

# Univerzális programozás

---

**Írd meg a saját programozás tankönyvedet!**

Ed. BHAX, DEBRECEN,  
2019. május 09, v. 1.0.0

Copyright © 2019 Dr. Bátfai Norbert

Copyright © 2019 Nemes Bence

Copyright (C) 2019, Norbert Bátfai Ph.D., batfai.norbert@inf.unideb.hu, nbatfai@gmail.com

Copyright (C) 2019, Nemes Bence, kemence0528@gmail.com

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

<https://www.gnu.org/licenses/fdl.html>

Engedélyt adunk Önnek a jelen dokumentum sokszorosítására, terjesztésére és/vagy módosítására a Free Software Foundation által kiadott GNU FDL 1.3-as, vagy bármely azt követő verziójának feltételei alapján. Nincs Nem Változtatható szakasz, nincs Címlapszöveg, nincs Hátlapszöveg.

<http://gnu.hu/fdl.html>

**COLLABORATORS**

	<i>TITLE :</i> Univerzális programozás		
<i>ACTION</i>	<i>NAME</i>	<i>DATE</i>	<i>SIGNATURE</i>
WRITTEN BY	Bátfai, Norbert ÁCs Nemes, Bence	2020. december 9.	

**REVISION HISTORY**

NUMBER	DATE	DESCRIPTION	NAME

# Ajánlás

„To me, you understand something only if you can program it. (You, not someone else!) Otherwise you don't really understand it, you only think you understand it.”

—Gregory Chaitin, *META MATH! The Quest for Omega*, [[METAMATH](#)]

# Tartalomjegyzék

<b>I. Bevezetés</b>	<b>1</b>
1. Vízió	2
1.1. Mi a programozás? . . . . .	2
1.2. Milyen doksikat olvassak el? . . . . .	2
1.3. Milyen filmeket nézzek meg? . . . . .	2
<b>II. Második felvonás</b>	<b>3</b>
2. Helló, Arroway!	5
2.1. OO szemlélet . . . . .	5
2.2. „Gagyi” . . . . .	6
2.3. Yoda . . . . .	7
3. Helló, Liskov!	9
3.1. Liskov helyettesítés sértése . . . . .	9
3.2. Szülő-gyerek . . . . .	10
3.3. Anti OO . . . . .	11
3.4. Ciklomatikus komplexitás . . . . .	12
4. Helló, Mandelbrot!	14
4.1. Reverse engineering UML osztálydiagram . . . . .	14
4.2. Forward engineering UML osztálydiagram . . . . .	15
4.3. Egy esettan . . . . .	20
4.4. BPMN . . . . .	31

---

<b>5. Helló, Chomsky!</b>	<b>32</b>
5.1. Encoding . . . . .	32
5.2. l334d1c45 . . . . .	36
5.3. Full screen . . . . .	38
<b>6. Helló, Stroustrup!</b>	<b>40</b>
6.1. JDK osztályok . . . . .	40
6.2. Másoló-mozgató szemantika . . . . .	42
6.3. Összefoglaló . . . . .	43
<b>7. Helló, Gödel!</b>	<b>45</b>
7.1. Gengszterek . . . . .	45
7.2. C++11 Custom Allocator . . . . .	46
7.3. STL map érték szerinti rendezése . . . . .	46
7.4. Alternatív Tabella rendezése . . . . .	48
<b>8. Helló, !</b>	<b>50</b>
8.1. OOCWC Boost ASIO hálózatkezelése . . . . .	50
8.2. SamuCam . . . . .	50
8.3. BrainB . . . . .	51
<b>9. Helló, Schwarzenegger!</b>	<b>53</b>
9.1. Port scan . . . . .	53
9.2. Android Játék . . . . .	54
<b>10. Helló, Calvin!</b>	<b>56</b>
10.1. MNIST . . . . .	56
10.2. CIFAR-10 . . . . .	58
10.3. Android telefonra a TF objektum detektálója . . . . .	60
<b>11. Helló, Berners-Lee!</b>	<b>63</b>
11.1. C++ és Java összehasonlítás . . . . .	63
11.2. Python . . . . .	63

---

<b>III. Irodalomjegyzék</b>	<b>65</b>
11.3. Általános . . . . .	66
11.4. C . . . . .	66
11.5. C++ . . . . .	66
11.6. Lisp . . . . .	66

# Ábrák jegyzéke

2.1. OO . . . . .	6
2.2. gagyi . . . . .	7
2.3. yoda . . . . .	8
2.4. yoda2 . . . . .	8
3.1. madarszulo . . . . .	10
3.2. antioo . . . . .	12
3.3. binfalizard . . . . .	13
4.1. reverseuml . . . . .	14
4.2. szorp . . . . .	31
5.1. mandelbrothalmaznagyító . . . . .	36
5.2. leet . . . . .	38
5.3. labprint . . . . .	39
5.4. labirintus . . . . .	39
6.1. JDK . . . . .	40
6.2. JDKC++command . . . . .	42
6.3. JDKC++result . . . . .	42
7.1. STL . . . . .	48
8.1. BrainB . . . . .	52
9.1. KapuSzkenner . . . . .	54
9.2. TicTacToe . . . . .	55
10.1. mnist . . . . .	58
10.2. cat . . . . .	59

---

10.3. airplane . . . . .	59
10.4. horse . . . . .	60
10.5. tfod1 . . . . .	61
10.6. tfod2 . . . . .	62

# Előszó

Amikor programozónak terveztem állni, ellenezték a környezetemben, mondván, hogy kell szövegszerkesztő meg táblázatkezelő, de az már van... nem lesz programozói munka.

Tévedtek. Hogy egy generáció múlva kell-e még tömegesen hús-vér programozó vagy olcsóbb lesz alkalmi igény szerint pár robot programozót a felhőből? A programozók dolgozók lesznek vagy papok? Ki tudhatná ma.

Minden esetre a programozás a teoretikus kultúra csúcsa. A GNU mozgalomban látom annak garanciáját, hogy ebben a szellemi kalandban a gyerekeim is részt vehessenek majd. Ezért programozunk.

## Hogyan forgasd

A könyv célja egy stabil programozási szemlélet kialakítása az olvasóban. Módszere, hogy hetekre bontva ad egy tematikus feladatcsokrot. minden feladathoz megadja a megoldás forráskódját és forrásokat feldolgozó videókat. Az olvasó feladata, hogy ezek tanulmányozása után maga adja meg a feladat megoldásának lényegi magyarázatát, avagy írja meg a könyvet.

Miért univerzális? Mert az olvasótól (kvázi az írótól) függ, hogy kinek szól a könyv. Alapértelmezésben gyerekeknek, mert velük készítem az iniciális változatot. Ám tervezem felhasználását az egyetemi programozás oktatásban is. Ahogy szélesedni tudna a felhasználók köre, akkor lehetne kiadása különböző korosztályú gyerekeknek, családoknak, szakköröknek, programozás kurzusoknak, felnőtt és továbbképzési műhelyeknek és sorolhatnánk...

## Milyen nyelven nyomjuk?

C (mutatók), C++ (másoló és mozgató szemantika) és Java (lebutított C++) nyelvekből kell egy jó alap, ezt kell kiegészíteni pár R (vektoros szemlélet), Python (gépi tanulás bevezető), Lisp és Prolog (hogy lássuk más is) példával.

## Hogyan nyomjuk?

Rántsd le a <https://gitlab.com/nbatfai/bhax> git repót, vagy méginkább forkolj belőle magadnak egy sajátot a GitLabon, ha már saját könyvön dolgozol!

Ha megvannak a könyv DocBook XML forrásai, akkor az alább látható **make** parancs ellenőrzi, hogy „jól formázottak” és „érvényesek-e” ezek az XML források, majd elkészíti a dblatex programmal a könyved pdf változatát, íme:

```
batfai@entropy:~$ cd glrepos/bhax/thematic_tutorials/bhax_textbook/
batfai@entropy:~/glrepos/bhax/thematic_tutorials/bhax_textbook$ make
rm -f bhax-textbook-fdl.pdf
xmllint --xinclude bhax-textbook-fdl.xml --output output.xml
xmllint --relaxng http://docbook.org/xml/5.0/rng/docbookxi.rng output.xml ←
    --noout
output.xml validates
rm -f output.xml
dblatex bhax-textbook-fdl.xml -p bhax-textbook.xls
Build the book set list...
Build the listings...
XSLT stylesheets DocBook - LaTeX 2e (0.3.10)
=====
Stripping NS from DocBook 5/NG document.
Processing stripped document.
Image 'dblatex' not found
Build bhax-textbook-fdl.pdf
'bhax-textbook-fdl.pdf' successfully built
```

Ha minden igaz, akkor most éppen ezt a legenerált `bhax-textbook-fdl.pdf` fájlt olvasod.



#### A DocBook XML 5.1 új neked?

Ez esetben forgasd a <https://tdg.docbook.org/tdg/5.1/> könyvet, a végén találod az informatikai szövegek jelölésére használható gazdag „API” elemenkénti bemutatását.

## I. rész

### Bevezetés

# 1. fejezet

## Vízió

### 1.1. Mi a programozás?

### 1.2. Milyen doksikat olvassak el?

- Olvasgasd a kézikönyv lapjait, kezd a **man man** parancs kiadásával. A C programozásban a 3-as szintű lapokat fogod nézegetni, például az első feladat kapcsán ezt a **man 3 sleep** lapot
- [KERNIGHANRITCHIE]
- [BMECPP]
- Az igazi kockák persze csemegéznek a C nyelvi szabvány [ISO/IEC 9899:2017](#) kódcsipeteiből is.

### 1.3. Milyen filmeket nézzek meg?

- 21 - Las Vegas ostroma, <https://www.imdb.com/title/tt0478087/>, benne a **Monty Hall probléma** bemutatása.

## **II. rész**

### **Második felvonás**

**Bátf41 Haxor Stream**

A feladatokkal kapcsolatos élő adásokat sugároz a <https://www.twitch.tv/nbatfai> csatorna, melynek permanens archívuma a <https://www.youtube.com/c/nbatfai> csatornán található.

---

## 2. fejezet

# Helló, Arroway!

### 2.1. OO szemlélet

A módosított polártranszformációs normális generátor beprogramozása Java nyelven. Mutassunk rá, hogy a mi természetes saját megoldásunk (az algoritmus egyszerre két normálist állít elő, kell egy példánytag, amely a nem visszaadottat tárolja és egy logikai tag, hogy van-e tárolt vagy futtatni kell az algot.) és az OpenJDK, Oracle JDK-ban a Sun által adott OO szervezés ua.! <https://arato.inf.unideb.hu/baffai.norbert/UDPROG> (16-22 fólia) Ugyanezt írjuk meg C++ nyelven is! (lásd még UDPROG repó: source/labor/polargen)

Az OO szemlélet egy programozási paradigma, ami az objektumok fogalmán alapul. Megmutatja azt, hogy az OO szemlélettel létrehozhatok saját magam is dolgokat. Jelentősen megkönnyíti a programozást, mivel könnyedén felhasználhatunk akár más programozók által készített osztályokat, azok minden tulajdonságával, viszont néhány feladatot túlbonyolít, és kezdő programozóknak nehezebb elsajátítani a módszereket. A megoldásban látszik, hogy a feladat valóban megoldható egy logikai (boolean nincsTarolt) és egy tört (double tarolt) változó segítségével. Első alkalommal a program biztosan belép az if ágba, hiszen deklarációjánál igazra állítottuk a boolean változónkat és az if csak annak igazságát vizsgálja. Ha a program belép az if ágba Math.random() függvény meghívásával 4 double változó segítségével és matematika műveletekkel létrehozunk egy tárolt tagot és a boolean változónkat ellenkező értékre állítjuk (jelen esetben hamis) Az else ág visszaadja az előzőleg eltárolt adatot.

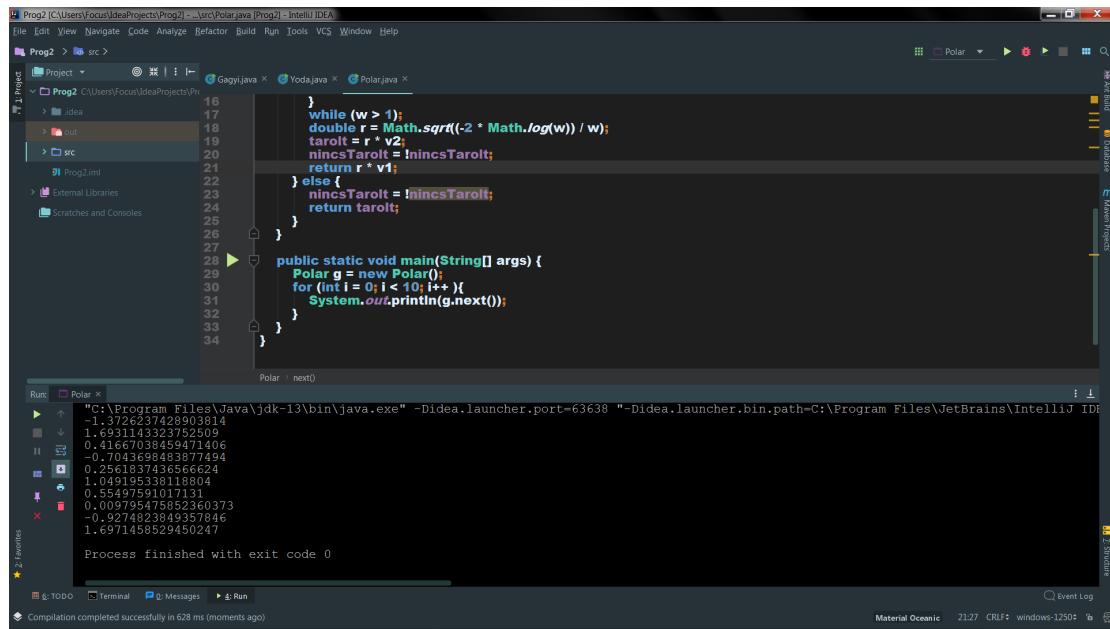
```
public class Polar {
    boolean nincsTarolt = true;
    double tarolt;
    public Polar() {
        nincsTarolt = true;
    }
    public double next() {
        if (nincsTarolt) {
            double u1, u2, v1, v2, w;
            do{
                u1 = Math.random();
                u2 = Math.random();
                v1 = 2 * u1 - 1;
                v2 = 2 * u2 - 1;
                if (v1 * v1 + v2 * v2 <= 1) {
                    nincsTarolt = false;
                    tarolt = Math.sqrt(-2 * Math.log(v1 * v1 + v2 * v2)) * Math.atan2(v2, v1);
                }
            } while (nincsTarolt);
        }
        return tarolt;
    }
}
```

```

        w = v1 * v1 + v2 * v2;
    }
    while (w > 1);
    double r = Math.sqrt((-2 * Math.log(w)) / w);
    tarolt = r * v2;
    nincsTarolt = !nincsTarolt;
    return r * v1;
} else {
    nincsTarolt = !nincsTarolt;
    return tarolt;
}
}

public static void main(String[] args) {
    Polar g = new Polar();
    for (int i = 0; i < 10; i++) {
        System.out.println(g.next());
    }
}
}

```



2.1. ábra. OO

## 2.2. „Gagyi”

Az ismert formális „while ( $x \leq t \&& x \geq t \&& t \neq x$ );” tesztkérdéstípusra adj a szokásosnál (miszerint  $x$ ,  $t$  az egyik esetben az objektum által hordozott érték, a másikban meg az objektum referenciaja) „mélyebb”

választ, írj Java példaprogramot mely egyszer végtelen ciklus, más x, t értékekkel meg nem! A példát építsd a JDK Integer.java forrására<sup>3</sup>, hogy a 128-nál inkluzív objektum példányokat poolozza!

A feladat példaként és bizonyítékként szolgál arra, hogy a -128 és 127 közti integer értékeket poolban előre megcsinálja, és ha kell hivatkozik is rájuk, ezért a while függvény feltételei nem igazak. Nagyobb értéknél viszont lefut a ciklus. Ennek a célja a Java programok teljesítményének növelése.

The screenshot shows the IntelliJ IDEA interface. In the top navigation bar, the title is 'Prog2 [C:\Users\Focus\IdeaProjects\Prog2]\src\Gagyi.java (Prog2) - IntelliJ IDEA'. The menu bar includes File, Edit, View, Navigate, Code, Analyze, Refactor, Build, Run, Tools, VCS, Window, Help. The left sidebar shows a project tree with 'Prog2' selected, containing 'src' and 'idea' subfolders. The 'src' folder contains 'Gagyi.java'. The code editor window displays the following Java code:

```
import java.util.Objects;
import java.util.Scanner;

public class Gagyi {
    public static void main(String[] args) {
        Integer x, t;
        Scanner sc = new Scanner(System.in);
        System.out.println("Kérlek adja meg az x-et:");
        x = sc.nextInt();
        System.out.println("Kérlek adja meg a t-t:");
        t = sc.nextInt();

        while(x <= t && x >= t && !Objects.equals(t, x)) {
        }
    }
}
```

The terminal window below shows the output of running the program:

```
"C:\Program Files\Java\jdk-13\bin\java.exe" -Didea.launcher.port=54249 "-Didea.launcher.bin.path=C:\Program Files\JetBrains\IntelliJ IDEA 2019.2.3\bin" -Dfile.encoding=UTF-8 Gagyi
Kérlek adja meg az x-et:
5
Kérlek adja meg a t-t:
6
Process finished with exit code 0
```

2.2. ábra. gagyi

## 2.3. Yoda

Írunk olyan Java programot, ami java.lang.NullPointerException-re leáll, ha nem követjük a Yoda conditions-t!  
[https://en.wikipedia.org/wiki/Yoda\\_conditions](https://en.wikipedia.org/wiki/Yoda_conditions)

Nem mindegy, hogy milyen sorrendben van az összehasonlítás, a YodaCondition szerint a konstanst kell összehasonlítani a változóval. Így nem kapunk kivételt. A YodaCondition azért van, hogy elkerülje egy elírás által okozott váratlan viselkedéseket. A Conditionot a Star Wars szereplőjéről nevezték el, mert egyik Yoda sem követi a szabályokat.

