

Koós Zoltán

Jegyzőkönyv

1.hét

Gagyí
Yoda
PiBBP

2.hét

Liskov
Ciklomatikus komplexitás
Szülő-gyerek

3.hét

BPMN
UML to Code
BME

4.hét

l334d1c4
FullScreen
Encoding

5.hét

JDK
Másoló-mozgató szemantika
Perceptron

6.hét

CustomAlloc
AlternatívTabella
STL map

7.hét

OOCWC lexer
SamuCam
BrainB

8.hét

PortScan
AndroidJáték
Összefoglaló-Android játék

9.hét

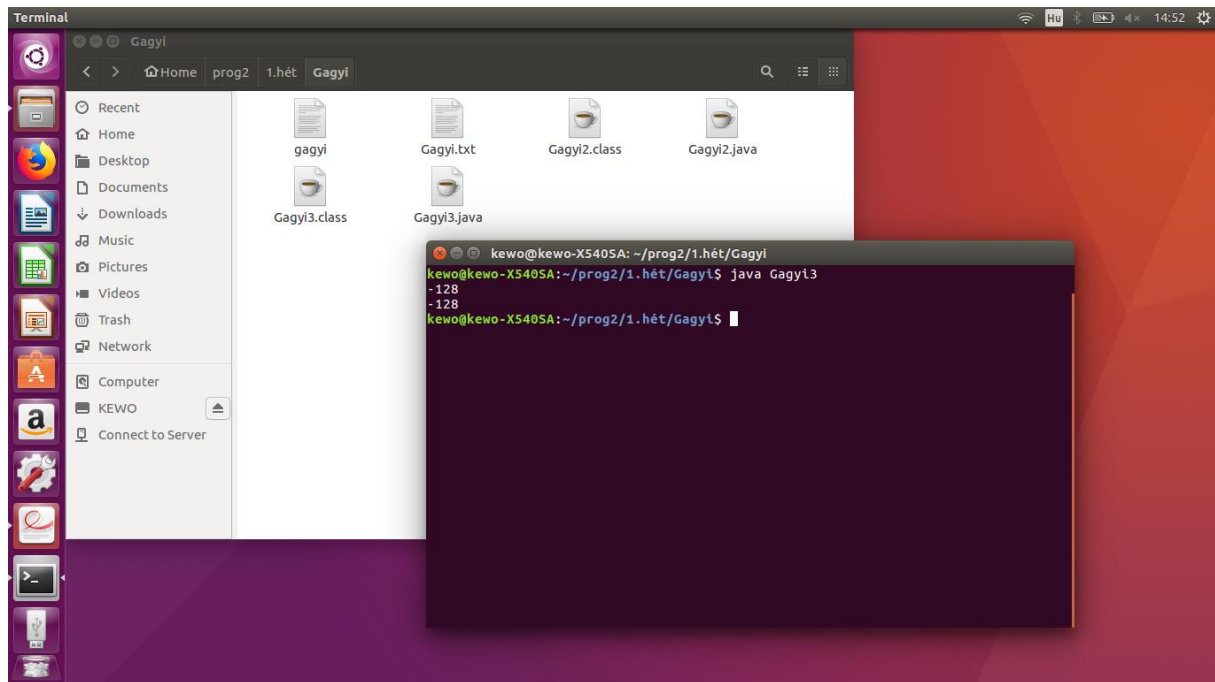
Android Telefon TF
Összefoglaló-Android Telefon TF

1.hét

1.1 Gagyí

Az x és t mint objektum van deklarálva. A -128 és +127 közötti ugyan arra a számra azonos objektumot ad. Ha ezen kívül esnek az értékek akkor két külön objektumot hoz létre ezért végtelen ciklusba kerülne.

