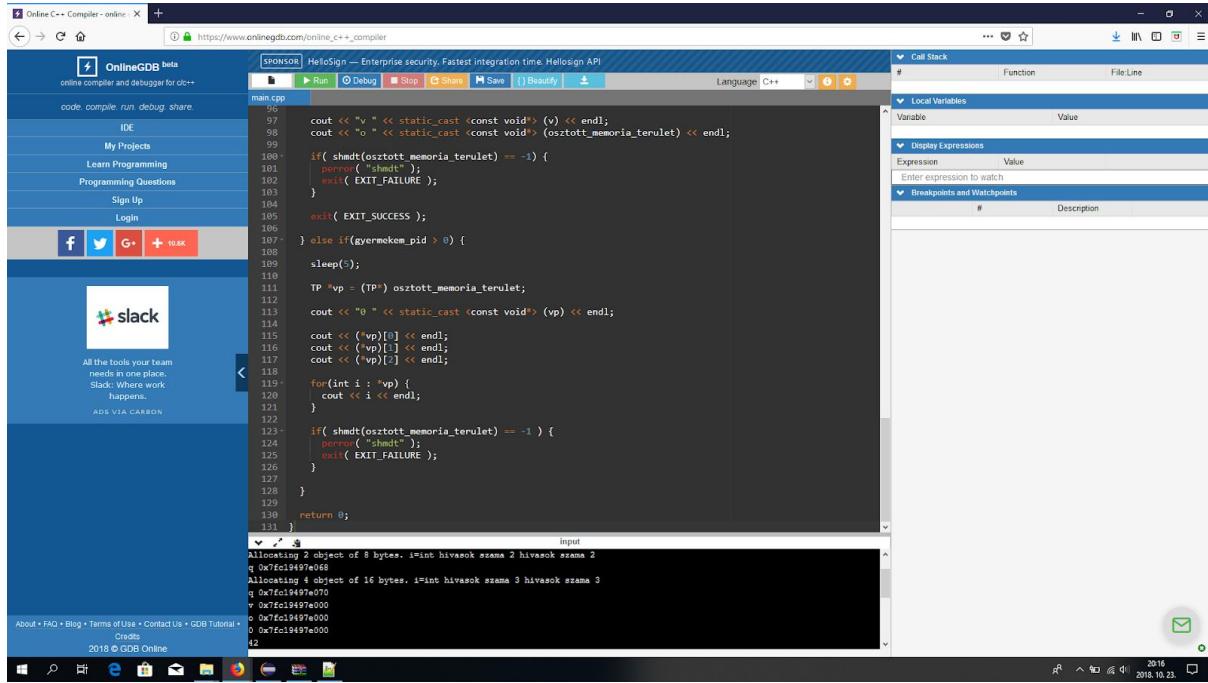
A Boost könyvtárak kiváló hírnevének köszönhetően, jól ismerik őket a mérnökök. A Boost könyvtárakban a fejlesztők általában ismerik a legújabb C ++ fejlesztéseket, és képesek megírni és megérteni a modern C ++ kódot.

Hogy ezeket a könyvtákat szálbiztosá tegyék, létrehozták a Boost.Thedset. Sok C++ szakértő szolgáltatott forrást a Boost.Threads megtervezéséhez. A felület az alapoktól lett megirva és nemcsak egyszerűen egy másik C szálas APIkat magába foglaló könyvtárátirat. A C++ számos szolgáltatása (úgymint a konstruktörök/ destruktörök, függvényobjektumok és sablonok) teljes mértékben felhasználódtak, így még rugalmasabbá téve a felületet.

6.hét

C++11CustomAllocator:

A C++ nyelvben allocatorok segítségével tudjuk szabályozni, hogy a programunk hogyan foglalja le a memoriában a helyet. Erre a c++ magától is képes, de hatékonyabbá tudjuk tenni programjainkat, ha magunk írunk allocator-t.