The screenshot shows the IntelliJ IDEA interface with a Java project named 'Prog2'. The code editor displays a file named 'Yoda.java' containing the following code:

```
public class Yoda {
    public static void main(String[] args) {
        String mystring = null;
        if (mystring.equals("foobar")) { // Erre kidobja NullPointerException-t, viszont fordítható
        }
        if ("foobar".equals(mystring)) { // Ez igaz lesz
        }
    }
}
```

The code editor highlights the first 'if' statement with a red squiggle under 'mystring.equals("foobar")', indicating a potential error. The status bar at the bottom right shows 'Compilation completed successfully in 624 ms (moments ago)'.

2.3. ábra. yoda

This screenshot shows the same IntelliJ IDEA interface as the previous one, but the code has been modified. The 'Yoda.java' file now contains:

```
public class Yoda {
    public static void main(String[] args) {
        String mystring = null;
        //if (mystring.equals("foobar")) { // Erre kidobja NullPointerException-t, viszont fordítható
        //}
        if ("foobar".equals(mystring)) { // Ez igaz lesz
        }
    }
}
```

The red squiggle under the first 'if' statement has disappeared, indicating that the code is now valid. The status bar at the bottom right shows 'Compilation completed successfully in 819 ms (moments ago)'.

2.4. ábra. yoda2

## 3. fejezet

# Helló, Liskov!

### 3.1. Liskov helyettesítés sértése

Írunk olyan OO, leforduló Java és C++ kódcsipetet, amely megséríti a Liskov elvet! Mutassunk rá a megoldásra: jobb OO tervezés. [https://arato.inf.unideb.hu/batfai.norbert/UDPROG/deprecated/Prog2\\_1.pdf](https://arato.inf.unideb.hu/batfai.norbert/UDPROG/deprecated/Prog2_1.pdf) (93-99 fólia) (számos példa szerepel az elv megsértésére az UDPROG repóban, lásd pl. source/binom/Batfai-Barki/madarak/)

A Madar objektumot meghívva a program kiírja hogy a madár tud e repülni. A Program osztálynak van egy metódusa amely rendelkezik egy Madar tipusú paraméterrel. Ez az eljárás meghívja a Madar objektum repul() metódusát. A Liskov elv megsértése ebben a példában ott jelenik meg, hogy a program.fgv(pingvin) parancsnál a program kiírja, hogy a pingvin tud repülni.

```
public class Program {  
    public void fgv(Madar madar) {  
        madar.repul();  
    }  
}
```

```
public class Madar {  
    public void repul(){  
        System.out.println("repül");  
    }  
  
    public static void main(String[] args) {  
        Program program = new Program();  
        Madar madar = new Madar();  
        program.fgv(madar);  
  
        Sas sas = new Sas();  
        program.fgv(sas);  
    }  
}
```

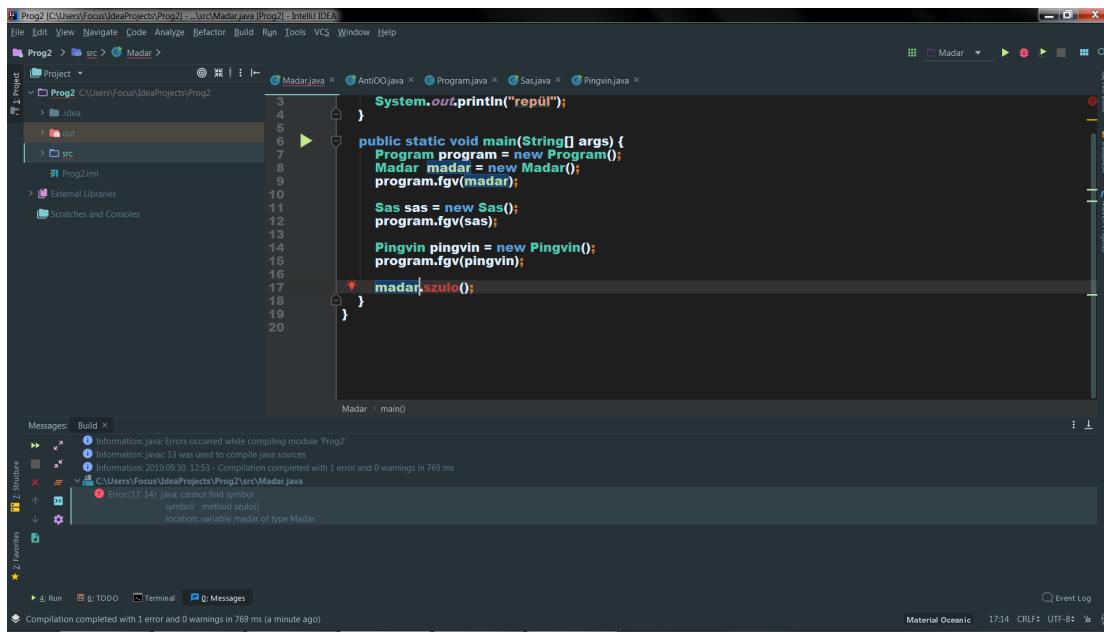
```
Pingvin pingvin = new Pingvin();
program.fgv(pingvin);
}
}
```

## 3.2. Szülő-gyerek

rjunk Szülő-gyerek Java és C++ osztálydefiníciót, amelyben demonstrálni tudjuk, hogy az ősön keresztül csak az ős üzenetei küldhetőek! [https://arato.inf.unideb.hu/batfai.norbert/UDPROG/deprecated/Prog2\\_1.pdf](https://arato.inf.unideb.hu/batfai.norbert/UDPROG/deprecated/Prog2_1.pdf) (98. fólia)4

A madaras példát dolgoztam át, hogy s Sas osztályba beleírtam egy szulo() metódust, amely a Madar-ban nem szerepel. A program fordulásnál hibát jelzett, hogy nem ismeri azt a szulo() metodust.

```
public class Sas extends Madar{
public void szulo(){
    System.out.println("szülője");
}
}
```



3.1. ábra. madarszulo

### 3.3. Anti OO

A BBP algoritmussal a Pi hexadecimális kifejtésének a 0. pozíciótól számított  $10^6$ ,  $10^7$ ,  $10^8$  darab jegyét határozzuk meg C, C++, Java és C# nyelveken és vessük össze a futási időket! <https://www.tankonyvtar.hu/hu/tartanitok-javat/apas03.html#id561066>

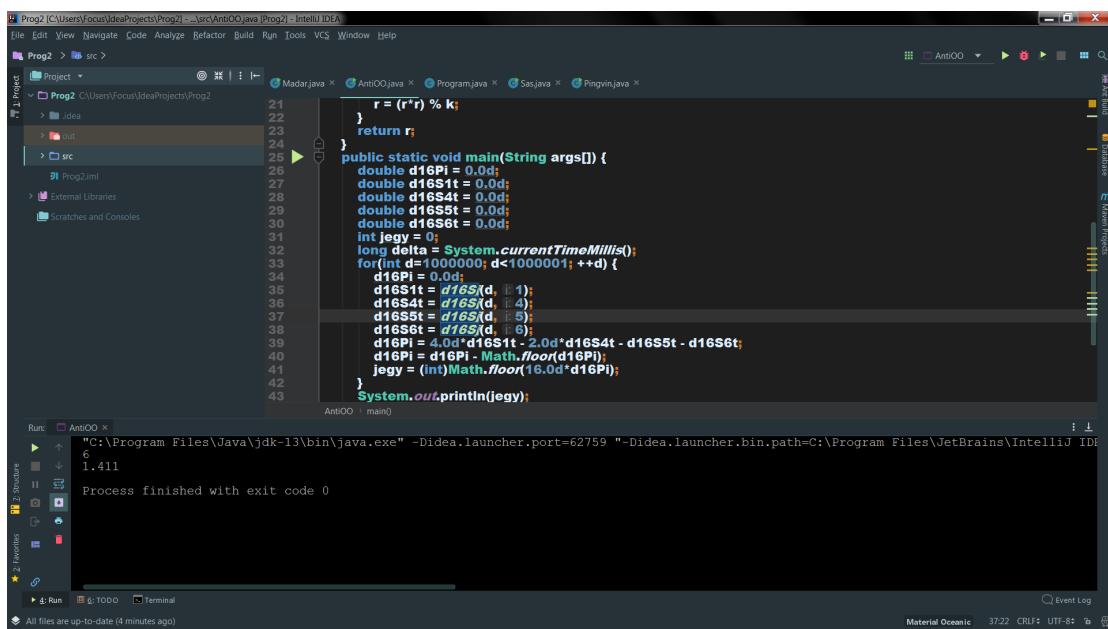
Javaban lefuttattam a programot.

```
public class Antioo {
    public static double d16Sj(int d, int j) {
        double d16Sj = 0.0d;
        for(int k=0; k<=d; ++k)
            d16Sj += (double)n16modk(d-k, 8*k + j) / (double)(8*k + j);
        return d16Sj - Math.floor(d16Sj);
    }
    public static long n16modk(int n, int k) {
        int t = 1;
        while(t <= n)
            t *= 2;
        long r = 1;
        while(true) {
            if(n >= t) {
                r = (16*r) % k;
                n = n - t;
            }
            t = t/2;
            if(t < 1)
                break;
            r = (r*r) % k;
        }
        return r;
    }
    public static void main(String args[]) {
        double d16Pi = 0.0d;
        double d16S1t = 0.0d;
        double d16S4t = 0.0d;
        double d16S5t = 0.0d;
        double d16S6t = 0.0d;
        int jegy = 0;
        long delta = System.currentTimeMillis();
        for(int d=1000000; d<1000001; ++d) {
            d16Pi = 0.0d;
            d16S1t = d16Sj(d, 1);
            d16S4t = d16Sj(d, 4);
            d16S5t = d16Sj(d, 5);
            d16S6t = d16Sj(d, 6);
            d16Pi = 4.0d*d16S1t - 2.0d*d16S4t - d16S5t - d16S6t;
            d16Pi = d16Pi - Math.floor(d16Pi);
            jegy = (int)Math.floor(16.0d*d16Pi);
```

```

        }
        System.out.println(jegy);
        delta = System.currentTimeMillis() - delta;
        System.out.println(delta/1000.0);
    }
}

```



3.2. ábra. antioo

### 3.4. Ciklomatikus komplexitás

Számoljuk ki valamelyik programunk függvényeinek ciklomatikus komplexitását! Lásd a fogalom tekintetében a [https://arato.inf.unideb.hu/batfai.norbert/UDPROG/deprecated/Prog2\\_2.pdf](https://arato.inf.unideb.hu/batfai.norbert/UDPROG/deprecated/Prog2_2.pdf) (77-79 fóliát)!

A ciklomatikus komplexitás a [https://hu.wikipedia.org/wiki/Ciklomatikus\\_komplexit%C3%A1s](https://hu.wikipedia.org/wiki/Ciklomatikus_komplexit%C3%A1s) oldal alapján gráfelméleten alapulva egy olyan számértéket határoz meg egy szoftver forráskódja alapján, amely annak a komplexitását jellemzi. Az elágazásokból felépülő gráf pontjai, és a köztük lévő élek alapján számítható ki ez a komplexitás az alábbi módon:  $M = E - N + 2P$  ahol • E: A gráf éleinek száma • N: A gráfban lévő csúcsok száma • P: Az összefüggő komponensek száma A Binfa program függvényeinek ciklomatikus komplexitását a [www.lizard.ws](http://www.lizard.ws) oldalon számoltam ki.

Try Lizard In Your Browser

File Type: Java | Token Count: 1167 | NLOC: 197

Function Name      NLOC      Complexity #      Token #      Parameter #

LZWBinFa::LZWBinFa	3	1	9	
LZWBinFa::egyBitFeldolg	19	4	104	
LZWBinFa::kilr	4	1	26	
LZWBinFa::kilr	4	1	22	
LZWBinFa::Csomopont::Csomopont	5	1	21	
LZWBinFa::Csomopont::nullasGyermek	3	1	8	
LZWBinFa::Csomopont::egyesGyermek	3	1	8	
LZWBinFa::Csomopont::ujNullasGyermek	3	1	11	
LZWBinFa::Csomopont::ujEgyesGyermek	3	1	11	
LZWBinFa::Csomopont::getBetu	3	1	8	
LZWBinFa::kilr	15	3	107	
LZWBinFa::getMelyseg	5	1	21	
LZWBinFa::getAtlag	6	1	32	
LZWBinFa::getSzoras	12	2	68	
LZWBinFa::melyseg	11	3	51	
LZWBinFa::rattag	12	4	66	
LZWBinFa::rszoras	12	4	78	
LZWBinFa::usage	3	1	14	
LZWBinFa::main	58	14	401	

3.3. ábra. binfalizard

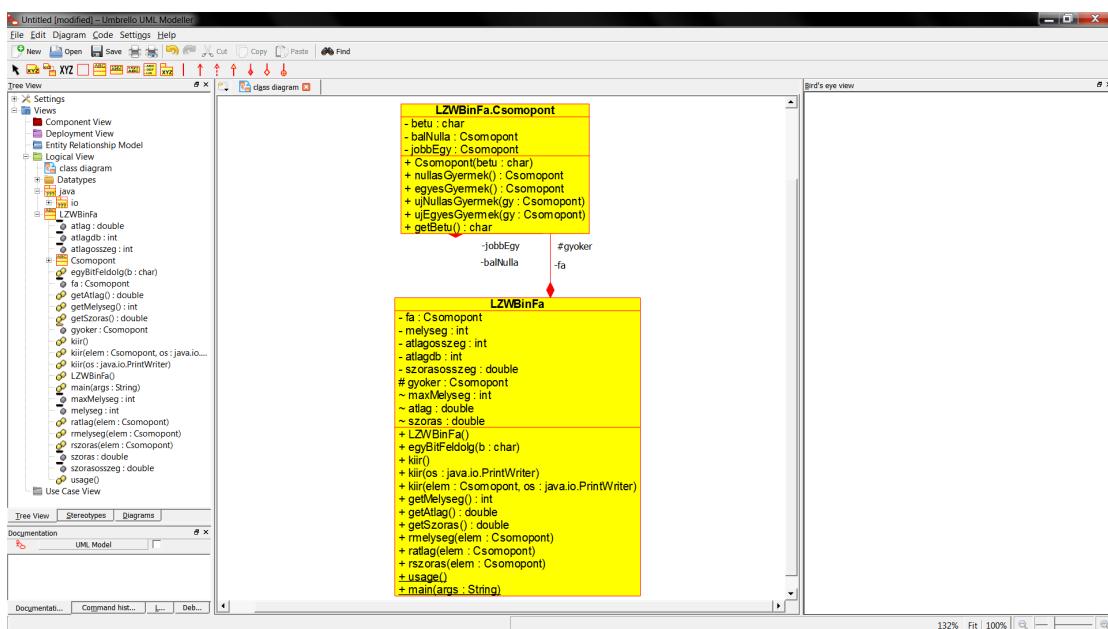
## 4. fejezet

# Helló, Mandelbrot!

### 4.1. Reverse engineering UML osztálydiagram

UML osztálydiagram rajzolása az első védési C++ programhoz. Az osztálydiagramot a forrásokból generáljuk (pl. Argo UML, Umbrello, Eclipse UML) Mutassunk rá a kompozíció és aggregáció kapcsolatára a forráskódban és a diagramon, lásd még: [https://youtu.be/Td\\_nIERIEOs](https://youtu.be/Td_nIERIEOs). <https://arato.inf.unideb.hu/batfai.norbert/UD> (28-32 fólia)

Az UML a Unified Modeling Language rövidítése. Általános célú modellező nyelv szoftvermérnökök és rendszertervezők számára. Az UML egy gyakorlati, objektum orientált modellező megoldás, nagy méretű programrendszerök modelljeinek vizuális dokumentálására alkalmas eszköz. Az UML leíró nyelv segítségevel különböző szöveges és grafikus modellek készíthetők. A feladat elkészítéséhez az Umbrello nevű programot használom. Beimportáltam a LZWBInfa Java forrását. Majd a program ebből generált egy UML diagramot. Két osztályból áll a kód, ezért két sárga részből épült fel a diagram is. Egy piros vonal két részre osztja ezt, a felsőben a változók, az alsóban a metódusok találhatók.