<pre>G:\prog2\1.hét\Gagyí\Gagyí3.java - Notepad++ Fájl Szerkesztés Keresés Nézet Kódolás Nyelv Beállítások Eszköz Gagyí3.java 1 public class Gagyí3 2 { 3 4 public static void main (String[]args) 5 { 6 7 Integer x = -128; 8 Integer t = -128; 9 10 System.out.println (x); 11 System.out.println (t); 12 13 while (x <= t && x >= t && t != x); 14 15 } 16 17 }</pre>	<pre>G:\prog2\1.hét\Gagyí\Gagyí2.java - Notepad++ Fájl Szerkesztés Keresés Nézet Kódolás Nyelv Beállítások Eszköz Leiras.txt Perceptron.cpp Összehasonlítás.txt malmo_install.p 1 2 public class Gagyí2 3 { 4 5 public static void main (String[]args) 6 { 7 8 Integer x = -129; 9 Integer t = -129; 10 11 System.out.println (x); 12 System.out.println (t); 13 14 while (x <= t && x >= t && t != x); 15 16 } 17 18 }</pre>
Java source file length : 192	Java source file length : 194



```

829     public static Integer valueOf(int i) {
830         if (i >= IntegerCache.low && i <= IntegerCache.high)
831             return IntegerCache.cache[i + (-IntegerCache.low)];
832         return new Integer(i);
833     }

```



-128 és a high érték közti értékek esetén Integer objektumok egy előre elkészített

pooljából kapjuk meg az értéknek megfelelő objektumot, így a -128 értékre kétszer adja vissza ugyanazt a objektumot, így nyilván a címük is azonos lesz, ezért bukik meg a while feltételünk.

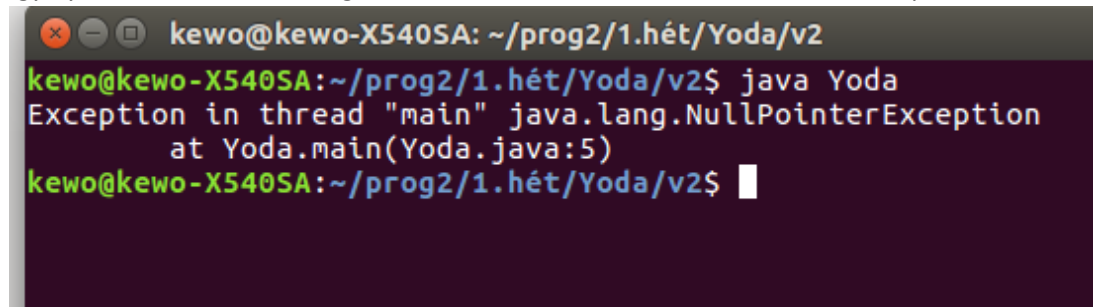
Míg a -129 esetén már a "return new Integer(i);" fog lefutni, azaz két különböző című objektumunk lesz, így válhat majd igazzá az $x != t$ feltétel.

1.2.Yoda

A kifejezésekben a konstansokat helyezzük az első helyre, így elkerülhető a null pointer hiba.

Ha viszont ezt nem tartjuk be, a string amihez hasonlítanánk lényegében egy null pointer,

így nyilván nem lesz lehetséges az összehasonlítás, és futás közbeni hibát kapunk:

A terminal window with a dark background. The title bar shows window control buttons and the text 'kewo@kewo-X540SA: ~/prog2/1.hét/Yoda/v2'. The prompt is 'kewo@kewo-X540SA:~/prog2/1.hét/Yoda/v2\$'. The user has entered 'java Yoda'. The output shows an exception: 'Exception in thread "main" java.lang.NullPointerException at Yoda.main(Yoda.java:5)'. The prompt is now 'kewo@kewo-X540SA:~/prog2/1.hét/Yoda/v2\$' with a cursor.

```
kewo@kewo-X540SA: ~/prog2/1.hét/Yoda/v2
kewo@kewo-X540SA:~/prog2/1.hét/Yoda/v2$ java Yoda
Exception in thread "main" java.lang.NullPointerException
    at Yoda.main(Yoda.java:5)
kewo@kewo-X540SA:~/prog2/1.hét/Yoda/v2$
```

```
class Yoda{
    public static void main(String[] args) {
        String yoda = null;

        if(yoda.equals("asd")){
            System.out.println("wasd");
        }
    }
}
```