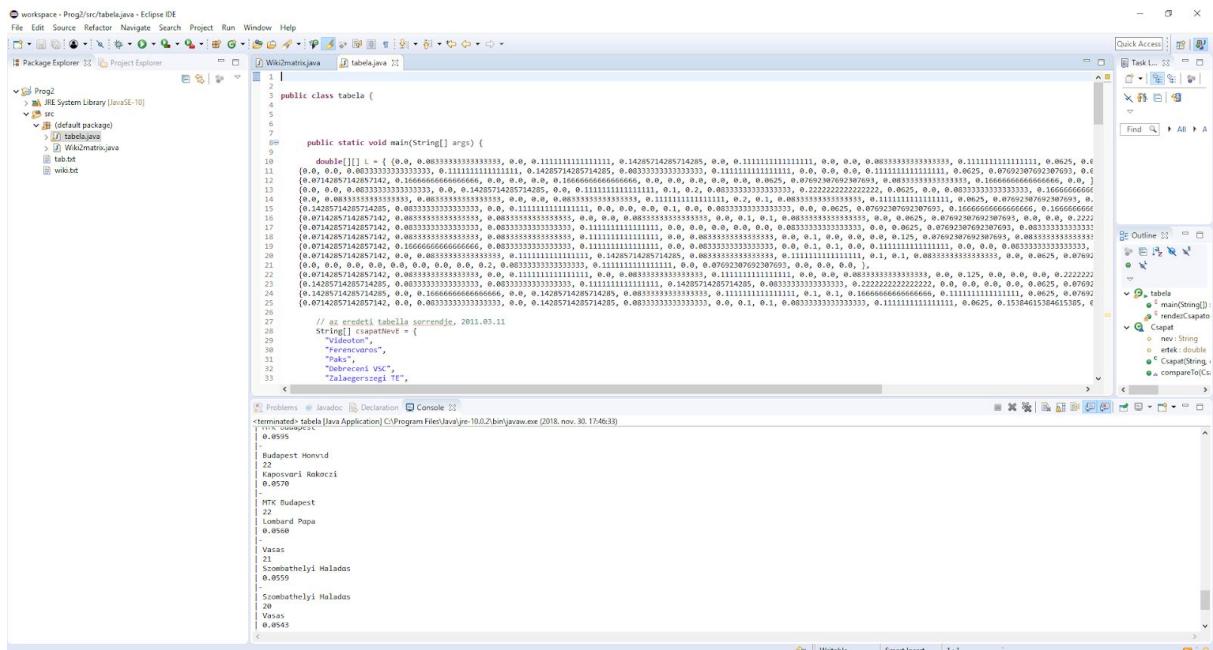
Alternativ Tabella rendezés:

A java comarable interface segítségével az osztály egyetlen adata alapján is képesek vagyunk

rendezni. Az interface a java.lang package-ben található, és egyetlen metódusa a compareTo(Obj o).

Miután megadtuk a Comparable interface-t implementáló Csatat osztályunkat, és azt, hogy hogy

viselkedjen a compareTo metódus, tudjuk rendezni tudjuk a rendezni kívánt osztályt.



The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows a project named "Prog2" containing a package "src" with files "Table.java" and "Wiki2matrix.java".
- Code Editor:** Displays the content of "Wiki2matrix.java". The code implements a search algorithm for a 3x3 matrix. It defines a matrix `kereszt` and a pointer matrix `pontotSzerez`. The algorithm iterates through the matrix to find specific values.
- Console:** Shows the output of the application, which lists names and their corresponding values.

```

1 public class Wiki2matrix {
2     public static void main(String[] args) {
3         int[][] kerest = {
4             {0, 0, 0, 1, 0, 3, 1, 3, 2, 0, 0, 0, 2, 2, 3},
5             {0, 0, 0, 1, 0, 3, 1, 3, 2, 0, 0, 0, 2, 2, 3},
6             {1, 1, 0, 0, 3, 1, 3, 0, 0, 0, 3, 1, 1, 0, 2, 0, 0},
7             {0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 2, 1, 1},
8             {1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 2, 1, 1},
9             {1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 2, 1, 1},
10            {1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 2, 1, 1},
11            {1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 2, 1, 1},
12            {1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 2, 1, 1},
13            {1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 2, 1, 1},
14            {1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 2, 1, 1},
15            {1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 2, 1, 1},
16            {1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 2, 1, 1},
17            {1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 2, 1, 1},
18            {1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 2, 1, 1},
19            {1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 2, 1, 1},
20            {1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 2, 1, 1},
21            {1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 2, 1, 1},
22            {1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 2, 1, 1},
23            {1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 2, 1, 1},
24            {1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 2, 1, 1},
25        };
26
27        int[][] pontotSzerez = new int[kereszt.length][kereszt.length];
28
29        for (int i = 0; i < pontotSzerez.length; i++) {
30            for (int j = 0; j < pontotSzerez[i].length; j++) {
31                pontotSzerez[i][j] = 0;
32            }
33        }
34    }
35
36    int[][][] pontotSzerez = new int[kereszt.length][kereszt.length];
37
38    for (int i = 0; i < pontotSzerez.length; i++) {
39        for (int j = 0; j < pontotSzerez[i].length; j++) {
40            pontotSzerez[i][j] = 0;
41        }
42    }
43
44    int[][] pontotSzerez = new int[kereszt.length][kereszt.length];
45
46    for (int i = 0; i < pontotSzerez.length; i++) {
47        for (int j = 0; j < pontotSzerez[i].length; j++) {
48            pontotSzerez[i][j] = 0;
49        }
50    }
51
52    for (int i = mymap.begin(); i != mymap.end(); i++) {
53        if (it->second) {
54            pair<string, int> p{it->first, it->second};
55            unsorted.push_back(p);
56        }
57    }
58
59    vector<pair<string, int>> sorted = sort_map(mymap);
60
61    for (int i = 0; i < sorted.size(); i++) {
62        cout << "sorted: " << sorted[i].first << ":" << sorted[i].second << "\t unsorted: ";
63    }
64
65    return 0;
66 }

```

STL map érték szerint rendezése:

A std :: map egy rendezett asszociatív tároló, amely egyedi kulcsokkal rendelkező kulcsértékpárokat tartalmaz.