4.1. ábra. reverseuml

## 4.2. Forward engineering UML osztálydiagram

UML-ben tervezünk osztályokat és generálunk belőle forrást!

Tulajdonképpen az előző feladat visszafelé. Az osztályok szerkezetét ábrázoló UML diagram segítségével generáljuk a kódot. Itt is az Umbrello programot használtam. Az előző feladatban kapott UML-t használtam fel. Íme az eredmény.

```
    /**
 * Class java
 */
public class java {

    //
    // Fields
    //

    //
    // Constructors
    //
    public java () { };

    //
    // Methods
    //

    //
    // Accessor methods
    //

    //
    // Other methods
    //

}

import java.io.PrintWriter;
import LZWBinFa.Csomopont;

/**
 * Class LZWBinFa
 */
public class LZWBinFa {

    //
    // Fields
```

```
//  
  
protected Csomopont gyoker;  
private Csomopont fa;  
private int melyseg;  
private int atlagosszeg;  
private int atlagdb;  
private double szorasosszeg;  
  
//  
// Constructors  
//  
public LZWBinFa () { };  
  
//  
// Methods  
//  
  
//  
// Accessor methods  
//  
  
/**  
 * Set the value of gyoker  
 * @param newVar the new value of gyoker  
 */  
protected void setGyoker (Csmopont newVar) {  
    gyoker = newVar;  
}  
  
/**  
 * Get the value of gyoker  
 * @return the value of gyoker  
 */  
protected Csmopont getGyoker () {  
    return gyoker;  
}  
  
/**  
 * Set the value of fa  
 * @param newVar the new value of fa  
 */  
private void setFa (Csmopont newVar) {  
    fa = newVar;  
}  
  
/**  
 * Get the value of fa  
 * @return the value of fa  
 */
```

```
/*
private Csomopont getFa () {
    return fa;
}

/**
 * Set the value of melyseg
 * @param newVar the new value of melyseg
 */
private void setMelyseg (int newVar) {
    melyseg = newVar;
}

/**
 * Get the value of melyseg
 * @return the value of melyseg
 */
private int getMelyseg () {
    return melyseg;
}

/**
 * Set the value of atlagosszeg
 * @param newVar the new value of atlagosszeg
 */
private void setAtlagosszeg (int newVar) {
    atlagosszeg = newVar;
}

/**
 * Get the value of atlagosszeg
 * @return the value of atlagosszeg
 */
private int getAtlagosszeg () {
    return atlagosszeg;
}

/**
 * Set the value of atlagdb
 * @param newVar the new value of atlagdb
 */
private void setAtlagdb (int newVar) {
    atlagdb = newVar;
}

/**
 * Get the value of atlagdb
 * @return the value of atlagdb
 */
private int getAtlagdb () {
```

```
        return atlagdb;
    }

/***
 * Set the value of szorasosszeg
 * @param newVar the new value of szorasosszeg
 */
private void setSzorasosszeg (double newVar) {
    szorasosszeg = newVar;
}

/***
 * Get the value of szorasosszeg
 * @return the value of szorasosszeg
 */
private double getSzorasosszeg () {
    return szorasosszeg;
}

//  

// Other methods  

//  

/***
 */
public void LZWBinFa()
{
}

/***
 * @param      b
 */
public void egyBitFeldolg(char b)
{
}

/***
 */
public void kiir()
{
}

/***
 * @param      os
 */
public void kiir(java.io.PrintWriter os)
{
```

```
}

/**
 * @param elem
 * @param os
 */
public void kiir(LZWBInFa.Csomopont elem, java.io.PrintWriter os)
{
}

/**
 * @return int
 */
public int getMelyseg()
{
}

/**
 * @return double
 */
public double getAtlag()
{
}

/**
 * @return double
 */
public double getSzoras()
{
}

/**
 * @param elem
 */
public void rmelyseg(LZWBInFa.Csomopont elem)
{
}

/**
 * @param elem
 */
public void ratlag(LZWBInFa.Csomopont elem)
{}
```

```
 /**
 * @param elem
 */
public void rszoras(LZWBinFa.Csomopont elem)
{
}

/**
 */
public static void usage()
{
}

/**
 * @param args
 */
public static void main(String args)
{
}

}
```

### 4.3. Egy esettan

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

A fejezet párhuzamot von különböző asszociációk, aggregáció, kompozíció és C++ nyelvben használt adattípusok között. A láthatóságról is említést tesz a fejezet elején. A package típusra nem találunk példát a nyelv hiányosságai miatt. A szerző megemlíti a forward engineering-et és reverse engineering-et vagyis a kódgenerálást és kódvisszafejtést a fejezet végén. Az esettanulmány fejezetben egy számítógép alkatrészekkel és számítógép konfigurációkkal foglalkozó kereskedésnek írunk programot, hogy a jövőben újabb termékekkel is bővülhet a készlet. Úgy kell megtervezni az osztályokat, hogy a termékek alkatrészek is lehetnek vagy azokból álló konfigurációk is. A Product osztályból ezért CompositeProduct, Display és HardDisk alosztályok is leszármaztathatók, illetve a CompositeProduct-ból leszármaztatható a Computer-Configuration alosztály is. Egy termék az alábbi tulajdonságokkal rendelkezik: GetDateOfAcquisition() (beszerzés időpontja), GetInitialPrice() (beszerzés ára) és GetName() (beszerzés neve). Ezek a Product tagfüggvényei. A GetAge() kiszámolja a termék korát az aktuális és beszerzési dátumból. A GetCurrentPrice() termékenként változó.

```
//File Proudct.H
```

```
#ifndef PRODUCT_H
#define PRODUCT_H

#include <iostream>
#include <ctime>
#include <string>

class Product {
protected:
    int initialPrice;
    time_t dateOfAcquisition;
    std::string name;
    virtual void printParams(std::ostream& os) const;
    virtual void loadParamsFromStream(std::istream& is);
    virtual void writeParamsToStream(std::ostream& os) const;
public:
    Product();
    Product(std::string name, int initialPrice, time_t dateOfAcquisition) ←
        ;
    virtual ~Product() {};
    int getInitialPrice() const;
    std::string getName() const;
    time_t getDateOfAcquisition() const;
    int getAge() const;
    virtual int getCurrentPrice() const;
    void Print(std::ostream& os) const;
    virtual std::string getType() const = 0;
    virtual char getCharCode() const = 0;
    bool operator< (const Product& other) const;

    friend std::istream& operator>>(std::istream& is, Product& product);
    friend std::ostream& operator<<(std::ostream& os, const Product& product) ←
        );
};

#endif

//File Product.cpp
#include "product.h"
#include <iostream>
#include <ctime>
#include <string.h>
#include <stdio.h>
#include <stdlib.h>

Product::Product() {}

Product::Product(std::string name, int initialPrice, time_t ←
    dateOfAcquisition):name(name), ← initialPrice(initialPrice), ←
    dateOfAcquisition ← ( ← dateOfAcquisition ← ) {}
```

```
time_t Product::getDateOfAcquisition() const {
    return dateOfAcquisition;
}
int Product::getInitialPrice() const {
    return initialPrice;
}
std::string Product::getName() const {
    return name;
}
int Product::getCurrentPrice() const {
    return initialPrice;
}
int Product::getAge() const {
    time_t currentTime;
    time(&currentTime);
    double timeDiffInSec = difftime(currentTime,dateOfAcquisition);
    return (int)(timeDiffInSec)/(3600*24);
}
void Product::Print(std::ostream& os) const {
    os << "Type: " << getType() << ", ";
    os << "Name: " << getName();
    printParams(os);
}
void Product::printParams(std::ostream& os) const {
    char strDateOfAcquisition[9];
    strftime(strDateOfAcquisition , 9, "%Y%m%d" , gmtime(&dateOfAcquisition ←
        ));
    os << ", " << "Initial Price: " << initialPrice << ", " << "Date of ←
        Acquisition: " << dateOfAcquisition << ", " << "Age: " << getAge() ←
        << ", " << "Current price: " << getCurrentPrice();
}
void Product::writeParamsToStream(std::ostream& os) const {
    char strDateOfAcquisition[9];
    tm* t = localtime(&dateOfAcquisition);
    strftime(strDateOfAcquisition,9,"%Y%m%d",t);
    os << " " << name << " " << initialPrice << " " << strDateOfAcquisition ←
        ;
}
void Product::loadParamsFromStream(std::istream& is) {
    is >> name;
    is >> initialPrice;
    char buff[9];
    is.width(9);
    is >> buff;
    if (strlen(buff) != 8) throw ("Invalid time format.");
    char workBuff[5];
    tm t;
    int year;
    strncpy(workBuff,buff,4);
    workBuff[4] = '\0';
```

```
year = atoi(workBuff);
t.tm_year = year - 1900;
strncpy(workBuff,&buff[4],2);
workBuff[2] = '\0';
t.tm_mon = atoi(workBuff) - 1;
strncpy(workBuff,&buff[6],2);
workBuff[2] = '\0';
t.tm_mday = atoi(workBuff);
t.tm_hour = t.tm_min = t.tm_sec = 0;
t.tm_isdst = -1;
dateOfAcquisition = mktime(&t);
}

bool Product::operator < (const Product& other) const {
    std::locale loc;
    return loc(name,other.name);
}

std::istream& operator>>(std::istream& is,Product& product) {
    product.loadParamsFromStream(is);
    return is;
}

std::ostream& operator<<(std::ostream& os, const Product& product) {
    os << product.getCharCode();
    product.writeParamsToStream(os);
    return os;
}

//File Display.h
#ifndef DISPLAY_H
#define DISPLAY_H

#include "product.h"
#include <string>

class Display : public Product {
    int inchWidth;
    int inchHeight;
public:
    Display();
    Display(std::string name , int price , time_t time , int number1 , int ←
            ← number2);
    std::string getType() const;
    char getCharCode() const;
protected:
    void printParams(std::ostream& os) const;
    void writeParamsToStream(std::ostream& os) const;
    void loadParamsFromStream(std::istream& is);
};

#endif

//File Display.cpp
```

```
#include "display.h"
#include "product.h"
#include <iostream>

Display::Display(std::string name , int price , time_t date , int inchWidth ←
    , int ← inchHeight) {
    Product::name = name;
    Product::dateOfAcquisition = date;
    Product::initialPrice = price;
    Display::inchWidth = inchWidth;
    Display::inchHeight = inchHeight;
}

Display::Display() {}

void Display::printParams(std::ostream& os) const {
    Product::printParams(os);
    os << ", " << "InchWidth: " << inchWidth;
    os << ", " << "InchHeight: " << inchHeight;
}

void Display::writeParamsToStream(std::ostream& os) const {
    Product::writeParamsToStream(os);
    os << ' ' << inchWidth << ' ' << inchHeight;
}

void Display::loadParamsFromStream(std::istream& is) {
    Product::loadParamsFromStream(is);
    is >> inchWidth >> inchHeight;
}

std::string Display::getType() const {
    return Product::name;
}

char Display::getCharCode() const {
    return 'd';
}

//file HardDisk.cpp
#include "product.h"
//#include <string>
class HardDisk : public Product {
    int speedRPM;
    static const char charCode = 'h';
public:
    HardDisk() {} HardDisk(std::string name, int price , time_t date ,int ←
        speedRPM) {
        Product::name = name;
        Product::initialPrice = price;
        Product::dateOfAcquisition = date;
        this->speedRPM = speedRPM;
    }
    int getCurrentPrice() const {
        int ageInDays = getAge();
        if (ageInDays < 30) return initialPrice;
    }
}
```

```
        else if (ageInDays >= 30 && ageInDays < 90) return (int)(←
            initialPrice * ← 0.9);
        else return (int)(initialPrice * ← 0.8);
    } std::string getType() const {
        return "HardDisk";
    } char getCharCode() const {
        return charCode;
    }
};

//File ProductFactory.h
#ifndef PRODUCTFACTORY_H
#define PRODUCTFACTORY_H
#include <iostream>
#include "product.h"
class ProductFactory{
static ProductFactory* instance;
public:
static void init(ProductFactory* pf);
static ProductFactory* getInstance();
Product* readAndCreateProduct(std::istream& is);
virtual Product* createProduct(char typeCode) const = 0;
};
#endif

//File ProductFactory.cpp
#include "productfactory.h"
ProductFactory* ProductFactory::instance = NULL;
Product* ProductFactory::readAndCreateProduct(std::istream& is) {
    if (!is.good()) return NULL;
    char typeCode;
    is >> typeCode;
    if (!is.good()) {
        if (is.eof()) return NULL;
        std::cout << "There was an error reading the product items!" << std::endl;
        return NULL;
    }
    Product* product = createProduct(typeCode);
    if (product == NULL) std::cout << "unknown product type" << std::endl;
    return product;
}
void ProductFactory::init(ProductFactory* pf) {
    instance = pf;
}
ProductFactory* ProductFactory::getInstance() {
    return instance;
}

//File ProductInventory.h
```

```
#ifndef PRODUCTINVENTORY_H
#define PRODUCTINVENTORY_H
#include <vector>
#include <iostream>
#include "product.h"
class ProductInventory {
    void emptyProducts();
protected:
    std::vector<Product*> products;
public:
    virtual ~ProductInventory();
    void readInventory(std::istream& is);
    void writeInventory(std::ostream& os) const;
    void printProducts(std::ostream& os) const;
    void addProduct(Product* product);
};

#endif

//File ProductInventory.cpp

#include "productinventory.h"
#include "productfactory.h"
ProductInventory::~ProductInventory() {
    emptyProducts();
}
void ProductInventory::emptyProducts() {
    for (unsigned int i = 0 ; i < products.size() ; i++) delete products[i];
    products.clear();
}
void ProductInventory::printProducts(std::ostream& os) const {
    for (unsigned int i = 0 ; i < products.size() ; i++) {
        os << i << ".: ";
        products[i]->Print(os);
        os << std::endl;
    }
}
void ProductInventory::readInventory(std::istream& is) {
    is >> std::ws;
    while (is.good()) {
        Product* product = ProductFactory::getInstance()->readAndCreateProduct(is);
        if (product) {
            is >> *product;
            addProduct(product);
        }
    }
    std::cout << "End of reading product items.";
}
void ProductInventory::writeInventory(std::ostream& os) const {
```

```
for (unsigned int i = 0 ; i < products.size() ; i++) {
    os << *products[i] << std::endl;
}
}

void ProductInventory::addProduct(Product* product) {
    if (product == NULL) {
        throw "The product parameter cannot be null";
    }
    products.push_back(product);
}

//File ComputerProductFactory.h
#ifndef COMPUTERPRODUCTFACTORY_H
#define COMPUTERPRODUCTFACTORY_H
#include "productfactory.h"
#include "display.h"
#include "hardisk.cpp"
class ComputerProductFactory : public ProductFactory {
public:
    Product* createProduct(char typeCode) const;
};

#endif

//File ComputerProductFactory.cpp
#include "computerproductfactory.h"
Product* ComputerProductFactory::createProduct(char typeCode) const {
    switch(typeCode) {
    case 'd':
        return new Display();
    case 'h':
        return new HardDisk(); //case 'c': return new ComputerConfiguration ←
        () ; } return NULL;
    }
}

//File CompositeProduct.h
#ifndef COMPOSITEPRODUCT_H
#define COMPOSITEPRODUCT_H
#include "product.h"
#include <vector>
#include <iostream>
class CompositeProduct : public Product {
    std::vector<Product*> parts;
protected:
    void printParams(std::ostream& os) const;
    void loadParamsFromStream(std::istream& is);
    void writeParamsToStream(std::ostream& os) const;
public:
    CompositeProduct();
    ~CompositeProduct();
    void addPart(Product* product);
};
```

```
#endif

//File CompositeProduct.cpp
#include "compositeproduct.h"
#include "productfactory.h"
#include "algorithm"
#include <iostream>
CompositeProduct::CompositeProduct():Product() {} void CompositeProduct ←
    ::addPart(Product* product) {
    parts.push_back(product);
}
CompositeProduct::~CompositeProduct() {
    for (unsigned int i = 0 ; i < parts.size() ; i++) delete parts[i];
    parts.clear();
//for_each(parts.begin(),parts.end(),delete_ptr());
}
void CompositeProduct::printParams(std::ostream& os) const {
    Product::printParams(os);
    os << std::endl << " Items:" ;
    for (unsigned int i = 0 ; i < parts.size() ; i++) {
        os << std::endl << i << ". :" ;
        parts[i]->Print(os);
    }
}
void CompositeProduct::writeParamsToStream(std::ostream& os) const {
    Product::writeParamsToStream(os);
    os << " " << parts.size();
    for (unsigned int i = 0 ; i < parts.size() ; i++) {
        os << std::endl << *parts[i];
    }
}
void CompositeProduct::loadParamsFromStream(std::istream& is) {
    Product::loadParamsFromStream(is);
    int itemCount;
    is >> itemCount;
    for (int i = 0; i < itemCount ; i++) {
        Product* product = ProductFactory::getInstance()-> ←
            readAndCreateProduct(is);
        if (product) {
            is >> *product;
            addPart(product);
        }
    }
}

//File Date.cpp
#include <ctime>
#include <string.h>
#include <iostream>
```

```
class Date {
    std::tm time;
    template<class charT, class Traits> friend std::basic_istream<charT, ←
        Traits>& operator>> (std::basic_istream<charT,Traits>&,Date&);
    template<class charT, class Traits> friend std::basic_ostream<charT, ←
        Traits>& operator<< (std::basic_ostream<charT,Traits>&, const ←
        Date&);
    template<class charT, class Traits> friend std::basic_ostream<charT, ←
        Traits>& operator<< (std::basic_ostream<charT,Traits>& os, const ←
        Date& t)
        std::basic_ostream<charT,Traits>::sentry sntr(os);
        if(!sntr) return os;
        char* pattern = "%Y%m%d";
        std::use_facet<std::time_put<charT,ostreambuf_iterator<charT, ←
            Traits>>>(os. ← getloc()).put(os,os,os.fill(),&t.time, ←
            pattern,pattern+strlen(pattern));
        os.width(0);
        return os;
    }
public:
    Date() {
        memset(&time,0,sizeof(std::tm));
        time_t now = std::time(NULL);
        time = *localtime(&now);
    }
    Date(const std::tm& time):time(time) {}
    Date(time_t t) {
        time = *localtime(&t);
    }
    int getElapsedDays() const {
        time_t currentTime;
        std::time(&currentTime);
        double timeDiffInSec = difftime(currentTime,mktime(const_cast<←
            tm*>(&time)));
        return (int)(timeDiffInSec/3600*24);
    }
};

//File ProductInventoryTest.cpp
#include <fstream>
#include "productinventory.h"
#include "productfactory.h"
#include "computerproductfactory.h"
#include "display.h"
//#include "hardisk.cpp"
void readInvFromFileTest(ProductInventory& inv);
void writeInvToFileTest(ProductInventory& inv);
int main(int argc, char** argv) {
    try {
        ProductFactory::init(new ComputerProductFactory());
    }
```