1.3PiBBP

The screenshot shows a Linux desktop environment. On the left is a vertical dock with various application icons. The main workspace contains two windows:

- PIBBP.java (~/prog2/1.hét/PIBBP) - gedit**: A text editor window showing the source code of the `PiBBP` class. The code includes comments in Hungarian and English, and implements the BBP algorithm to calculate the hexadecimal digits of pi. The code is as follows:

```
/*
 * PiBBP.java
 *
 * DIGIT 2005, Javat tanítok
 * Bátfai Norbert, nbatfai@inf.unideb.hu
 */
/**
 * A BBP (Bailey-Borwein-Plouffe) algoritmust a Pi hexa
 * jegyeinek számolását végző osztály. A könnyebb olvashatóság
 * kedvéért a változó és metódus neveket megpróbáltuk az algoritmust
 * bemutató [BBP ALGORITMUS] David H. Bailey: The BBP Algorithm for Pi.
 * cikk jelöléséhez.
 *
 * @author Bátfai Norbert, nbatfai@inf.unideb.hu
 * @version 0.0.1
 */
public class PiBBP {
    /** A Pi hexa kifejtésében a d+1. hexa jegytől néhány hexa jegy.*/
    String d16PiHexaJegyek;
    /**
     * Létrehoz egy <code>PiBBP</code>, a BBP algoritmust a Pi-hez
     * alkalmazó objektumot. A [BBP ALGORITMUS] David H. Bailey: The
     * BBP Algorithm for Pi. alapján a
     * {16^d Pi} = {4*{16^d S1} - 2*{16^d S4} - {16^d S5} - {16^d S6}}
     * kiszámítása, a {} a törtészrt jelöli.
     *
     * @param d a Pi hexa kifejtésében a d+1. hexa jegytől
     * számoljuk a hexa jegyeket
     */
    public PiBBP(int d) {
        double d16Pi = 0.0d;
        double d16S1t = d16S1(d, 1);
        double d16S4t = d16S4(d, 4);
        double d16S5t = d16S5(d, 5);
        double d16S6t = d16S6(d, 6);
        d16Pi = 4.0d*d16S1t - 2.0d*d16S4t - d16S5t - d16S6t;
    }
}
```
- Terminal**: A terminal window showing the execution of the program. The commands and output are:

```
kewo@kewo-X540SA: ~/prog2/1.hét/PIBBP
kewo@kewo-X540SA:~/prog2/1.hét/PIBBP$ javac PiBBP.java
kewo@kewo-X540SA:~/prog2/1.hét/PIBBP$ java PiBBP
6C65E5308kewo@kewo-X540SA:~/prog2/1.hét/PIBBP$
```

Below the terminal window, there is a list of links and references:

- 4 Tavalyi prog2 első védés volt.
- 5 https://www.facebook.com/groups/udprog/permalink/437825193072042/?comment_id=437862206401674&reply_comment_id=437863669734861&comment_tracking=%7B%22u%22%3A%22R3%22%7D
- 6 A JDK telepítési könyvtárban az src.zip-ben találd.

At the bottom right, there is a link to the BBP algorithm page: [bl.gov/~dhbailey/dhbpapers/bbp-lgoritmus megvalósítását!](http://bl.gov/~dhbailey/dhbpapers/bbp-lgoritmus%20megvalositasat!)