The screenshot shows the Geany IDE and a terminal window. The code in the editor is as follows:

```

17:37:27: This is Geany 1.33.
17:37:27: New file "untitled" opened.
17:37:44: File /home/vampire/Desktop/prog2/StlMap.cpp saved.
17:37:46: File /home/vampire/Desktop/prog2/StlMap.cpp saved.

```

The terminal window shows the execution of the program and its output:

```

vampire@wolf:~/Desktop/prog2$ g++ StlMap.cpp -o stl
vampire@wolf:~/Desktop/prog2$ ./stl
sorted: NVN:961 unsorted: CAV:11587
sorted: VYK:3776 unsorted: LCL:113668
sorted: RAA:5905 unsorted: EUP:8559
sorted: RAA:5905 unsorted: LBR:11812
sorted: YIH:5788 unsorted: VNN:961
sorted: EUF:8550 unsorted: PMR:113430
sorted: CAY:11587 unsorted: RAA:5905
sorted: LBR:11812 unsorted: VYK:3776
sorted: PMR:13430 unsorted: KWN:4415
sorted: CCA:13668 unsorted: VIH:5788

```

Összefoglalás

A C++ nyelv Szabványos Sablonkönyvtára (Standard Template Library -STL) osztály- és függvény sablonokat tartalmaz, amelyekkel elterjedt adatstruktúrákat

(vektor, sor, lista, halmaz, szótár stb.) és algoritmusokat (rendezés, keresés, összefésülés stb.) építhetünk be a programunkba. A sablonos megoldás lehetővé teszi, hogy az adott néven szereplő osztályokat és függvényeket (majdnem) minden típushoz felhasználhatjuk, a program igényeinek megfelelően. Az STL alapvetően három csoportra épül, a konténerekre (tárolókra), az algoritmusokra és az iterátorokra (bejárókra). Egyszerűen megfogalmazva az algoritmusokat a konténerekben tárolt adatokon hajtjuk végre az iterátorok felhasználásával.

Az STL (azaz Standard Template Library) egyik fontos részét képezik a tárolók, azok az adatszerkezetek, amelyek különféle tárolási stratégiákat implementálva hatékonyan, biztonságosan, kivételbiztosan és típushelyesen képesek tárolni az adatokat, ellentétben a C-stílusú, beépített tömbökkel és kézzel írt láncolt adatszerkezetekkel. Tipikusan kiegynélyezett bináris keresőfával implementálják (a legtöbb esetben Vörös-fekete fával), de a szabvány ezt nem köti ki. A map egy variánsa a multimap, amelyben egy kulcs többször is szerepelhet, így értelmetlenné válik a [] operátor, viszont megjelenik egy kulcs számossága.

A map és a multimap konténerekben adatpárok tárolódnak, melyek első (first) tagja maga a kulcs, míg a második (second) tagja az adat. A szokásos konténerműveleteken túlmenően a minden változtatás nélkül alkalmazhatók a halmazoknál bemutatott keresési tagfüggvények (find(), count(), lower_bound(), stb.).

A map és az unordered_map konténerek esetén szintén használhatjuk az at() függvényt és az indexelés operátorát. Mindkét esetben argumentumként a kulcsot kell megadnunk, és a vele párosan tárolt adat referenciáját kapjuk vissza. Ha kulcs nem létezik, az at(kulcs) hívás out_of_range kivételt hoz létre, míg az indexelés esetén egy új elemmel bővül az asszociatív tömb, melynek adat része az AdatTípus {} értékkal inicializálódik.