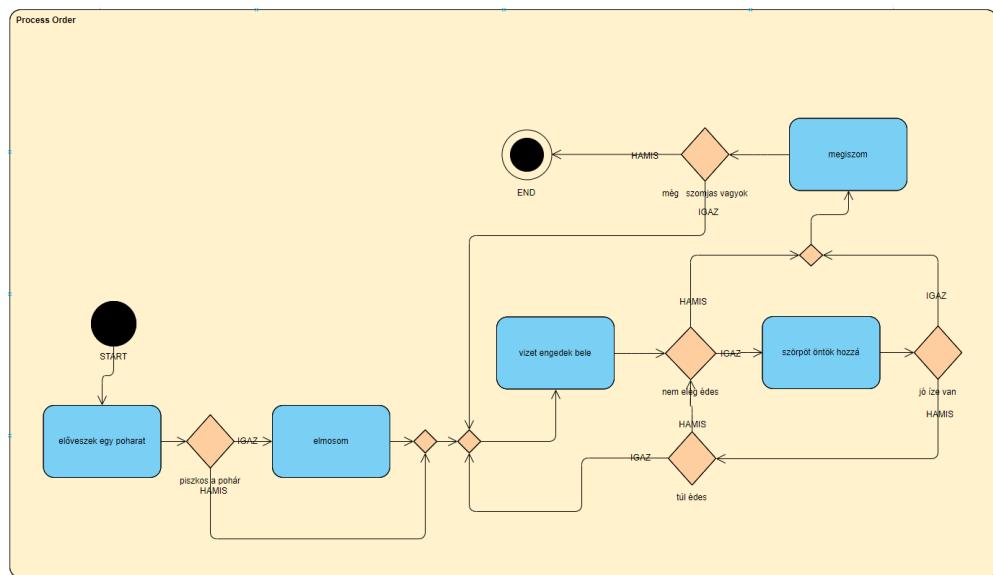
```
    std::cout << "Test1: creating inventory and printing" << std::endl;
    time_t currentTime;
    time(&currentTime);
    ProductInventory inv1;
    inv1.addProduct(new Display("TF1", 30000, currentTime, 13, 12));
    inv1.addProduct(new HardDisk("wd", 25000, currentTime, 7500));
    inv1.printProducts(std::cout);
    std::cout << "press any key to continue" << std::endl;
    std::cin.get();
    std::cout << std::endl;
    std::cout << "Test 2 loading inventory form a file ( ←
        computerproducts.txt) and then ← writing to a file ( ←
        computerproducts_out.txt)" << std::endl;
    ProductInventory inv2;
    readInvFromFileTest(inv2);
    writeInvToFileTest(inv2);
    std::cout << std::endl;
    std::cout << "Done. ";
    std::cin.get();
    return 0;
}
catch(const std::exception& e) {
    std::cerr << "There was an error" << std::endl;
    std::cerr << e.what() << std::endl;
}
catch(...) {
    std::cout << "unexpected error occurred" << std::endl;
}
void readInvFromFileTest(ProductInventory& inv) {
    std::ifstream fs("computerproducts.txt");
    if(!fs) {
        std::cerr << "Error opening file" << std::endl;
        return;
    }
    inv.readInventory(fs);
    std::cout << "The content of computerproducts.txt is:" << std::endl ←
        ;
    inv.printProducts(std::cout);
    std::cout << std::endl;
}
void writeInvToFileTest(ProductInventory& inv) {
    std::ofstream fs("computerproducts_out.txt");
    if (!fs) {
        std::cerr << "Error opening file" << std::endl;
        return;
    }
    inv.writeInventory(fs);
    std::cout << "The content has been written to computerproducts_out. ←
```

```
    txt" << std::endl;
}
```

## 4.4. BPMN

Rajzoljunk le egy tevékenységet BPMN-ben! <https://arato.inf.unideb.hu/batfai.norbert/UDPROG/deprecated/Prog34-47.pdf> (34-47 fólia)

A folyamatábra a szörpivást írja le. Előveszek egy poharat. Ha piszkos, akkor elmosom, különben meg (1) töltök bele vizet. (2) Ha nem elég édes, öntök hozzá szörpöt, különben megiszom. Ha jó íze van, akkor is megiszom, különben ha túl édes, akkor visszaugrás a 1-es pontra. De ha a túl édes elágazás hamis, akkor ugrás a 2-es pontra. Mikor megittam és szomjas vagyok még, akkor ugrás az 1-es pontra. Ha nem vagyok szomjas akkor csillapítva lett a szomjam, és véget ér az algoritmus.



4.2. ábra. szorp

## 5. fejezet

# Helló, Chomsky!

### 5.1. Encoding

Fordítsuk le és futtassuk a Javat tanítok könyv MandelbrotHalmazNagyító.java forrását úgy, hogy a fájl nevekben és a forrásokban is meghagyjuk az ékezes betűket! <https://www.tankonyvtar.hu/hu/tartalom/tkt/javat-tanitok-javat/adatok.html>

Általában a programjaink nevében és a forráskódban nem alkalmazunk ékezes betűket, mert ez kódolási problémákhoz vezethet. A példa programban nem tartjuk ezt be. Ezért esetünkben ISO8859\_1-os kódolást használunk. Így a programunk képes kezelni az ékezes karaktereket. A Settings -> Editor -> File Encodings fülön lehet beállítani. És pluszba még a másik két java fájlt is bemásoljuk a

```
/*
 * MandelbrotHalmazNagyító.java
 *
 * DIGIT 2005, Javat tanítok
 * Bátfai Norbert, nbatfai@inf.unideb.hu
 *
 */
/**
 * A Mandelbrot halmazt nagyító és kirajzoló osztály.
 *
 * @author Bátfai Norbert, nbatfai@inf.unideb.hu
 * @version 0.0.2
 */
public class MandelbrotHalmazNagyító extends MandelbrotHalmaz {
    /** A nagyítandó kijelölt területet bal felső sarka. */
    private int x, y;
    /** A nagyítandó kijelölt terület szélessége és magassága. */
    private int mx, my;
    /**
     * Létrehoz egy a Mandelbrot halmazt a komplex sík
     * [a,b]x[c,d] tartománya felett kiszámoló és nyígtani tudó
     * <code>MandelbrotHalmazNagyító</code> objektumot.
     *
```

```
* @param      a          a [a,b]x[c,d] tartomány a koordinátája.
* @param      b          a [a,b]x[c,d] tartomány b koordinátája.
* @param      c          a [a,b]x[c,d] tartomány c koordinátája.
* @param      d          a [a,b]x[c,d] tartomány d koordinátája.
* @param      szélesség    a halmazt tartalmazó tömb szélessége.
* @param      iterációsHatár a számítás pontossága.
*/
public MandelbrotHalmazNagyító(double a, double b, double c, double d,
        int szélesség, int iterációsHatár) {
    // Az ōs osztály konstruktorának hívása
    super(a, b, c, d, szélesség, iterációsHatár);
    setTitle("A Mandelbrot halmaz nagyításai");
    // Egér kattintó események feldolgozása:
    addMouseListener(new java.awt.event.MouseAdapter() {
        // Egér kattintással jelöljük ki a nagyítandó területet
        // bal felső sarkát vagy ugyancsak egér kattintással
        // vizsgáljuk egy adott pont iterációt:
        public void mousePressed(java.awt.event.MouseEvent m) {
            // Az egérmutató pozíciója
            x = m.getX();
            y = m.getY();
            // Az 1. egér gombbal a nagyítandó terület kijelölését
            // végezzük:
            if(m.getButton() == java.awt.event.MouseEvent.BUTTON1) {
                // A nagyítandó kijelölt területet bal felső sarka: (x, ←
                y)
                // és szélessége (majd a vonzolás növeli)
                mx = 0;
                my = 0;
                repaint();
            } else {
                // Nem az 1. egér gombbal az egérmutató mutatta c
                // komplex számból indított iterációkat vizsgálhatjuk
                MandelbrotIterációk iterációk =
                    new MandelbrotIterációk(
                        MandelbrotHalmazNagyító.this, 50);
                new Thread(iterációk).start();
            }
        }
        // Vonzolva kijelölünk egy területet...
        // Ha felengedjük, akkor a kijelölt terület
        // újraszámítása indul:
        public void mouseReleased(java.awt.event.MouseEvent m) {
            if(m.getButton() == java.awt.event.MouseEvent.BUTTON1) {
                double dx = (MandelbrotHalmazNagyító.this.b
                            - MandelbrotHalmazNagyító.this.a)
                            /MandelbrotHalmazNagyító.this.szélesség;
                double dy = (MandelbrotHalmazNagyító.this.d
                            - MandelbrotHalmazNagyító.this.c)
                            /MandelbrotHalmazNagyító.this.magasság;
```

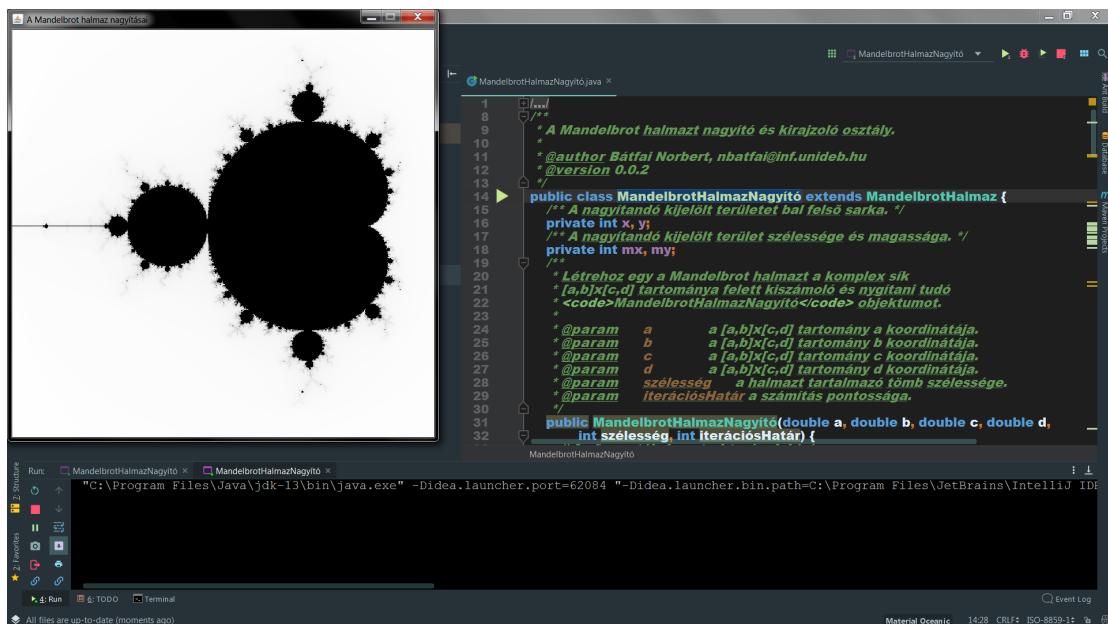
```
// Az új Mandelbrot nagyító objektum elkészítése:  
new MandelbrotHalmazNagyító(  
    MandelbrotHalmazNagyító.this.a+x*dx,  
    MandelbrotHalmazNagyító.this.a+x*dx+mx*dx,  
    MandelbrotHalmazNagyító.this.d-y*dy-my*dy,  
    MandelbrotHalmazNagyító.this.d-y*dy,  
    600,  
    MandelbrotHalmazNagyító.this.iterációsHatár);  
}  
}  
});  
// Egér mozgás események feldolgozása:  
addMouseMotionListener(new java.awt.event.MouseMotionAdapter() {  
    // Vonszolással jelöljük ki a négyzetet:  
    public void mouseDragged(java.awt.event.MouseEvent m) {  
        // A nagyítandó kijelölt terület szélessége és magassága:  
        mx = m.getX() - x;  
        my = m.getY() - y;  
        repaint();  
    }  
});  
}  
/**  
 * Pillanatfelvételek készítése.  
 */  
public void pillanatfelvétel() {  
    // Az elmentendő kép elkészítése:  
    java.awt.image.BufferedImage mentKép =  
        new java.awt.image.BufferedImage(szélesség, magasság,  
            java.awt.image.BufferedImage.TYPE_INT_RGB);  
    java.awt.Graphics g = mentKép.getGraphics();  
    g.drawImage(kép, 0, 0, this);  
    g.setColor(java.awt.Color.BLACK);  
    g.drawString("a=" + a, 10, 15);  
    g.drawString("b=" + b, 10, 30);  
    g.drawString("c=" + c, 10, 45);  
    g.drawString("d=" + d, 10, 60);  
    g.drawString("n=" + iterációsHatár, 10, 75);  
    if(számításFut) {  
        g.setColor(java.awt.Color.RED);  
        g.drawLine(0, sor, getWidth(), sor);  
    }  
    g.setColor(java.awt.Color.GREEN);  
    g.drawRect(x, y, mx, my);  
    g.dispose();  
    // A pillanatfelvétel képfájl nevének képzése:  
    StringBuffer sb = new StringBuffer();  
    sb = sb.delete(0, sb.length());  
    sb.append("MandelbrotHalmazNagyitas_");  
    sb.append(++pillanatfelvételszámláló);
```

```
sb.append("_");
// A fájl nevébe belevesszük, hogy melyik tartományban
// találtuk a halmazt:
sb.append(a);
sb.append("_");
sb.append(b);
sb.append("_");
sb.append(c);
sb.append("_");
sb.append(d);
sb.append(".png");
// png formátumú képet mentünk
try {
    javax.imageio.ImageIO.write(mentKép, "png",
        new java.io.File(sb.toString()));
} catch(java.io.IOException e) {
    e.printStackTrace();
}
}
/**
 * A nagyítandó kijelölt területet jelző négyzet kirajzolása.
 */
public void paint(java.awt.Graphics g) {
    // A Mandelbrot halmaz kirajzolása
    g.drawImage(kép, 0, 0, this);
    // Ha éppen fut a számítás, akkor egy vörös
    // vonallal jelöljük, hogy melyik sorban tart:
    if(számításFut) {
        g.setColor(java.awt.Color.RED);
        g.drawLine(0, sor, getWidth(), sor);
    }
    // A jelző négyzet kirajzolása:
    g.setColor(java.awt.Color.GREEN);
    g.drawRect(x, y, mx, my);
}
/**
 * Hol áll az egérmutató?
 * @return int a kijelölt pont oszlop pozíciója.
 */
public int getX() {
    return x;
}
/**
 * Hol áll az egérmutató?
 * @return int a kijelölt pont sor pozíciója.
 */
public int getY() {
    return y;
}
/**
```

```

* Példányosít egy Mandelbrot halmazt nagyító obektumot.
*/
public static void main(String[] args) {
    // A kiinduló halmazt a komplex sík [-2.0, .7]x[-1.35, 1.35]
    // tartományában keressük egy 600x600-as hálóval és az
    // aktuális nagyítási pontossággal:
    new MandelbrotHalmazNagyító(-2.0, .7, -1.35, 1.35, 600, 255);
}
}

```



5.1. ábra. mandelbrothalmaznagyító

## 5.2. I334d1c45

Írj olyan OO Java vagy C++ osztályt, amely leet cipherként működik, azaz megvalósítja ezt a betű helyettesítést: <https://simple.wikipedia.org/wiki/Leet> (Ha ez első részben nem tettek meg, akkor írasd ki és magyarázd meg a használt struktúratomb memória foglalását!)