2.hét

2.1 Liskov

A Liskov elv a következő: ha S altípusa T-nek, akkor minden olyan helyen ahol T-t felhasználjuk S-t is minden gond nélkül behelyettesíthetjük anélkül, hogy a programrész tulajdonságai megváltoznának.

```
1 // ez a T az LSP-ben
2 class Madar {
3 public:
4     virtual void repul() {};
5 };
6
7 // ez a két osztály alkotja a "P programot" az LPS-ben
8 class Program {
9 public:
10     void fgv ( Madar &madar ) {
11         madar.repul();
12     }
13 };
14
15 // itt jönnek az LSP-s S osztályok
16 class Sas : public Madar
17 {};
18
19 class Pingvin : public Madar // ezt úgy is lehet/kell olvasni, hogy a pingvin tud repülni
20 {};
21
22 int main ( int argc, char **argv )
23 {
24     Program program;
25     Madar madar;
26     program.fgv ( madar );
27
28     Sas sas;
29     program.fgv ( sas );
30
31     Pingvin pingvin;
32     program.fgv ( pingvin ); // sérül az LSP, mert a P::fgv röptetné a Pingvint, ami ugye lehetetlen.
33 }
34
35
```

2.2 Ciklomatikus komplexitás

PiBBP.java Ciklomatikus komplexitása:

- public PiBBP(int d) komplexitása: 3 (1 + 1db while + 1db if)
- public double d16Sj(int d, int j) komplexitása: 2 (1 + 1 db for)
- public long n16modk(int n, int k) komplexitása: 5 (1 + 2 db while + 2db if)
- public String toString() komplexitása : 1 (1 + 0)
- public static void main(String args[]) komplexitása : 1 (1 + 0)

(<http://gmetrics.sourceforge.net/gmetrics-CyclomaticComplexityMetric.html> alapján)

2.3 Szülő-Gyerek

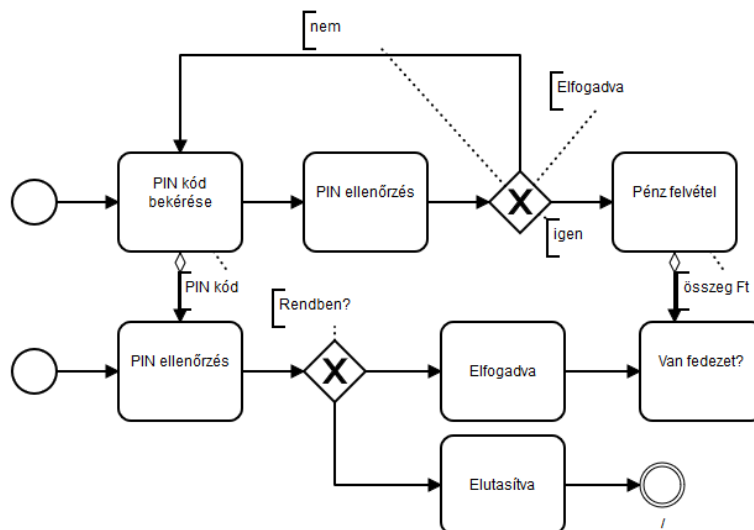
```
class SzuloGyerek{  
    public static void main (String[]args){  
        Szulo szulo = new Gyerek();  
        System.out.println("Gyerek létrehozasa: \nSzulo szulo = new Gyerek()");  
  
        System.out.println("\nA szulo objektum fuggvenyet használjuk:\nszulo.szulokiir()");  
  
        szulo.szulokiir();  
  
        System.out.println("\nA szulo.gyerekkiir()-re már errort adna");  
  
        szulo.gyerekkiir();  
    }  
}
```

```
1 class Szulo{  
2     int num = 10;  
3  
4     public void szulokiir(){  
5         System.out.println("Ez a szulo");  
6     }  
7  
8 }  
9  
10 class Gyerek extends Szulo{  
11     int num = 20;  
12  
13     public void gyerekkiir(){  
14         System.out.println("Ez a gyerek");  
15     }  
16 }
```