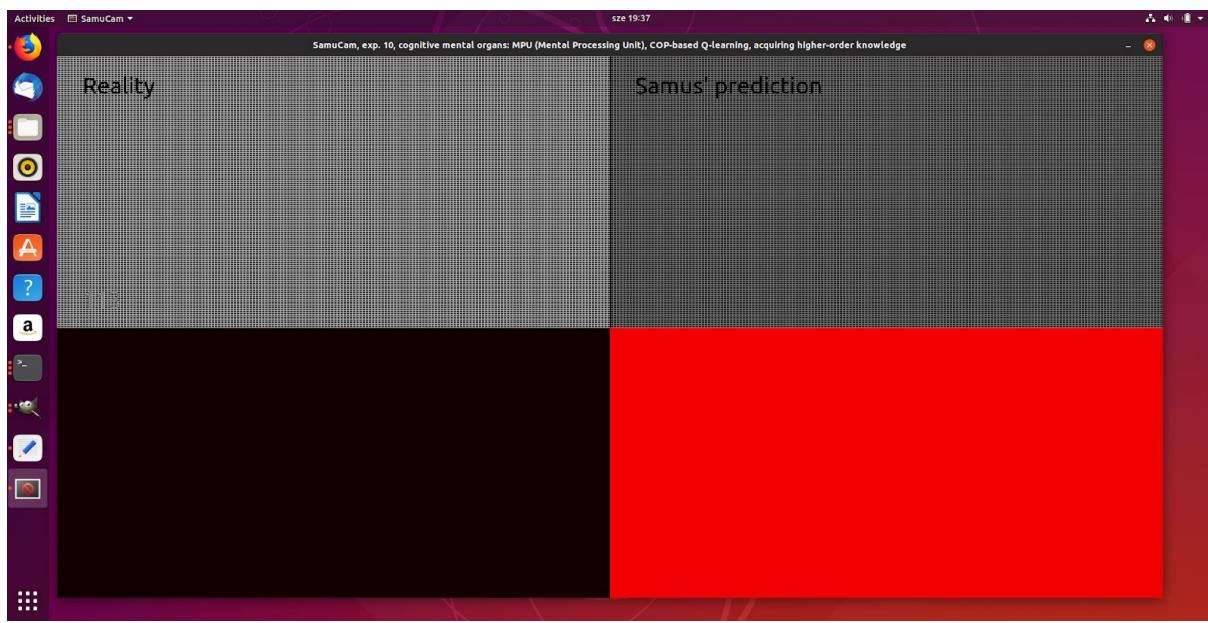
A egyedi kulcsot tartalmazó asszociatív tárolók (set, map, stb.) esetén a beszúrás csak akkor lesz sikeres, ha a kulcs még nem szerepel a konténerben. Az ilyen tárolók használatakor a ret visszaadott érték egy pair<iterator,bool> pár, ahol az iterátor a beszúrt elemre hivatkozik, vagy arra, ami megakadályozta a beillesztést, a logikai érték pedig true-val jelzi, ha megtörtént a művelet. A több, azonos kulcsot is megengedő asszociatív tárolóknál (multiset, multimap, stb.) a ret egy iterátor, ami kijelöli a beszúrt elemet. A map esetén, ha az indexelés során megadott kulcs nem létezik, akkor az adott kulcsú elem minden létrejön, még akkor is, ha lekérdezés volt a célunk. Ekkor az adat a típusának megfelelő alapértékkel inicializálódik.

7.hét

Samucam:

Ehez a programhoz Opencv3.1.0-t, és Qt5.11.2-t használtam.

A SamuCam elindításához a .pro file-t módosítottam: hozzáadtam a LIBS-hez -lopencv_core-t és sok egyebet.



BrainB:

Ehez a programhoz Opencv-t és Qt-t használtam.

A BrainB elindításához a .pro file-t módosítottam: hozzáadtam a LIBS-hez -lopencv_core-t pl.

A Qt signalok arra használatosak, hogy jelezzék, ha az objektum megváltozott.

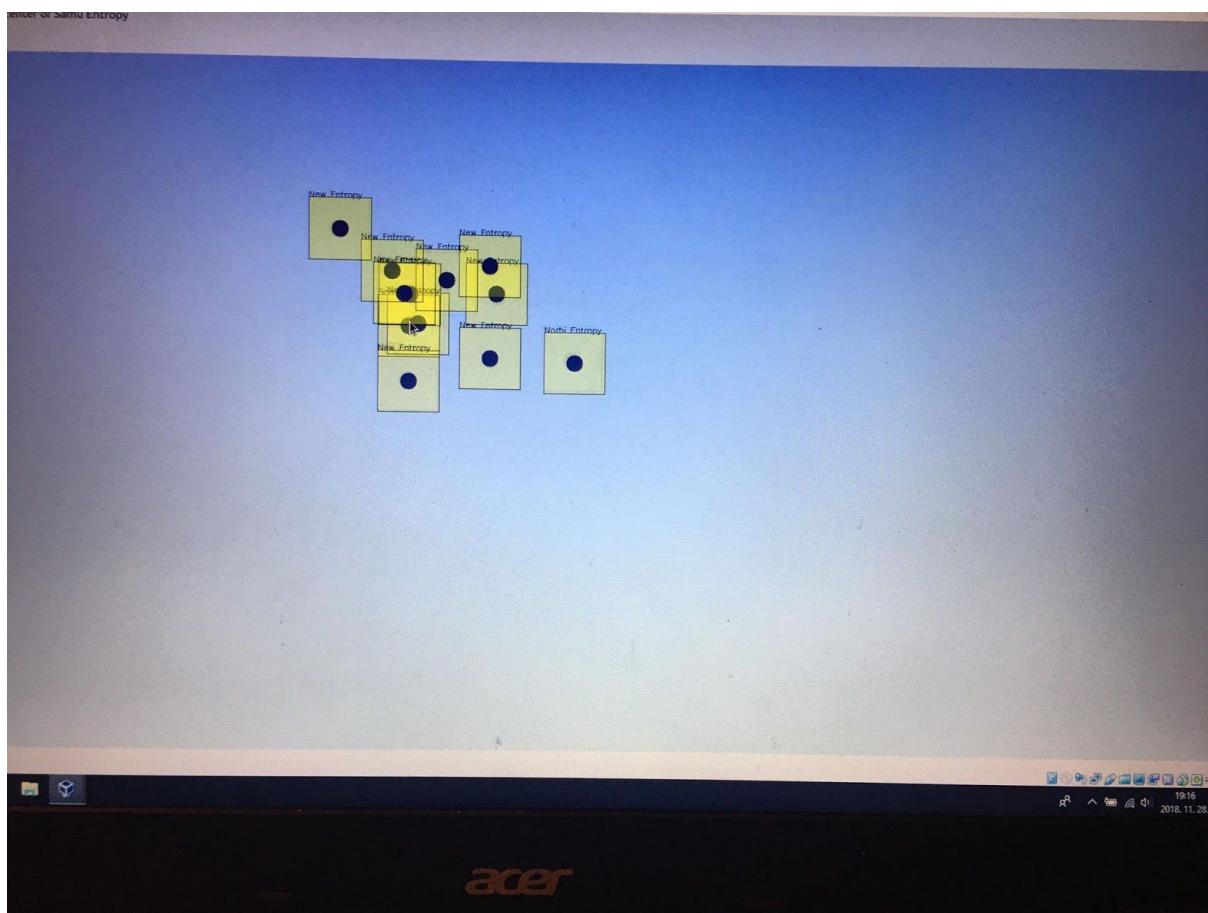
A SLOT-ok egyszerű metódusok amiket meg tudunk hívni a main-en belül is.