Létrehoztam egy függvényt csere néven amely tartalmaz két string tömböt. Az egyikben az abc betűi a másikban a leet abc betűi vannak. A program bekér egy szót, majd a csere függvényem elindítom ezzel a be paraméterrel. A függvény két for ciklus segítségével megkeresi a betűk hanyadik indexen szerepelnek az abc tömbben és kicséri a leet string tömb azonos indexű elemére. A leet abc megtalálható itt: <https://simple.wikipedia.org/wiki/Leet>

```
import java.util.Scanner;
```

```
public class Leet {  
  
    public static String csere (String szo){  
        String ki="";  
        String abc[]={ " ", "a", "b", "c", "d", "e", "f", "g", "h", "i", "j", "k", "l", "m", "n", "o", "p", "q", "r", "s",  
                    "t", "u", "v", "w", "z", "0", "1", "2", "3", "4", "5", "6", "7", "8", "9" };  
        String leet[]={ " ", "4", "8", "(", "|)", "3", "|=", "6", "|-", "1", "_/", "|< ",  
                     "|_", "|v|", "/v", "0", "|o", "O_ ",  
                     "|2", "5", "-|-", "|/", "VV", "><", "'/", "2", "O", "I", "Z", "E", "A",  
                     "S", "G", "T", "X", "J" };  
        for (int i = 0; i<szo.length();i++) {  
            for (int j = 0; j < abc.length; j++) {  
                if (String.valueOf(szo.charAt(i)).equals(abc[j]))  
                    ki=ki+leet[j];  
            }  
        }  
        return ki;  
    }  
    public static void main(String[] args) {  
  
        Scanner sc = new Scanner(System.in);  
        String be = sc.next();  
        System.out.println(csere(be));  
  
    }  
}
```

The screenshot shows the IntelliJ IDEA interface. The top navigation bar includes File, Edit, View, Navigate, Code, Analyze, Refactor, Build, Run, Tools, VCS, Window, and Help. The main window has a Project view on the left showing a 'Prog2' project with a 'src' folder containing 'Leet.java'. The 'Leet.java' code editor is open, displaying the following Java code:

```

String ki="";
String abc[]={"a","b","c","d","e","f","g","h","i","j","k","l","m","n","o","p","q","r","s",
              "t","u","v","w","x","y","z"};
String leet[]={"4","8","|","3","|=","6","|_|","1","|_|","|<","|_","|V|","/V","0","o","O_",
               "|2","|5","|_|","|/","VV",">","|>","2","0","l","z","E","A","S","G","T","X","J"};
for (int i = 0; i<szzo.length();i++) {
    for (int j = 0; j < abc.length; j++) {
        if (String.valueOf(szzo.charAt(i)).equals(abc[j])) {
            ki=ki+leet[j];
        }
    }
}
return ki;
}
public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    String be = sc.nextLine();
    System.out.println(csere(be));
}

```

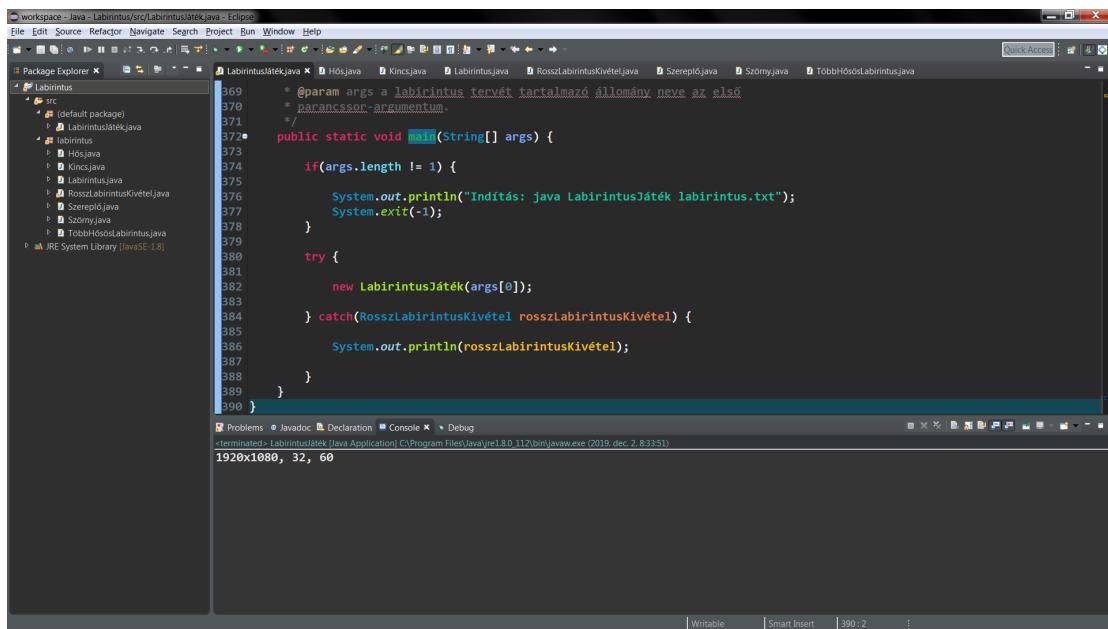
The terminal window at the bottom shows the command run: "C:\Program Files\Java\jdk-13\bin\java.exe" -Didea.launcher.port=59486 -Didea.launcher.bin.path=C:\Program Files\JetBrains\IntelliJ IDEA 2019.2.3\bin" and the output: "leet" and "33-1-". Below the terminal is a status bar indicating "All files are up-to-date (moments ago)".

5.2. ábra. leet

### 5.3. Full screen

Készítünk egy teljes képernyős Java programot! Tipp: [https://www.tankonyvtar.hu/en/tartalom/tkt/javat-tanitok-javat/ch03.html#labirintus\\_jatek](https://www.tankonyvtar.hu/en/tartalom/tkt/javat-tanitok-javat/ch03.html#labirintus_jatek)

Követtem az alábbi linken [https://www.tankonyvtar.hu/en/tartalom/tkt/javat-tanitok-javat/ch03.html#labirintus\\_jatek](https://www.tankonyvtar.hu/en/tartalom/tkt/javat-tanitok-javat/ch03.html#labirintus_jatek) lévő utasításokat. Létrehoztam egy új projectet az Eclipseben. Beimportáltam a LabirintusJáték.java fájlt. Később ezzel indítottam a játékot, mert ebben van a main függvény. Készítettem egy labirintus csomagot, amibe 7 db fájlt kellett másolni a hibátlan futás érdekében, hogy pontosan melyek azok, az a képen fel van tüntetve. A labirintus.txt állományt, melyet futási argumentumként kellett beírni a Labirintus könyvtáron belül hoztam létre. Végül még ide bemásoltam a szükséges png állományokat. Elég érdekes volt a méhek mozgása, mert sokszor mintha falán keresztül is láttak volna és követték a mozgásom. Sokadik próbálkozásra sikerült megnyerni. A teljes képernyős futáshoz a java.awt.GraphicsEnvironment.getLocalGraphicsEnvironment() metódus segít hozzá. Innen tudjuk lekérdezni a számítógép képernyőjét. Ezt a graphicsEnvironment.getDefaultScreen() függvény segítségével kell megtenni. Ez a képernyő változó tovább van adva a teljesKépernyősMód eljárásnak. Ez az eljárás végzi az alkalmazás teljes képernyőssé állítását. Nem támogatja minden kijelző a teljes képernyős módot, ezért arra szolgál a graphicsDevice.isFullScreenSupported() függvény, hogy ezt leellenőrizze. Ha támogatja, akkor a graphicsDevice.setFullScreenWindow(this) eljárással teljes képernyőssé tudjuk állítani a képet.



The screenshot shows the Eclipse IDE interface with the 'LabirintusJáték.java' file open in the editor. The code implements a command-line application for a labyrinth game. It checks if the correct number of arguments are provided and prints an error message if not. It then creates a new instance of the 'LabirintusJáték' class with the provided argument and catches any potential exceptions related to a bad labyrinth specification.

```
369     * Úgy param args a labirintus tervét tartalmazó állomány neve az első
370     * parancssor argumentumum.
371     */
372    public static void main(String[] args) {
373        if(args.length != 1) {
374            System.out.println("Indítás: java LabirintusJáték labirintus.txt");
375            System.exit(-1);
376        }
377        try {
378            new LabirintusJáték(args[0]);
379        } catch(RosszLabirintusKivétel rosszLabirintusKivétel) {
380            System.out.println(rosszLabirintusKivétel);
381        }
382    }
383 }
```

5.3. ábra. labprint



5.4. ábra. labirintus

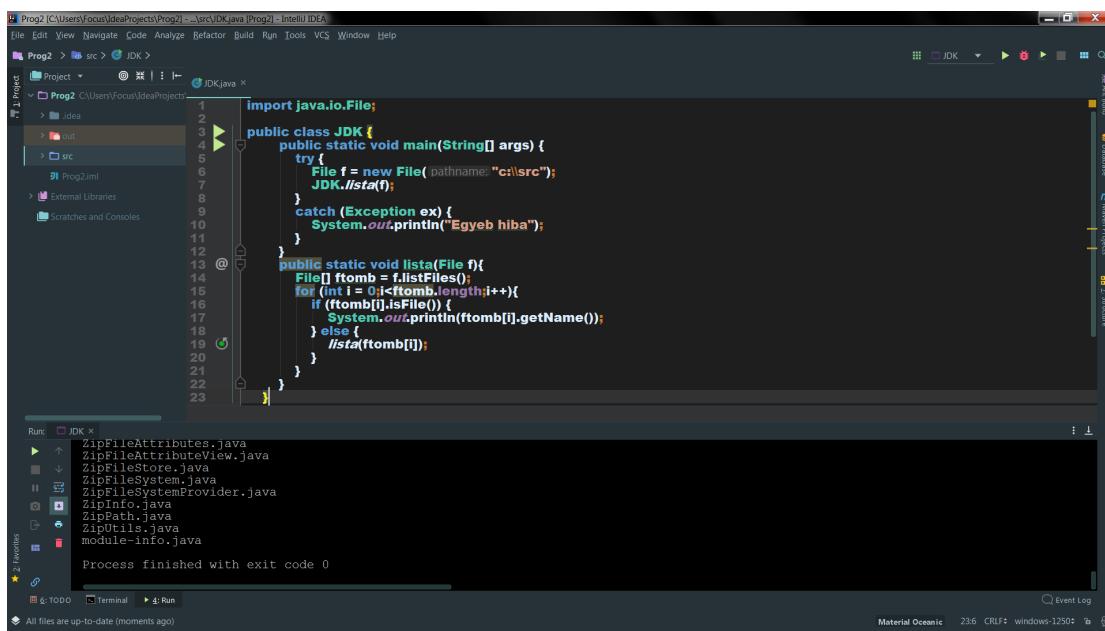
## 6. fejezet

# Helló, Stroustrup!

### 6.1. JDK osztályok

Írunk olyan Boost C++ programot (indulj ki például a fénykardból) amely kilistázza a JDK összes osztályát (miután kicsomagoltuk az src.zip állományt, arra ráengedve)!

Létrehozok egy lista nevű eljárást ami egy for ciklus segítségével vizsgálja a c:\src könyvtárba másolt src.zip kitömörített tartalmát. Az ftomb fogja tartalmazni az aktuális fileokat. A f.listFiles() metódus segítségével adtuk át az értékeket. Vizsgáljuk egy if elágazással, hogy file e. Ha igen, akkor kiiratjuk, különben pedig meghívjuk rá a lista eljárását. Ekkor hívja meg magát rekurzívan.



The screenshot shows the IntelliJ IDEA interface with a Java file named 'JDK.java' open. The code defines a recursive class 'JDK' with a main method that lists files in the 'src' directory. It uses a try-catch block to handle exceptions, specifically catching 'Exception' and printing 'Egyeb hiba'. The 'list' method takes a 'File' object and prints its name if it's a file, otherwise it calls itself recursively on each file in the list.

```
import java.io.File;
public class JDK {
    public static void main(String[] args) {
        try {
            File f = new File("c:\\src");
            JDK.lista(f);
        } catch (Exception ex) {
            System.out.println("Egyeb hiba");
        }
    }
    public static void lista(File f){
        File[] ftomb = f.listFiles();
        for (int i = 0;i<ftomb.length;i++){
            if (ftomb[i].isFile()){
                System.out.println(ftomb[i].getName());
            } else {
                lista(ftomb[i]);
            }
        }
    }
}
```

6.1. ábra. JDK

Szegedi Csaba tutorált engem, pontosabban a C++ kódot futtatta, és készített print screent, mert itt órán a gépen nem volt telepítve a boost, ezért nem tudtam futtatni.

```
#include <iostream>
#include <vector>
#include <string>
#include <stdio.h>
#include <boost/filesystem.hpp>
#include <boost/filesystem/fstream.hpp>
#define GetCurrentDir getcwd
using namespace std;
vector<string> searchRootFolders(vector<string> folders);
void readClasses(string path, vector<string> &classes);
string GetCurrentWorkingDir(void)
{
    char buff[FILENAME_MAX];
    GetCurrentDir(buff, FILENAME_MAX);
    string current_working_dir(buff);
    return current_working_dir;
}
int main(int argc, char const *argv[])
{
    vector<string> roots = {
        GetCurrentWorkingDir() + "/" + "src"};
    vector<string> classes = searchRootFolders(roots);
    cout << endl
        << "-----Printing java classes-----" << endl;
    for (auto &i : classes)
    {
        cout << i << endl;
    }
    cout << "All in all there were " << classes.size() << " classes found\n";
    return 0;
}
void readClasses(boost::filesystem::path path, vector<string> &classes)
{
    if (is_regular_file(path))
    {
        std::string ext(".java");
        if (!ext.compare(boost::filesystem::extension(path)))
        {
            classes.push_back(path.string());
        }
    }
    else if (is_directory(path))
        for (boost::filesystem::directory_entry &entry : boost::filesystem::directory_iterator(path))
            readClasses(entry.path(), classes);
}
vector<string> searchRootFolders(vector<string> folders)
```

```

vector<string> classes;
for (const auto &path : folders)
{
    boost::filesystem::path root(path);
    readClasses(root, classes);
}
return classes;
}

```

```

csapi@csapi-ThinkPad-T430:~/Desktop/Prog2/book_Examples/5.kotet/5.kotet/jdk$ g++ jdk.cpp -o jdk -lboost_system -lboost_filesystem
csapi@csapi-ThinkPad-T430:~/Desktop/Prog2/book_Examples/5.kotet/5.kotet/jdk$ cd openjdk/
csapi@csapi-ThinkPad-T430:~/Desktop/Prog2/book_Examples/5.kotet/5.kotet/jdk/openjdk$ ./jdk