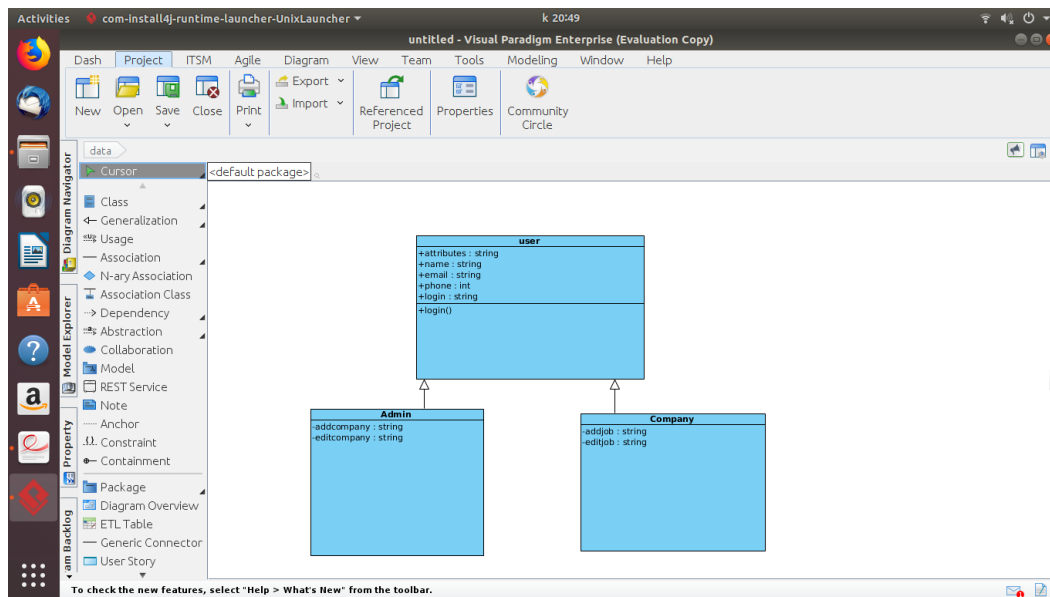
```
kewo@kewo-X540SA: ~/prog2/2.hét/Parent-child/New  
kewo@kewo-X540SA:~/prog2/2.hét/Parent-child/New$ javac SzuloGyerek.java  
SzuloGyerek.java:13: error: cannot find symbol  
        szulo.gyerekkiir();  
            ^  
symbol:   method gyerekkiir()  
location: variable szulo of type Szulo  
1 error
```

3.hét

3.1 BPMN



3.2 Forward engineering UML osztálydiagram



```
1 public class user {
2
3     public string attributes;
4     public string name;
5     public string email;
6     public int phone;
7     public string login;
8
9     public void login() {
10         // TODO - implement user.login
11         throw new UnsupportedOperationException();
12     }
13
14 }
```

G:\prog2\3.hét\UML_to_Code\New\Company.java - Notepad++

Fájl Szerkesztés Keresés Nézet Kódolás Nyelv Beállítások Eszközök Makró

```
1 public class Company extends user {
2
3     private string addjob;
4     private string editjob;
5
6 }
```

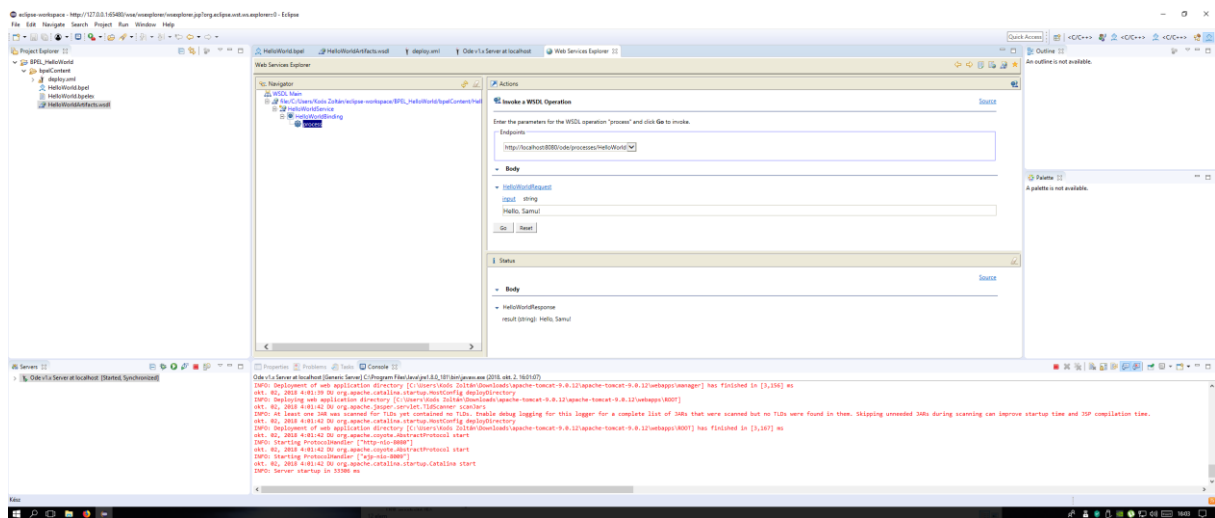
G:\prog2\3.hét\UML_to_Code\New\Admin.java - Notepad++