A Qt slot signal a BrainB esetén a connect-tel a brainBThread-ben lévő signal hatására végrehatja az

objektumon a slot utasítást.

(pl. a brainBThread-ben lévő heroesChanged signal hatására ez az objektum végrehajtja a slotnak

kikiáltott updateHeroes funkciót)



ASIO:

Source code:

https://github.com/martva383/prog2/tree/master/7_gyak/Elsz%C3%A1mol%C3%A1s/ASIO

Future:

Az ActivityEditor egy javaban íródott program, aminek a futtatásához javafx kell.
A program feladata a FUTURE szerekezetének feltérképezése, vizuális megjelenítése,
illetve szerkesztése.

Az volt a feladat, hogy javítsunk a ActivityEditoron, szóval hozzáadtam egy új menüpontot,
amit

jobbgombbal való kattintással hozhatunk elő.

Az új menüpont: „Elérési út”, amire hogyha rákattintunk, akkor kiírja a terminálra az utat.

Source code:

https://github.com/martva383/prog2/tree/master/7_gyak/Future

8.hét

AOP:

Az aspektus orientált programozás egy programozási paradigmája, aminek célja hogy a programunkat

kisebb részegységekre bontsuk, ezzel hatékonyabbá, könnyebben áttekinthetővé téve azt.

The screenshot shows the Eclipse IDE interface with several windows open:

- Package Explorer**: Shows the project structure with files like `LWVInfo.java`, `Compost.java`, and `Manjara.java`.
- Java Editor**: Displays the `LWVInfo.java` file. The cursor is at line 48, where the code `fa.egyesgyermek(new Compost('1'));` is being typed. A code completion dropdown is open, listing suggestions such as `fa.egyesgyermek(new Compost('1'))`, `fa.egyesgyermek(new Compost('2'))`, `fa.egyesgyermek(new Compost('3'))`, `fa.egyesgyermek(new Compost('4'))`, `fa.egyesgyermek(new Compost('5'))`, `fa.egyesgyermek(new Compost('6'))`, `fa.egyesgyermek(new Compost('7'))`, `fa.egyesgyermek(new Compost('8'))`, and `fa.egyesgyermek(new Compost('9'))`.
- Output Window**: Shows the command `java -jar smapletz.jar` running and outputting the message "Smapletz started".
- File Explorer**: Shows the file system with folders like `src`, `bin`, and `output.txt`.
- Search View**: Shows search results for "smapletz" across various files.
- Help View**: Shows help documentation for the `System.out.println` method.
- Quick Access**: Shows recently used files and projects.

Összefoglaló:

A kivételezés egy programozási mechanizmus, melynek célja a program futását szándékosan vagy nem szándékolt módon megszakító esemény (hiba) vagy utasítás kezelése. Az eseményt magát kivételek hívjuk.

A programok alapvetően szekvenciális – így a folyamatában jól nyomon követhető – működésében sok esetben következhetnek be olyan események, amelyekre a programozó nem készül, vagy nem készülhet fel (esetleg nem is akar felkészülni). Az ilyen hibajelenségek az alapvető utasításkészlettel régen nem voltak kezelhetőek (elkaphatók). A normál hibakezelés általában körülönműves

Voltak kezelőiök (elkaphatók). A normál hibakezelés általában korlátozott, sok változó beiktatását és figyelését, valamint rengeteg feltételes utasítást igényel. A hagyományos hibakezelés esetén az utasítások végrehajtása jól követhető

nyomvonalakon halad végig, jó esetben ugrások (például GOTO) nélkül. Azonban ha az adott problémát nem kezeljük le, akkor a program defektje kárt okozhat a rendszerben. (Például az alkalmazás leáll, vagy rossz esetben adatsérülés lesz az)

eredmény.)

A megszakítás-elvű kivételkezeléses rendszerben a hibákat könnyebb elfogni, valamint a nyelvek általában egy beépített eljárással kezelik a hibákat, ha mi nem tettük volna meg. Ezért a kivételek használata és az ezekkel való munka mindenkorban egy magasabb szintű, felügyelt hibakezelést tesz lehetővé.