```

6.2. ábra. JDKC++command

```

/home/csapi/Desktop/Prog2/book_Examples/5.kotet/jdk/openjdk/src/jdk.accessibility/share/classes/com/sun/java/accessibility/util/EventQueueMonitor.java
/home/csapi/Desktop/Prog2/book_Examples/5.kotet/jdk/openjdk/src/jdk.accessibility/share/classes/com/sun/java/accessibility/util/AWTEventMonitor.java
/home/csapi/Desktop/Prog2/book_Examples/5.kotet/jdk/openjdk/src/jdk.accessibility/share/classes/com/sun/java/accessibility/util/TopLevelWindowListener.java
/home/csapi/Desktop/Prog2/book_Examples/5.kotet/jdk/openjdk/src/jdk.accessibility/share/classes/com/sun/java/accessibility/util/AccessibilityListenerList.java
/home/csapi/Desktop/Prog2/book_Examples/5.kotet/jdk/openjdk/src/jdk.accessibility/share/classes/com/sun/java/accessibility/util/Translator.java
/home/csapi/Desktop/Prog2/book_Examples/5.kotet/jdk/openjdk/src/jdk.accessibility/share/classes/com/sun/java/accessibility/util/TopLevelWindowMulticaster.java
/home/csapi/Desktop/Prog2/book_Examples/5.kotet/jdk/openjdk/src/jdk.accessibility/share/classes/com/sun/java/accessibility/util/package-info.java
/home/csapi/Desktop/Prog2/book_Examples/5.kotet/jdk/openjdk/src/jdk.accessibility/share/classes/com/sun/java/accessibility/util/GUIInitializedMulticaster.java
/home/csapi/Desktop/Prog2/book_Examples/5.kotet/jdk/openjdk/src/jdk.accessibility/share/classes/module-info.java
/home/csapi/Desktop/Prog2/book_Examples/5.kotet/jdk/openjdk/src/jdk.accessibility/windows/classes/com/sun/java/accessibility/internal/AccessBridge.java
/home/csapi/Desktop/Prog2/book_Examples/5.kotet/jdk/openjdk/src/jdk.accessibility/internal/ProviderImpl.java
/home/csapi/Desktop/Prog2/book_Examples/5.kotet/jdk/openjdk/src/jdk.management.agent_unix/classes/jdk/internal/agent/FileSystemImpl.java
/home/csapi/Desktop/Prog2/book_Examples/5.kotet/jdk/openjdk/src/jdk.management.agent/share/classes/jdk/internal/agent/AgentConfigurationError.java
/home/csapi/Desktop/Prog2/book_Examples/5.kotet/jdk/openjdk/src/jdk.management.agent/share/classes/jdk/internal/agent/spi/AgentProvider.java
/home/csapi/Desktop/Prog2/book_Examples/5.kotet/jdk/openjdk/src/jdk.management.agent/share/classes/jdk/internal/agent/Agent.java
/home/csapi/Desktop/Prog2/book_Examples/5.kotet/jdk/openjdk/src/jdk.management.agent/share/classes/jdk/internal/agent/ConnectorAddressLink.java
/home/csapi/Desktop/Prog2/book_Examples/5.kotet/jdk/openjdk/src/jdk.management.agent/share/classes/sun/management/jdp/JdpPacketWriter.java
/home/csapi/Desktop/Prog2/book_Examples/5.kotet/jdk/openjdk/src/jdk.management.agent/share/classes/sun/management/jdp/JdpController.java
/home/csapi/Desktop/Prog2/book_Examples/5.kotet/jdk/openjdk/src/jdk.management.agent/share/classes/sun/management/jdp/JdpGenericPacket.java
/home/csapi/Desktop/Prog2/book_Examples/5.kotet/jdk/openjdk/src/jdk.management.agent/share/classes/sun/management/jdp/JdpBroadcaster.java
/home/csapi/Desktop/Prog2/book_Examples/5.kotet/jdk/openjdk/src/jdk.management.agent/share/classes/sun/management/jdp/JdpException.java
/home/csapi/Desktop/Prog2/book_Examples/5.kotet/jdk/openjdk/src/jdk.management.agent/share/classes/sun/management/jdp/JdpPacket.java
/home/csapi/Desktop/Prog2/book_Examples/5.kotet/jdk/openjdk/src/jdk.management.agent/share/classes/sun/management/jdp/JdpPacketReader.java
/home/csapi/Desktop/Prog2/book_Examples/5.kotet/jdk/openjdk/src/jdk.management.agent/share/classes/sun/management/jdp/JdpJmxPacket.java
/home/csapi/Desktop/Prog2/book_Examples/5.kotet/jdk/openjdk/src/jdk.management.agent/share/classes/sun/management/jdp/package-info.java
/home/csapi/Desktop/Prog2/book_Examples/5.kotet/jdk/openjdk/src/jdk.management.agent/share/classes/sun/management/jmxremote/ConnectorBootstrap.java
/home/csapi/Desktop/Prog2/book_Examples/5.kotet/jdk/openjdk/src/jdk.management.agent/share/classes/sun/management/jmxremote/LocalRMIServerSocketFactory.java
/home/csapi/Desktop/Prog2/book_Examples/5.kotet/jdk/openjdk/src/jdk.management.agent/share/classes/sun/management/jmxremote/SingleEntryRegistry.java
/home/csapi/Desktop/Prog2/book_Examples/5.kotet/jdk/openjdk/src/jdk.management.agent/windows/classes/jdk/internal/agent/FileSystemImpl.java
All in all there were 20649 classes found
csapi@csapi-ThinkPad-T430:~/Desktop/Prog2/book_Examples/5.kotet/jdk/openjdk$ 

```

6.3. ábra. JDKC++result

## 6.2. Másoló-mozgató szemantika

Kódcsipeteken (copy és move ctor és assign) keresztül vesd össze a C++11 másoló és a mozgató szemantikáját, a mozgató konstruktort alapozd a mozgató értékkadásra!

A feladat megoldásához egy feladatot használtam fel az LZWBinFa-t. Arra kell figyelnünk a feladat során, ha alapértelmezetten definiálunk egy osztályt, akkor a másolás során csak a memóriacím adódik át, és az

érték nem. Ebben az esetben ha módosítjuk az egyik értéket, akkor a másik érték is módosulni fog. Ez elég sok kellemetlenséget tud okozni. Ennek kiküszöböléseképpen vannak különböző szemantikák. Ebben a példában az általunk megírt másoló konstruktur akkor hívódik meg, ha az objektumunk konstruktoraiba egy bal oldali referenciájú azonos típusú objektumot írunk. Ezután az új objektumnak átadjuk a régi értékét, de csak azt. Így ha az egyiket módosítjuk, akkor a másik nem fog változni. Ugyanez vonatkozik a mozgató szemantikára is. Csak ott a memoriában lefoglalt helyet "átmozgatjuk" az új változónak a benne tárolt értékekkel együtt, és utána a régit töröljük, ezzel felszabadítva a helyet az új adatnak. Ez a kódban itt jelenik meg:

```
class LZWBinFa {
public:

    LZWBinFa () :fa (gyoker = new Csomopont ()) {}
    ~LZWBinFa () {
        std::cout << "LZWBinFa dtor" << std::endl;
        szabadit (gyoker);
    }

    LZWBinFa ( const LZWBinFa & regi ) {
        std::cout << "LZWBinFa copy ctor" << std::endl;
        gyoker = masol(regi.gyoker, regi.fa);
    }

    LZWBinFa (LZWBinFa && regi) {
        std::cout << "LZWBinFa move ctor" << std::endl;
        gyoker = nullptr;
        *this = std::move(regi);
        // gyoker = regi.gyoker;
        // regi.gyoker = nullptr;
    }

    LZWBinFa& operator=(LZWBinFa && regi) {
        std::cout << "LZWBinFa move assign" << std::endl;
        std::swap(gyoker, regi.gyoker);
        return *this;
    }
}
```

## 6.3. Összefoglaló

Az előző 4 feladat egyikéről írj egy 1 oldalas bemutató „esszé szöveget!

Az osztály definíciója 2 fő alkotóelemből áll: az osztály deklarációjából, és az osztály törzséből. Az osztály deklaráció az osztály kódjának az első sora. Minimálisan az osztály deklaráció meghatározza az osztály nevét. Az osztálytörzs az osztály deklarációt követi, és kapcsos zárójelek között áll. Az osztály törzs tartalmazza minden a kódot, amely hozzájárul az osztályból létrehozott objektumok életciklusához: konstruktork, új objektumok inicializálására, változó deklarációk, amelyek megadják az osztály és objektumának

állapotát, és eljárásokat az osztály és objektumai viselkedésének meghatározására. Az osztály három tagváltozót definiál az osztálytörzsön belül. A következő konsztruktor a tagváltozók kezdőértékeinek beállítására biztosít lehetőséget. Megadhatunk egy osztályt egy másik osztály tagjaként. Egy ilyen osztályt beágyazott osztálynak hívunk. A beágyazott osztályokat arra használjuk, hogy kifejezzük és érvényesítsük két osztály között a kapcsolatot. Megadhatunk egy osztályt egy másik osztályon belül, hogyha a beágyazott osztálynak a magába foglaló osztályon belüli környezetben van értelme. Pl. a szövegkurzornak csak a szövegkomponensen belüli környezetben van értelme. A beágyazó osztály tagjaként a beágyazott osztály kiváltságos helyzetben van. Korlátlan hozzáféréssel rendelkezik a beágyazó osztályok tagjaihoz még akkor is, hogy ha azok privátként vannak deklarálva. Azonban ez a speciális kiváltság nem minden speciális. A hozzáférést biztosító tagok korlátozzák a hozzáféréseket az olyan osztálytagokhoz, amelyek a beágyazó osztályon kívül esnek. A beágyazott osztály a beágyazó osztályon belül található, ebből kifolyólag hozzáférhet a beágyazó osztály tagjaihoz. Mint ahogyan más tagokat is, a beágyazott osztályokat is statikusként, avagy nem statikusként lehet deklarálni, ezért ezeket pontosan így is hívják: statikus beágyazott osztály. A nem statikus beágyazott osztályokat belső osztályoknak hívjuk. Ahogy a statikus metódusok és változók esetén, amelyeket mi osztálymetódusoknak és változóknak hívunk, a statikus beágyazott osztályt a beágyazó osztályával kapcsoljuk össze. Ahogy az osztálymetódusok, a statikus beágyazott osztályok sem hivatkozhatnak közvetlenül olyan példányváltozóra vagy metódusakra, amely az Ő beágyazó osztályában van megadva. A példánymetódusok és változók esetén egy belső osztály az Ő beágyazó osztályának a példányával kapcsolódik össze, és közvetlen hozzáférése van annak az objektumnak a példányváltozóhoz és metódusaihoz. Mivel egy belső osztályt egy példánnyal társítanak, ezért önmaga nem definiálhat akármilyen statikus tagot. Ezeket az osztályokat listázza ki a programunk.

## 7. fejezet

# Helló, Gödel!

### 7.1. Gengszterek

Gengszterek rendezése lambdával a Robotautó Világbajnokságban <https://youtu.be/DL6iQwPx1Yw> (8:05-től)

A lambdás rendezés minden paraméteres megoldást takar. Itt a függvényhívás úgy történik, hogy először a paramétereket adom meg, ami jelen esetben két gangster objektum, majd a függvény törzsét. Így akár egy paramétereirek számított parancsról is lehetne beszélni. Itt az std::sort függvénye szolgál rendezésre, ahol megadom, hogy mettől meddig rendezze a vektort, majd megadom a visszatérési értéknél azt a vektort, melyet a rendezéssel hozok létre. A dst(cop, x.to) függvény megadja a paraméterként megadott gangster és rendor távolságát. Majd ezt megcsináljuk az y gangsterre. Ha közelebb van a cop-hoz az x, akkor igazzal tér vissza a függvény.

```
int main{
/*reading all gangsters into a vector*/
int idd {0};
unsigned f, t, s, ret = 0;
int n {0};
int nn {0};
std::vector<unsigned> gangsters;
while (std::sscanf (data+nn, "<OK \d \u \u \u>\n", &idd, &f, &t, &s, &n ) ←
    xx 4) {
nn +x n;
gangsters.push_back ( f );
}
std::sort (gangsters.begin(), gangsters.end(), [this, cop] (unsigned x, ←
    unsigned y)
{
return dst(cop, x) < dst (cop, y);
} );
if (gangsters.size() > 0)
return gangsters[0];
else
```

```
return 0;  
}
```

## 7.2. C++11 Custom Allocator

<https://prezi.com/jvvbytkwgsxj/high-level-programming-languages-2-c11-allocators/> a CustomAlloc-os példa, lásd C forrást az UDPORG repóban!

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

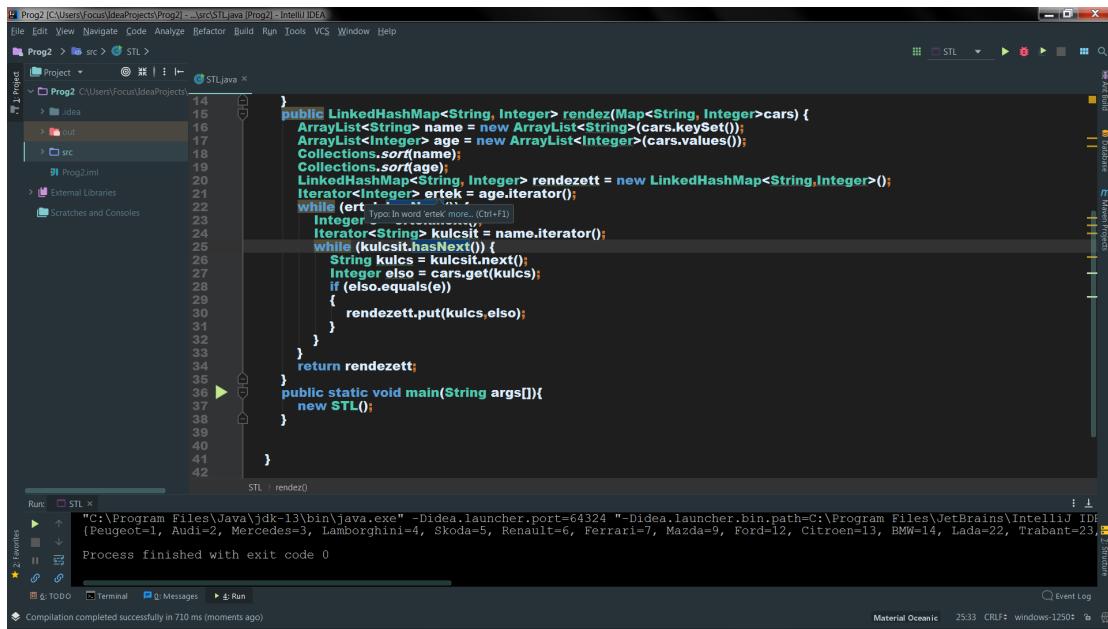
## 7.3. STL map érték szerinti rendezése

Például: <https://github.com/nbatfai/future/blob/master/cs/F9F2/fenykard.cpp#L180>

Létrehozok egy konstruktort, amiben implementállok egy map adatszerkezet. Feltöltöm az autók neveivel, (ezek a kulcsok) majd megadom mennyi idősek (ezek az értékek). Meghívom a rendez() metódusomat, majd kiíratom a rendezett mapot. Az rendez() függvény egy linkedHashMap-et ad vissza, és paraméterként vár egy Map-ot, aminek a kulcsa String, és értéke Integer. Különválasztom két ArrayList-be az autókat és korukat, majd a Collections.sort metódussal növekvő sorrendbe rendezem. Létrehozok egy hashmapot, amiben a rendezett értékeket rakkom, majd a függvény visszatér vele. Egy iterátort csinálok az age-re és egy while ciklus segítségével végigmegyek az age ArrayList-en. Egy integer objektumba az iterátor következő elemét lekérjük, majd csinálok még egy iterátort az autók nevére és egy belső while ciklussal azon is végigmegyek. Ha az elso hasonlít az e objektumra, akkor a hashMap-be berakjuk a kulcs és elso változókat. A végén visszaadjuk a rendezés eredményét. A main-be példányosítom a fentebb implementált STL konstruktort.

```
import java.util.*;  
  
public class STL {  
    STL() {  
        Map<String, Integer> cars = new HashMap<String, Integer>();  
        cars.put("Audi", 2); cars.put("Skoda", 5); cars.put("BMW", 14);  
        cars.put("Lada", 22); cars.put("Wartburg", 25); cars.put("←  
            Trabant", 23);  
        cars.put("Mercedes", 3); cars.put("Renault", 6); cars.put("←  
            Peugeot", 1);  
        cars.put("Citroen", 13); cars.put("Ferrari", 7); cars.put("←  
            Mazda", 9);  
        cars.put("Lamborghini", 4); cars.put("Polski", 30); cars.put("←  
            Ford", 12);  
        cars = rendez(cars);  
    }  
  
    void rendez(Map<String, Integer> m) {  
        ArrayList<String> autok = new ArrayList<String>();  
        ArrayList<Integer> korok = new ArrayList<Integer>();  
        for (Map.Entry<String, Integer> entry : m.entrySet()) {  
            autok.add(entry.getKey());  
            korok.add(entry.getValue());  
        }  
        Collections.sort(autok);  
        Collections.sort(korok);  
        int i = 0;  
        while (i < autok.size()) {  
            String a = autok.get(i);  
            Integer k = korok.get(i);  
            Iterator<Map.Entry<String, Integer>> it = m.entrySet().iterator();  
            while (it.hasNext()) {  
                Map.Entry<String, Integer> entry = it.next();  
                if (entry.getKey().equals(a)) {  
                    entry.setValue(k);  
                    it.remove();  
                }  
            }  
            i++;  
        }  
    }  
}
```

```
        System.out.println(cars);
    }
    public LinkedHashMap<String, Integer> rendez(Map<String, Integer> ↪
        cars) {
        ArrayList<String> name = new ArrayList<String>(cars.keySet());
        ArrayList<Integer> age = new ArrayList<Integer>(cars.values());
        Collections.sort(name);
        Collections.sort(age);
        LinkedHashMap<String, Integer> rendezett = new LinkedHashMap< ↪
            String, Integer>();
        Iterator<Integer> ertek = age.iterator();
        while (ertek.hasNext()) {
            Integer e = ertek.next();
            Iterator<String> kulcsit = name.iterator();
            while (kulcsit.hasNext()) {
                String kulcs = kulcsit.next();
                Integer also = cars.get(kulcs);
                if (also.equals(e))
                {
                    rendezett.put(kulcs,also);
                }
            }
        }
        return rendezett;
    }
    public static void main(String args[]){
        new STL();
    }
}
```



7.1. ábra. STL

## 7.4. Alternatív Tabella rendezése

Mutassuk be a [https://progpater.blog.hu/2011/03/11/alternativ\\_tabella](https://progpater.blog.hu/2011/03/11/alternativ_tabella) a programban a java.lang Interface Comparable T szerepét!