Fájl Szerkesztés Keresés Nézet Kódolás Nyelv Beállítások Eszközök Makró

```
1 public class Admin extends user {
2
3     private string addcompany;
4     private string editcompany;
5
6 }
```


3.3 BPEL Helló, Világ! - egy visszhang folyamat

Tutorial alapján:



4.hét

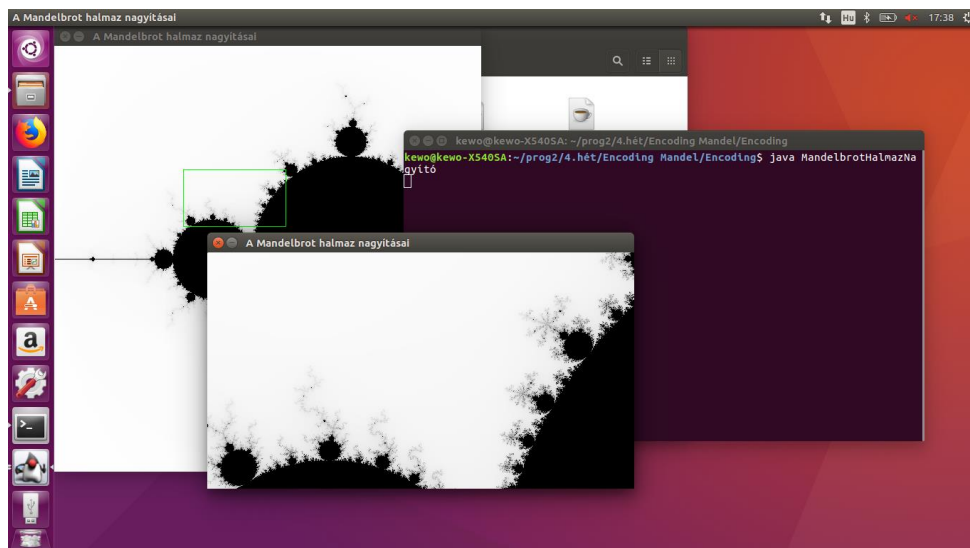
4.1 Encoding

A JRE telepítésekor kiválaszt az operációs rendszer alapján egy karakterkódolást. Azért maradhat benne az ékezet, mert érzékelte a magyar nyelvet.