Működése:

Az operációs rendszer által kezelt szoftveres megszakítások felhasználása révén a különböző programnyelvek lehetőséget nyújtanak olyan blokkok beiktatására, amelyek a hiba események bekövetkeztekor reagálni tudnak. A hiba eseményt (például rossz helyről olvasunk a memoriában) a processzor (illetve az operációs rendszer) érzékeli, majd szoftveres megszakítással él. A megszakításkezelő a megfelelő alkalmazás hibakezelőjére adja a vezérlést. Ez az eljárás a veremben, vagy más adatközegben felkutatja a legközelebbi hibakezelő blokkot, és oda ugrik. Ha nincs ilyen, akkor az alapértelmezett hibakezelőt használja.

A nyelvi megjelenésük roppant egyszerű. Általában 3 szakaszban definiálhatóak, és mindenhez megtagadott szerepe van, melyek közül kettőt kell párosítva használni: az elsőt, illetve a második és harmadik elem közül az elérni kívánt hatás szerint választottat.

A „try” kódblokk:

Ebbe a szakaszba helyezzük el azokat az utasításokat, amelyekben hibát "várunk". Tulajdonképpen kipróbáljuk, hogy történik-e hiba. Ha nem, akkor a blokkban szereplő utasítások lefutnak, majd a következő utasításhalmazhoz kerülünk.

A „catch” kódblokk:

Ezen blokkokba kerül a végrehajtás, ha valamilyen hiba fordult elő. Általában a programnyelvek támogatják a kivételeket szerinti szelekciót, vagy a kivételekből információk kinyerését. E kódszakaszba tehetünk minden kezelőt vagy reagáló elemet, amit a hiba kiküszöbölése vagy észlelése miatt be akarunk venni.

Végül a „finally”(lezáró) szakasz:

E kódblokk tartalmaz minden olyan kritikus utasítást, aminek mindenkorban le kell futni – vagyis még kivétel bekövetkezte esetén is. Ide helyezendő minden erőforrás-kezelő rutin – általában az összes felszabadító és elengedő szekvenciát ide érdemes halmozni, ami kapcsolatban lehet a védendő kódszakasszal.

Port Scan:

scan:

Ennél a kivételkezelésnél 2 blokkot definiáltunk. Ez a két blokk a „try” illetve a „catch” kódblokkok.

Az első blokk az úgymond „próbára tett kódblokk” itt helyezzük el azokat az utasításokat, amelyeket szeretnénk tesztelni, hogy hibát adnak-e vissza. Ebben a példában a ezek a következő utasítások:

```
java.net.Socket socket = new java.net.Socket(args[0], i);
```

```
System.out.println(i + „figyeli”);
socket.close();
```

Ha ezek közül bármelyik is hibát dob ki (ebben a példában valószínűleg az első sornál fog hibát találni), akkor teljesülnek azok az utasítások, amelyek a „catch” ágban szerepelnek. Ebben az esetben kiírja a program, hogy nem figyeli az adott portot.

Lényegében a program annyit csinál, hogy végigmegy 0tól 1023ig a portokon, és megpróbál mindegyikkel TCP kapcsolatot létesíteni, ha ez sikerül, akkor kiírja, hogy az adott portot figyeli, de ha bármilyen utasítás ami a „try” blokkban szerepel hibát ad vissza, akkor a program kiírja, hogy nem figyeli az adott portot.

The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** workspace - Prog2/src/Port.java - Eclipse IDE
- Menu Bar:** File Edit Source Refactor Navigate Search Project Run Window Help
- Toolbar:** Standard toolbar icons
- Project Explorer:** Shows a Java project named "Prog2" with a single source file "Port.java" under the "src" folder.
- Code Editor:** Displays the following Java code:

```
1 public class Port {
2     public static void main(String[] args) {
3         for(int i = 0; i < 10000; i++) {
4             try {
5                 java.net.Socket socket = new java.net.Socket(args[0], i);
6                 System.out.println(i + " figyeli");
7                 socket.close();
8             } catch (Exception e){
9                 System.out.println(i + " nem figyeli");
10            }
11        }
12    }
13 }
```

- Console View:** Shows the output of the program execution:
0 nem figyeli
1 nem figyeli
2 nem figyeli
3 nem figyeli
4 nem figyeli
5 nem figyeli
6 nem figyeli
7 nem figyeli
8 nem figyeli
9 nem figyeli
10 nem figyeli
11 nem figyeli

Junit teszt:

Source code:

https://github.com/martva383/prog2/tree/master/8_gyak/Junit%20teszt

```

eclipse-workspace - sampletest/src/sampletest/Main.java - Eclipse IDE
File Edit Source Refactor Search Project Run Window Help
Package Explor... Main.java C:\sampletest\Main.java
src sampletest
  > RFS System Library [Java]
  > sampletest
    > sampletest
      > C:\sampletest\Main.java
      > LZWBit.java
      > EszTest.java
      > Main.java
    > module-info.java
  > testproject
Quick Access
Output.txt - Jegyzetömb
Fájl Szerkesztés Formátum Nézet Súgó
---->-1(1)
-->/(9)
-----1(3)
-----1(2)
-----0(1)
-----0(2)
-----0(4)
-----0(3)
melysség = 4
átlag = 2.6666666666666665
szorás = 1.5275252316519465
Keresés sampletest
1 KB
1 KB
1 KB
1 KB