Célja a csapatok teljesítményére vonatkozó rendezés, amit nem csak a vereségek, döntetlenek, és győzelmek aránya alapján határoz meg, hanem figyelembe veszi azt is, hogy erősebb csapat ellen többet ér a győzelem. A `Comparable<T>` egy interface amelyet az implements kulcsszóval lehet implementálni az osztályhoz. Ez az interface egy függvényt definiál, a public int compareTo(To)-t. Ez a függvény összehasonlítja a paraméterként kapott T objektumot egy másik objektummal. Visszatérésként kapunk egy számot a két objektum relációjától függően. Három fajta lehet. Az egyik a kisebb, akkor visszatér negatívvval. A másik a hasonló, akkor visszaad nullát. És a harmadik a nagyobb, akkor pozitívvval tér vissza. Ahogy látható a kódból is implementáltuk a Comparable-t az Csapat osztályra. Majd kifejtettük a compareTo(Csapat csapat). Itt az osztály attribútumaként megadott ertek-et hasonlíttuk össze a paraméterként átadott Csapat objektum értékével. A visszatérés a fentebb leírt módon történik.

```

class Csapat implements Comparable<Csapat> {

    protected String nev;
    protected double ertek;
}

```

```
public Csapat(String nev, double ertek) {
    this.nev = nev;
    this.ertek = ertek;
}

public int compareTo(Csapat csapat) {
    if (this.ertek < csapat.ertek) {
        return -1;
    } else if (this.ertek > csapat.ertek) {
        return 1;
    } else {
        return 0;
    }
}
```

## 8. fejezet

# Helló, !

### 8.1. OOCWC Boost ASIO hálózatkezelése

Mutassunk rá a scanf szerepére és használatára! <https://github.com/nbatfai/robocaremulator/blob/master/justine/ro>

A sscanf függvény az std könyvtár egy tagja, amely adatot olvas a standard bemenetről, majd eltárolja azokat a paraméter típusa szerint egy megadott helyre amit további argumentumként kap meg a függvény. A % jel után írt karakter mindenkor a beolvadt típust, míg a & jel után a helyet jelenti. A sscanf egy while ciklusban helyezkedik el, és a szerepe, hogy addig dolgozza fel az adott formázott string-ből az adatokat amíg, megfelel a Gangster kritériumának, amelyhez 4 szám szükséges. A %n az olvasott karakterek számát rögzíti. Az érdekesebb része a programnak, hogy a while ciklusban az "nn+=n" számat növeljük, hogy képesek legyünk tényleges while ciklust használni, nem csak az első inputot feldolgozni, és a többi potenciális Gangster-rel nem törödni. Megjegyezném azt is, hogy a sscanf visszatéríti a sikeresen feltöltött változók számát. Itt jön be felhasználásra az az adat, hogy 4 argumentum szükséges a Gangster típushoz.

```
while ( std::sscanf ( data+nn, "<OK %d %u %u %u>%n", &idd, &f, &t, &s, &n ) == 4 )
{
    nn += n;
    gangsters.push_back ( Gangster {idd, f, t, s} );
}
```

### 8.2. SamuCam

Mutassunk rá a webcam (pl. Androidos mobilod) kezelésére ebben a projektben: <https://github.com/nbatfai/Samu>

A SamuCam konstruktörét a szélesség és magasság és egy string bemenet adják. Ennek segítségével elindítja a videostreamet. Ehhez a header fájlban definiált cv::videostreamet használja, így nyitja meg magát a streamet és elvégzi a beállításokat. Majd erre ráengedi a CascadeClassifier-t, amit egy XML alapján hozott létre, és ezáltal válik lehetővé az arcfelismerés. A load függvény segítségével megnyit egy classifier-t, ami egy emberi arcot ír majd le. Amíg a videókamera használatban van (50 ms-ként történő ellenőrzéssel),

folyamatosan kéri a frame-eket 80 ms-ként, melyeket újraméretez és szürkeárnyalatosra fest, majd elindítja az arcfelismerést soronként és oszloponként, és a megfelelő képpontokat adja vissza, így a kapott vektorokkal egy téglalapba zárható az arc. Ha nem ismer fel arcot, akkor egyszerűen a webcam képét adja vissza. Az openCV felelős a videó felvételéről, a videókból a képek csinálásáért, a képek formázásáért (kicsinyítés, Grayscale), és az arc felismeréséért. A QT az univerzális grafikus felület.

### 8.3. BrainB

Mutassuk be a Qt slot-signal mechanizmust ebben a projektben: <https://github.com/nbatfai/esporttalents-search>

Tehetségkutató játék ami azt nézi, hogy meddig tudjuk rajta tartani a kurzort a középen lévő ponton. Minél több új játékos keletkezik, a játék annál nehezebb lesz. A Qt(slot signal) jel rés mechanizmusa segítségével lekezeljük a Kilépés gomb megnyomását. Qt-ban a slot-signal mechanizmust objektumok közti kommunikációra lehet használni. A slot-signal mechanizmus egy Qt által nyújtott lehetőség, hogy egy megadott eseményre egy úgynevezett Signal keletkezik, A slot egy olyan függvény amit egy bizonyos signal bekövetkezte esetén kell meghívni. Az egeret lenyomva tartva a Samu Entropy körén egy signalt idéz elő, mely kiváltja az újabb négyzetek megjelenését a mozgás gyorsulása a Slot részben. Hasznos link ehhez a példához: <https://doc.qt.io/archives/3.3/signalsandslots.html> A slot signal mechanizmus: A signal mechanizmust az osztályoknál kell keresni, a Q\_Objectet, ami egy meta. Ugyebár a meta object compiler az kezeli a Qt-s C++ kiterjesztéseket. Ez olvassa a header fájlokat, és ha több osztály tartalmazza ezt a macrot, akkor előállít egy olyan forrást, amely tartalmazza ezeknek a metaobjektum kódját. Ez a metaobjektum kód szükséges a jel(signal) és a slot mechanizmushoz. Mindemellett a futási időben előfordulhat típusinformációkhoz. A signal-oknál 2 függény van.

```
void heroesChanged ( const QImage &image, const int &x, const int &y );
void endAndStats ( const int &t );
```

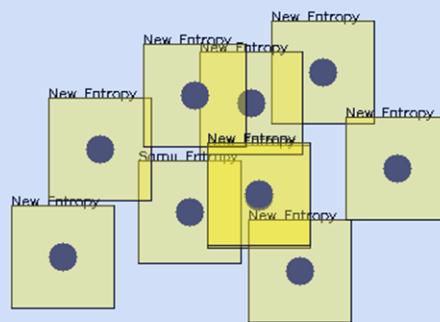
Következtethetünk a metódusnevekből, a signal-ok lefutására. Az első ugyebár az objektumok mozgásakor, mikor az entropy-k koordinátája változik, a másodikban pedig az eredmény kiíratására a futás végén. Ha signal-t hívunk, akkor az emit kulcsszót kell alkalmazni. A heroesChanged-et a thread headerben találjuk a kirajzolásnál. Az end stat-ot a futtatásnál a thread.cpp-ben.

```
emit heroesChanged ( dest, heroes[0].x, heroes[0].y );
emit endAndStats ( endTime );
```

A signal-okat bekapcsolni a connect-el kell a slot-okhoz. A slot-okat akkor hívjuk meg ha egy signal bekövetkezik. Ezzel kerülhetjük el egy másik osztály függvény meghívását. A connect-eket a win.cpp-ben találjuk. A signal-ok és a slot-ok felépítésének meg kell egyezni.

```
connect ( brainBThread, SIGNAL ( heroesChanged ( QImage, int, int ) ), this ←
, SLOT ( updateHeroes ( QImage, int, int ) ) );
```

```
connect ( brainBThread, SIGNAL ( endAndStats ( int ) ), this, SLOT ( ←  
    endAndStats ( int ) ) );
```



8.1. ábra. BrainB

## 9. fejezet

# Helló, Schwarzenegger!

### 9.1. Port scan

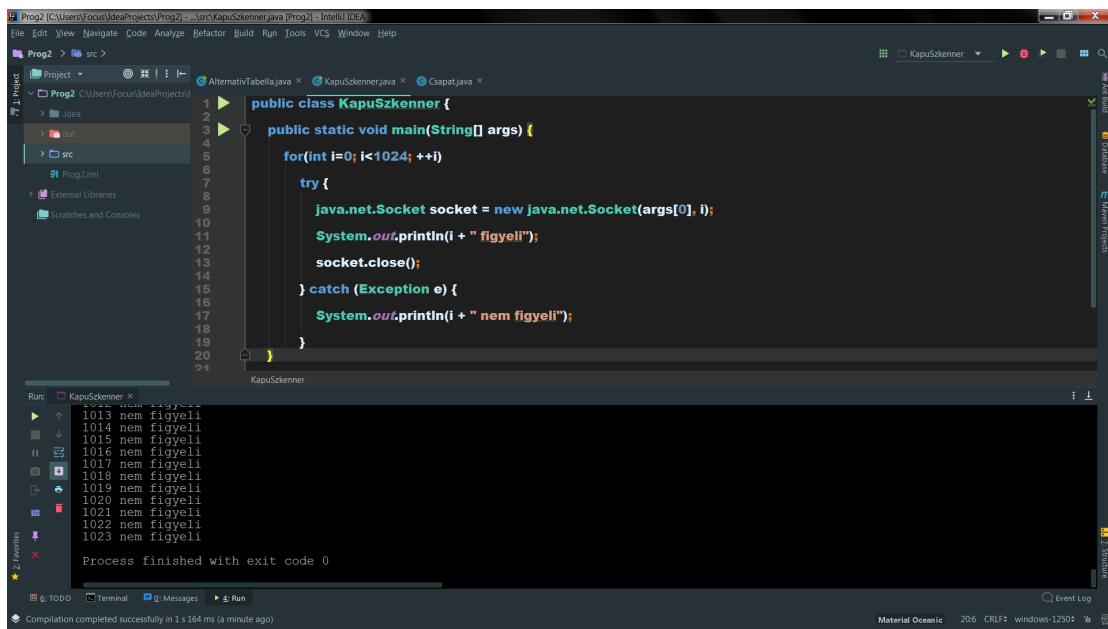
Mutassunk rá ebben a port szkennelő forrásban a kivételkezelés szerepére! <https://www.tankonyvtar.hu/hu/tartalom/tanitok-javat/ch01.html#id527287>

A kivételkezelés egy programozási mechanizmus, melynek célja a program futását szándékosan vagy nem szándékos módon megszakító esemény (hiba) vagy utasítás kezelése. Programunkban a függvény 0-1024-ig végignézi a portokat, hogy melyiken tud létrehozni kapcsolatot. A socket létrehozásakor 2 argumentumú konstruktort használunk. Az első a hostnak a neve, a második pedig a portnak a száma. Ha tudunk létesíteni kapcsolatot, akkor egyszerűen kiírjuk, hogy az a port figyeli, és le is zárjuk azt, ha sikertelen a próbálkozás, akkor pedig azt írja ki, hogy sikertelen. Ha találna egy ilyen portot, az azt jelentené, hogy "sebezhető" az eszközünk, vagyis könnyű lenne a privát adatokhoz hozzáférni. A programban azért szükséges a kivételkezelés, mert ha nem lehet kapcsolódni az adott porthoz, vagyis nem figyeli, akkor kivételt fog dobni, és leáll a program. Ez elkerülhető, ugyanis a kivételkezelés miatt ilyenkor a catch blokk fog végrehajtódni. Jelen esetünkben kiírásra kerül, hogy az adott portot semmi sem figyeli. Viszont ha sikeres a kapcsolódás, akkor a catch rész nem fog lefutni, vagyis a program jelezni fogja, hogy a portot egy folyamat figyeli.

```
public class KapusZkenner {

    public static void main(String[] args) {
        for(int i=0; i<1024; ++i)
            try {
                java.net.Socket socket = new java.net.Socket(args[0], i);
                System.out.println(i + " figyeli");
                socket.close();
            } catch (Exception e) {
```

```
System.out.println(i + " nem figyeli");  
}  
}  
}
```

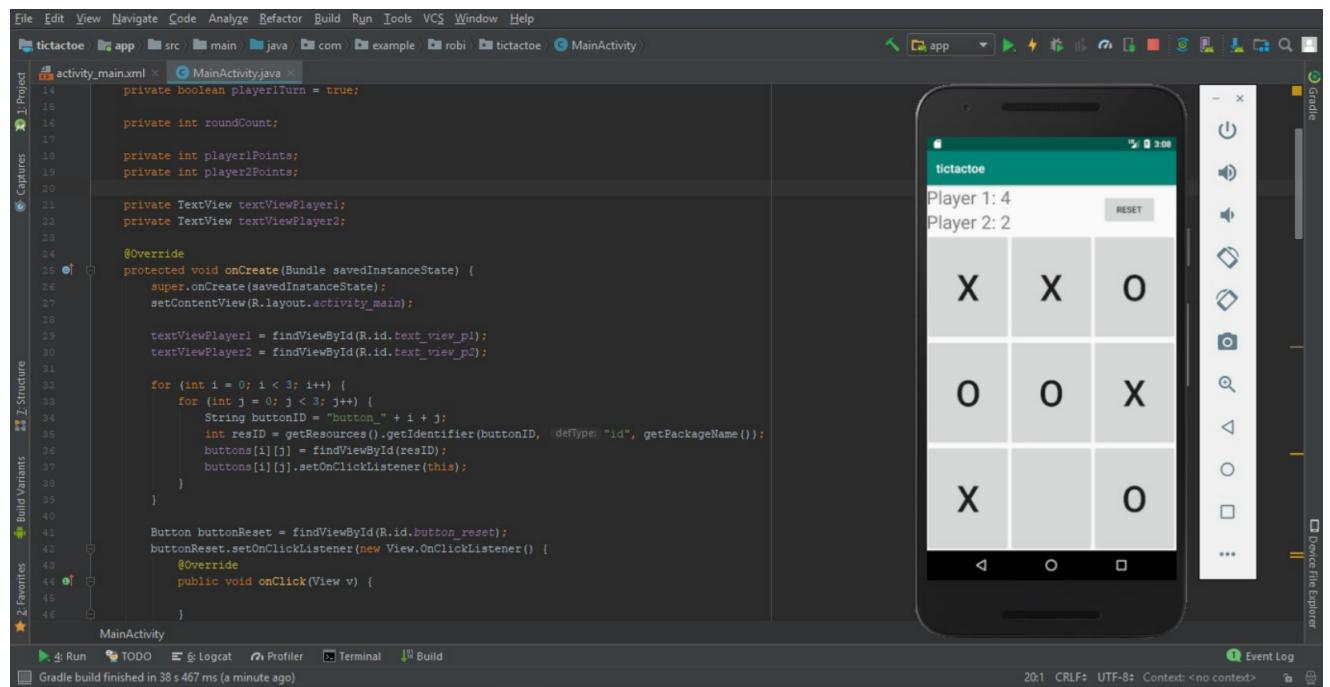


9.1. ábra. KapuSzkenner

## 9.2. Android Játék

Írunk egy egyszerű Androidos „játékot”! Építkezzünk például a 2. hét „Hello, Android!” feladatára!

Egy működő androidos játéket kellett írni, amit én ezt a alábbi youtube tutorial alapján valósítottam meg: <https://www.youtube.com/watch?v=apDL78MFR3o> A játék egy tictactoe játék, ami a játék szabályai alapján működik, számolja a játékosok pontjait, valamint egy reset funkció is bele van építve, ami az eddig összegyűjtött pontokat nullázza.