```

9.hét

Összefoglaló:

A számítógépi hardver teljesítmények növekedése ösztönzően hat a különböző célú szoftverek kidolgozására, többek között a gépi tanulási módszerek fejlesztésére. A mesterséges neurális hálók kutatása több évtizedes múltra tekint vissza, kiemelkedő népszerűségére pedig a képfelismerésnek és a beszédfeldolgozásnak köszönhetően tett szert. A többrétegű neurális hálók programozására a Google kutatói 2015-ben kifejlesztették a TensorFlow rendszert.

A TensorFlow egy szoftverkönyvtár, gépi tanulási algoritmusok leírására és végrehajtására. A TensorFlow roppant felxibilis, nagyon széles körű algoritmusok megvalósítására alkalmas beleértve a deep neural network alkalmazásait, például a beszédfelismerésben, a számítógépi látásban, megjelenítésben, a robotikában, az információ kinyerésében, a számítógépek elleni támadások felderítésében, és az agyvitatásban. A TensorFlow rendszerben kifejlesztett számítások kevés változtatással végrehajthatók a nagyon eltérő hardvereken is.

A TensorFlow rendszer kifejlesztésének előzményei a GoogleBrain projektre nyúlnak vissza, amely 2011-ben indult azzal a céllal, hogy mind kutatási, mind Google alkalmazások számára sokrétegű neurális hálók használata váljon lehetővé nagyon széles körben. A projekt első korai részeként a DistBelief rendszer készült el széles körű kutatásokra.

A GoogleBrain kutatócsoporttal szorosan együttműködve több mint 50 kutatócsoport alkotott mély neurális háló modelleket a DistBelief rendszer használatával, többek között a Google keresőrendszer, Google Fotók, Google térkép, Google Fordító, YouTube számára. A DistBelief rendszerrel szerzett tapasztalatok és

neurális hálókat használó rendszerek jobb megismerése alapján került kifejlesztésre a TensorFlow rendszer, amely képes nagy tömegű adat alapján gépi tanulási modellek létrehozására. A fejlesztők ügyeltek arra, hogy a TensorFlow rendszer egyszerű és flexibilis legyen a kutatás számra, másrészt a valós alkalmazásokkal szemben támásztott követelményeknek megfelelő robosztus és hatékony legyen.

A TensorFlow számítást egy irányított gráf írja le. Adatáramlás a gráf élei mentén történik. A TensorFlow gráfban mindegyik csúcs egy műveletet reprezentálhat és minden csúcsnak lehet nulla vagy több inputja, ugyanúgy nulla vagy több outputja. A gráf normál élei mentén áramló értékek tenzorok, tetszőleges dimenziójú vektorok. Egy-egy elem típusát a gráf konstruálásakor specifikálják. Lehetnek a gráfban speciális élek is, amelyek mentén nem történik adatáramlás, hanem kontrol célok szolgálnak.

Egy TensorFlow műveletnek neve van és egy absztrakt számítást reprezentál.

A műveletnek lehetnek attribútumai, amelyeket a gráf konstruálásakor kell megadni. A TensorFlow kernel egy művelet egy olyan konkrét implementációja, amely egy adott eszköztípuson futtatható. A kliens programok a TensorFlow rendszerrel session létrehozásával kerülnek kapcsolatba.