9.2. ábra. TicTacToe

# 10. fejezet

## Helló, Calvin!

### 10.1. MNIST

Az alap feladat megoldása, +saját kézzel rajzolt képet is ismerjen fel, [https://progpater.blog.hu/2016/11/13/hello\\_sbol](https://progpater.blog.hu/2016/11/13/hello_sbol) Háttérként ezt vetítsük le: <https://prezi.com/0u8ncvvoabcr/no-programming-programming/>

A python alapvetően egy objektum orientált, procedurális imperatív nyelv, míg a Tensorflow az adatfolyam programozást képviseli. A TensorFlow egy szoftverkönyvtár, gépi tanulási algoritmusok leírására és végrehajtására. A TensorFlow rendszerben kifejlesztett számítások változatlanul vagy csekély változtatással végrehajthatók nagyon eltérő hardver eszközökön a mobil telefonuktól és tabletektől kezdve, grafikus kártyákon át, sok számítógépből álló elosztott számítógéprendszerekig. A TensorFlow roppant flexibilis, nagyon széles körű algoritmusok megvalósítására alkalmas. Alkalmazható a számítástudomány más területein is. A TensorFlow számítást egy irányított gráf írja le. Adatáramlás a gráf élei mentén történik. A TensorFlow gráfban minden egyik csúcs egy műveletet reprezentálhat és minden egyik csúcsnak lehet nulla vagy több inputja, ugyanígy nulla vagy több outputja. A gráf normál elei 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 célokot szolgálnak. A program futtatásához szükség volt a TensorFlow könyvtárra, amit a Google fejlesztett ki a másfajta numerikus számításokhoz. A TensorFlow egy opensource gépi tanuláshoz használt szoftverkönyvtár. Továbbá kellett még egy 28x28-as kép, amit a program feldolgozva megmondta hogy hányas számot jelenít meg. Szükségeim volt a Python3-dev-re és a Python3-pip-ra. A pip segítségével megnéztük hogy az importált könyvtárak megvannak-e. Ha nincsenek, akkor leszedte nekünk. Importálni kellett a szükséges könyvtárakat. Keras-t használtam amit a következőképpen töltöttem le.

```
import keras
from keras.datasets import fashion_mnist
from keras.layers import Dense, Activation, Flatten, Conv2D, MaxPooling2D
from keras.models import Sequential
from keras.utils import to_categorical, np_utils
from PIL import Image
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
import os
```

Betöltöttem az adatbázist, melyet a következő parancssal tettek.

```
(train_X,train_Y), (test_X,test_Y) = tf.keras.datasets.mnist.load_data()
```

Ezek a sorok a vektorok elkészítéséért felelősek, melyeket a reshape függvény segítségével hozunk megfelelő formára. Azaz 28-szor 28 db elemet tartalmazó vektorra, ez a 2. és a 3. paramétere a függvénynek. Az első paraméter a -1, ami azt jelenti, hogy minden egyes tagra értelmezni kell. Alapvetően ide az kerülne hány darabot kell konvertálni (ha az első 10-et akarjuk, akkor egy tízest kellett írni). A 4. paraméter pedig a kép színcsatornájával köthető össze, mivel mi grayscale képeket használunk, ezért egyes kerül ide, de ha színes azaz RGB képeket használnánk, akkor a hármas kerülne ide.

```
train_X = train_X.reshape(-1, 28,28, 1)
test_X = test_X.reshape(-1, 28,28, 1)
```

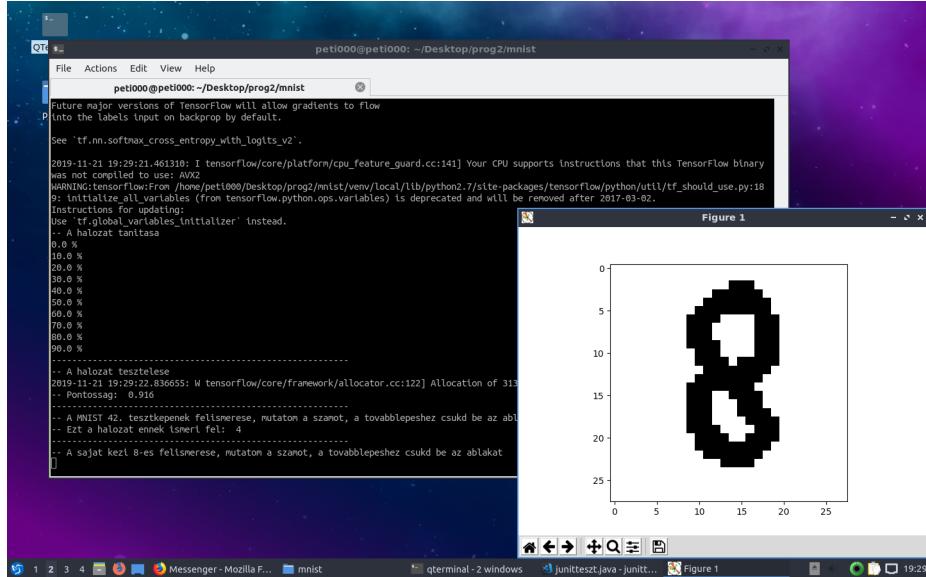
A következő kódrészlet felelős, hogy a tanulás minél gyorsabban végbemenjen. Az egyes pixelek értékeinek módosítása.

```
train_X = train_X.astype('float32')
test_X = test_X.astype('float32')
train_X = train_X / 255
test_X = test_X / 255
```

Majd felépítjük a szekvenciális modellt. A rétegek egymásra helyezése az add() függvény segítségével fog történni. Első paraméterként a neuronok számát kell megadni (64), a második paraméter a detektor (3x3), a harmadik paramétere az input\_shape, amik 28 X 28-as grayscale-s képek lesznek. Ezután a következő sorban aktiválunk egy relu-t (Rectified Linear Unit). Utána megadjuk, hogy mennyi adat kerüljön feldolgozásra egyszerre, ezt a pool\_size-al tudjuk elérni. Ez egy vertikális és horizontális értéket vár paraméterként. Végül a compile függvénytel megindítjuk a tanulási folyamatot.

```
model = Sequential()
model.add(Conv2D(64, (3,3), input_shape=(28, 28, 1)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Conv2D(64, (3,3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Flatten())
model.add(Dense(64))
model.add(Dense(10))
```

```
model.add(Activation('softmax'))
model.compile(loss=keras.losses.categorical_crossentropy, optimizer=keras.optimizers.Adam(), metrics=['accuracy'])
```



10.1. ábra. mnist

## 10.2. CIFAR-10

Az alap feladat megoldása, +saját fotót is ismerjen fel, [https://progpater.blog.hu/2016/12/10/hello\\_samu\\_a\\_cifar-10\\_tf\\_tutorial\\_peldabol](https://progpater.blog.hu/2016/12/10/hello_samu_a_cifar-10_tf_tutorial_peldabol)

Ennek a feladatnak az elvégzésben segítségemre volt Harmati Norbert. A Tensorflow honlapjáról letöltöm a pip-et. Követem az ott leírtakat. Létrehozok egy virtual environment-et, és abban futtatom a cifar10\_train.py-t. Feladatunk hasonló az előzőhez, Csak itt tárgyakat, állatokat kell felismertetni számok helyett. Itt valószínű, hogy pontatlanabb lesz a neurális hálónk, mert nehezebb feladata lesz. A kód sokban hasonlít az előző feladatban lévőhöz. Egy másik adatbázisból dolgozunk.

```
(train_X, train_Y), (test_X, test_Y) = cifar10.load_data()
```

Itt a reshape fügvény, aminek a paraméterezésén változtattam. Most ugye 32\*32-es képeket kap és RGB színkódolásút, ezért 4. paraméterként egy hármast kapott.

```
train_X = train_X.reshape(-1, 32, 32, 3)
test_X = test_X.reshape(-1, 32, 32, 3)
```

A Dense réteg az előző réteg összes kimenetét továbbítja minden neuronjához, ebben az esetben a neuronok számát a 4-szeresére növeltük.

```
model.add(Dense(256))
```

Különbséget a Conv2D rétegen láthatunk. Itt a filterek (neuronok) számát a duplajára növeltük, és az input\_shape is változik a felismerendő képek miatt.

```
model.add(Conv2D(128, (3, 3), activation='relu', input_shape=(32, 32, 3)))
```

Itt külön létre kellett hoznunk, egy tömböt, amiben indexelve vannak ugye az adott kategóriában lévő képek, így használható a one\_hot kódolás, ami elengedhetetlen programunk működéséhez.

```
cifar_classes = ['airplane', 'automobile', 'bird', 'cat', 'deer', 'dog',  
'frog', 'horse', 'ship', 'truck']
```

Elindult a tanulási folyamat. Az első képet macskának ismerte fel.



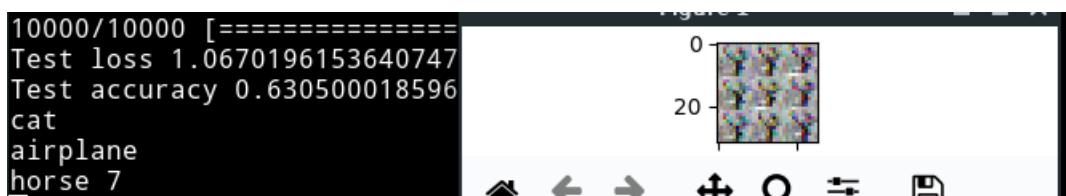
10.2. ábra. cat

A második képet repülőgépnek ismerte fel.



10.3. ábra. airplane

Végül a harmadik képet pedig egy lónak.



10.4. ábra. horse

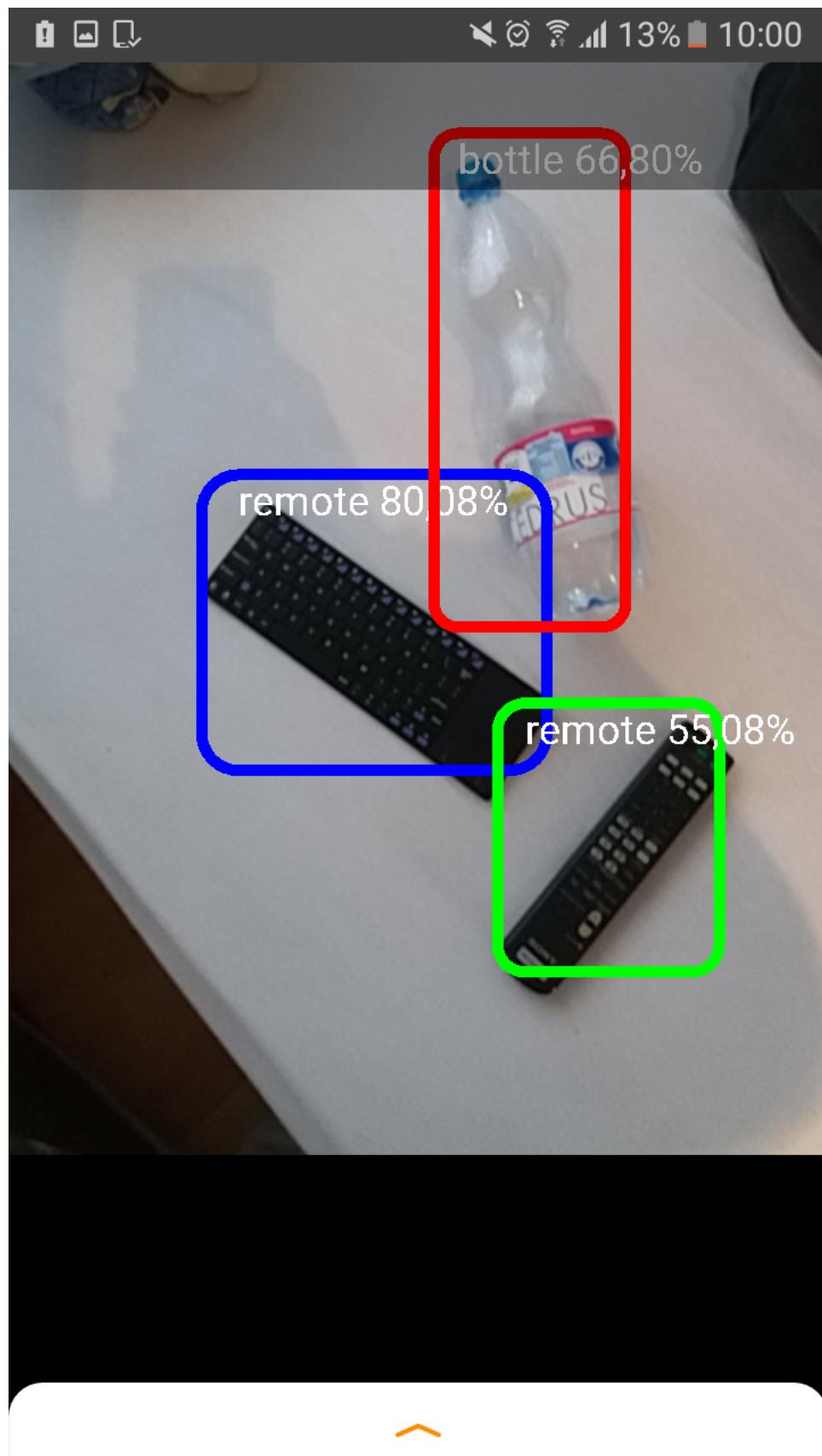
### 10.3. Android telefonra a TF objektum detektálója

Telepítsük fel, próbáljuk ki!

Az alkalmazást a Google Play Áruházból telepítettem. A program lényege hogy valós időben felismerje azokat a tárnyakat amit a kamera segítségével mutatunk a programnak. Tesztelgettem, hogy mit képes felismerni, illetve próbáltam találni pontokat ahol a program minden rosszul ismeri fel az objektumot. Sajnos közel sem tökéletes, néha teljesen másnak ismeri fel, mint amit a képen látunk (a lenti képen például az egérpadot egy bőröndnek nézte). Ha nincs nagyon világos, akkor nehezére esik a programnak felismerni a tárgyakat, amiket egyébként könnyen felismerne. Kis négyzettel jelzi amit éppen detektál, és odaírja, hogy minek ismerte fel százalékos pontossággal.



10.5. ábra. tfod1



10.6. ábra. tfod2

# 11. fejezet

## Helló, Berners-Lee!

### 11.1. C++ és Java összehasonlítás

C++: Benedek Zoltán, Levendovszky Tíhamér Szoftverfejlesztés C++ nyelven

Java: Nyékyné Dr. Gaizler Judit et al. Java 2 útikalauz programozóknak 5.0 I-II

A Java nyelv jelölésrendszerét figyelve sok minden vett át a C++ nyelvből, de vigyáztak a megbízhatóságra. C++-ban vannak mutatók és referenciai, viszont Java-ban csak referencia van. A Java és a C++ is objektumorientált, objektumok és ezek mintáinak tekinthető osztályokból épül fel. Az objektumorientált programozás olyan programozási paradigma, amely a programokat objektumokból építi fel. Legfontosabb alapelvei: öröklődés, egységezés, polimorfizmus. A program működése objektumok kommunikációját jelenti. Az objektumok az objektumorientált technológia alapjai. Két jellemzővel rendelkeznek: állapottal és viselkedéssel. Az állapotot egy vagy több változóval, a viselkedést az objektumhoz rendelt metódussal (függvényel) írjuk le. Az objektum egy elemre úgy tudunk hivatkozni, hogy az objektum neve után írunk egy pontot, és utána írjuk az elem nevét. Egy osztály mezőkből (változókból) és metódusokból. Bizonyos nyelvekben a metódusokat függvényeknek neveznek, bár nincs visszatérési értékük. A Java-ban és C++-ban is a main() függvény hívódik meg a program futtatása során. A program argumentumai tömbben kerülnek meghívásra. C++-ban fordítás után az adott platform architektúráján futtatjuk a programot bármilyen segítség nélkül, viszont Javában másik programra úgynevezett interpreterre van szükség. Ez sokat visszavesz a sebességből, ezért a legtöbb interpreter platformfüggő kódra fordítja át a programot futtatás előtt ezzel gyorsítva a folyamatot. A Java segítségével tudunk appleteket, melyek html kódba ágyazva tudunk futtatni webes környezetben. Mindkét nyelvben megtalálhatók különböző típusú változók (pl: boolean, int, string, char). Java-ban a final kulcsszót használjuk konstans megadására, a C++-ban a const-t. Mindkét nyelvben megjegyzésekkel tudunk elhelyezni a programkódban a // (egy soros) vagy a /\* \*/ (több soros) jelölések segítségével.

### 11.2. Python

Python: Forstner Bertalan, Ekler Péter, Kelényi Imre: Bevezetés a mobilprogramozásba. Gyors prototípusfejlesztés Python és Java nyelven (35-51 oldal), a kijelölt oldalakból élmény-olvasónapló

A Python-t Guido van Russom hozta létre 1990-ben, hogy megkönnyítse a fejlesztők dolgát. Szkriptnyelvként van besorolva, de használhatjuk összetettebb problémák megoldására is. Nagyon sok már előre

megírt kódkönyvtárral rendelkezik, amelyek a hálózatkezelő, fájlkezelő vagy felhasználói felület kialakítására szolgálnak gyorsítva a fejlesztést. A tesztelés során a fordítási fázis kimarad (ellentétben Java és C++), ezzel gyorsítva a munkát, és szinte minden operációs rendszeren elérhető. Jól működik más nyelvekkel, egyszerű használni. A begin és end kulcsszavakra nincs szükség. A kódsor csoportosítása zárójelek nélkül történik. A széttagolásra sortörést és tabulátorokat használnak. Nincs szükség változók definiálására sem, mert a program futás közben érzékeli a típusát az értékük által (sztringek, számok, listák, szótárak, ennek). Egy utasítás a sor végéig tart, ezért nincs pontossesszű sem. A sorokat tokenekre bontja, melyek fajtái: kulcsszó, operátor, delimiter, literál vagy azonosító. Megjegyzésket a programokódba a # segítséggel lehet.

## **III. rész**

### **Irodalomjegyzék**

## 11.3. Általános

[MARX] Marx, György, *Gyorsuló idő*, Typotex , 2005.

## 11.4. C

[KERNIGHANRITCHIE] Kernighan, Brian W. & Ritchie, Dennis M., *A C programozási nyelv*, Bp., Műszaki, 1993.

## 11.5. C++

[BMECPP] Benedek, Zoltán & Levendovszky, Tíhamér, *Szoftverfejlesztés C++ nyelven*, Bp., Szak Kiadó, 2013.

## 11.6. Lisp

[METAMATH] Chaitin, Gregory, *META MATH! The Quest for Omega*, [http://arxiv.org/PS\\_cache/math/pdf/0404/0404335v7.pdf](http://arxiv.org/PS_cache/math/pdf/0404/0404335v7.pdf) , 2004.

Köszönet illeti a NEMESPOR, <https://groups.google.com/forum/#!forum/nemespor>, az UDPORG tanulószoba, <https://www.facebook.com/groups/udprog>, a DEAC-Hackers előszoba, <https://www.facebook.com/groups/DEACHackers> (illetve egyéb alkalmi szerveződésű szakmai csoportok) tagjait inspiráló érdeklődésekért és hasznos észrevételeikért.

Ezen túl kiemelt köszönet illeti az említett UDPORG közösséget, mely a Debreceni Egyetem reguláris programozás oktatása tartalmi szervezését támogatja. Sok példa eleve ebben a közösségen született, vagy itt került említésre és adott esetekben szerepet kapott, mint oktatási példa.