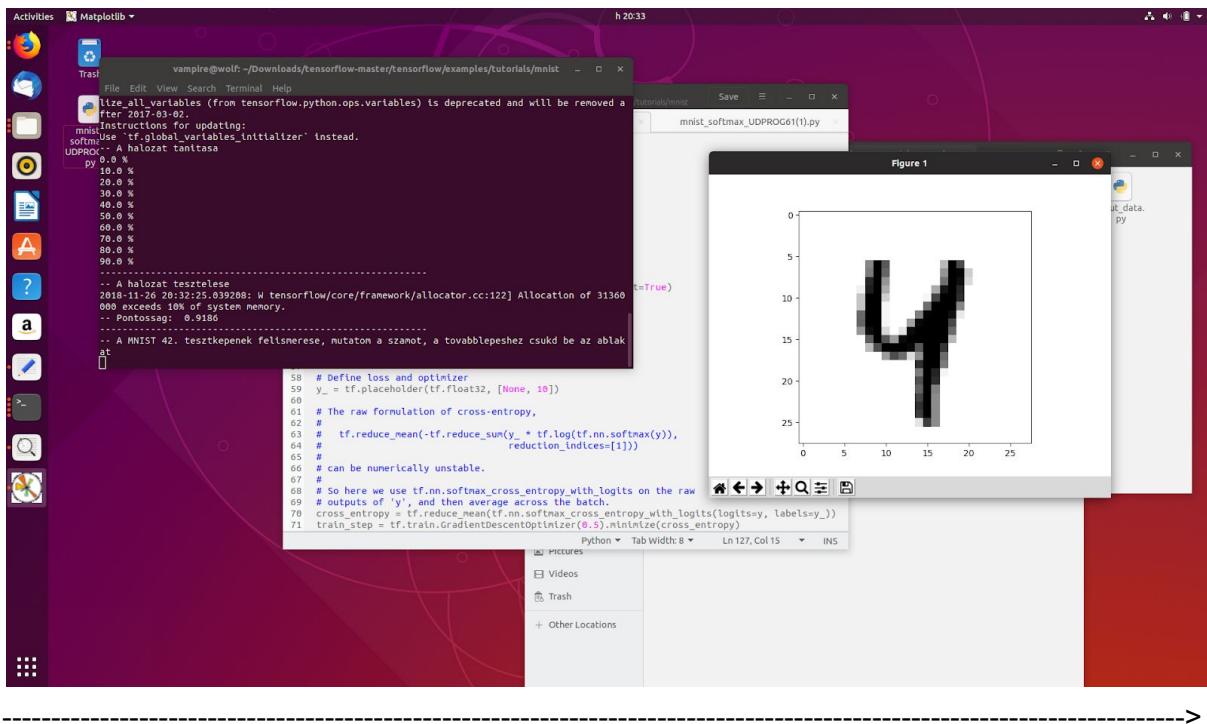
A TensorFlow rendszer tartalmaz optimalizálást a számítási gráfban előforduló redundancia kiszűrésére, a memóriahasználatra, az adatbevitelre, a kernel implementáció kiválasztására.

Mnist:

A feladat az, hogy a telepítsük a TensorFlow-t, majd egy bizonyos elérési útnál létre kellett hozni az „mnist_softmax_UDPROG61.py” nevű fájlt, majd a feladat leírásában szereplő linkek lévő kódokat bele kellett másolni.

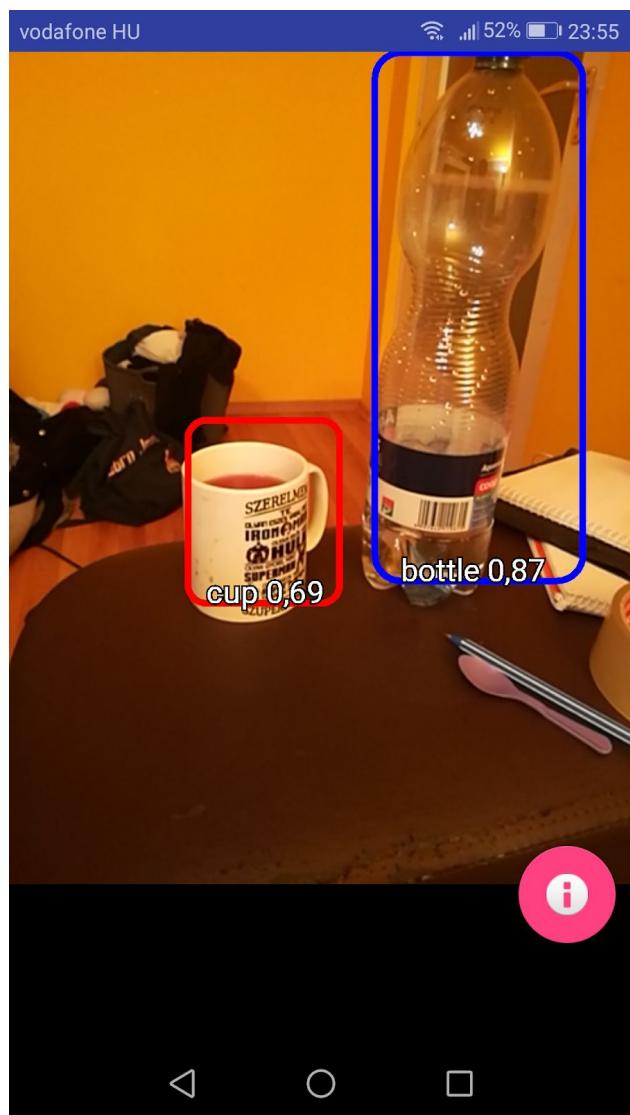
A program a TensorFlow képfelismerő algoritmusán alapszik, pontosabban a program felismeri a számjegyeket 0-tól 9ig.

A program egy 28x28 pixelből álló képet vár ami fekete háttérű, és fehérrrel van ráírva az általunk vizsgálni kívánt számjegy.



Andorid Telefonra TF:

A második program a TF Detect nevet kaptam, mely annyival tud többet előző társánál, hogy nem csak kiírja, hogy milyen tárgyat lát milyen valószínűsséggel, hanem azt a részt is bekeretezi, ahol a tárgyat érzékeli a képen.



* A feladatok megoldásában: Varga Sándor Mátyás, Keserű Kristóf, Péter Erdős Udprog közössége, kereszttapukám segített.