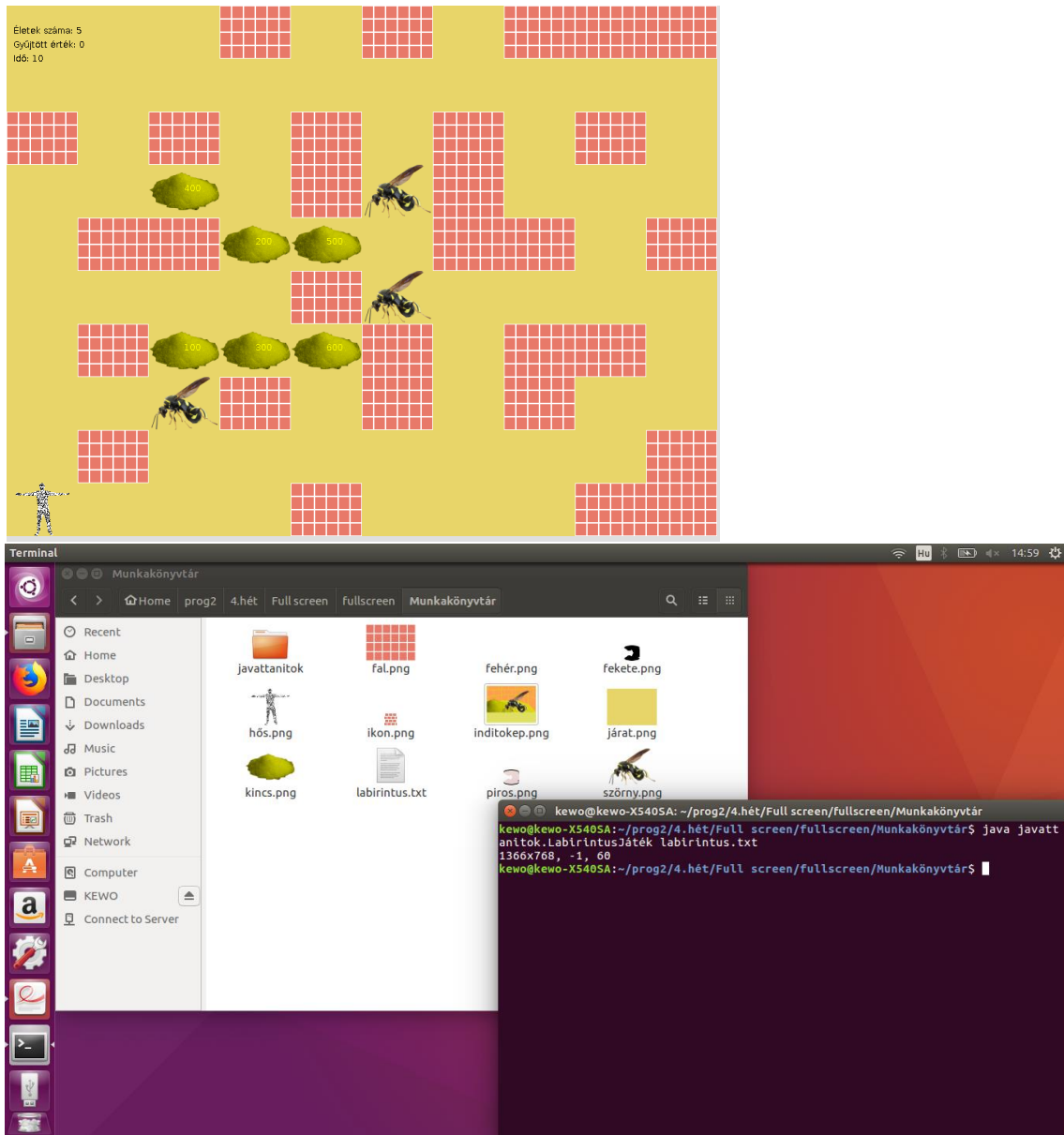
Ha a kódolást megváltoztatnánk olyanra, amely nem tartalmazza az ékezeteket, akkor nem fordulna le.

"By default, the JRE 8 installer installs a European languages version if it recognizes that the host operating system only supports European languages."

<https://docs.oracle.com/javase/8/docs/technotes/guides/intl/encoding.doc.html>



4.2 FullScreen



4.3 l334d1c4

```
import java.io.*;
import java.util.*;

public class leet {
    public static void main(String[] args) throws FileNotFoundException {
        Scanner input = new Scanner(new File("normal.txt"));
        PrintStream out = new PrintStream(new File("leet.txt"));
        leetSpeak(input, out);
    }

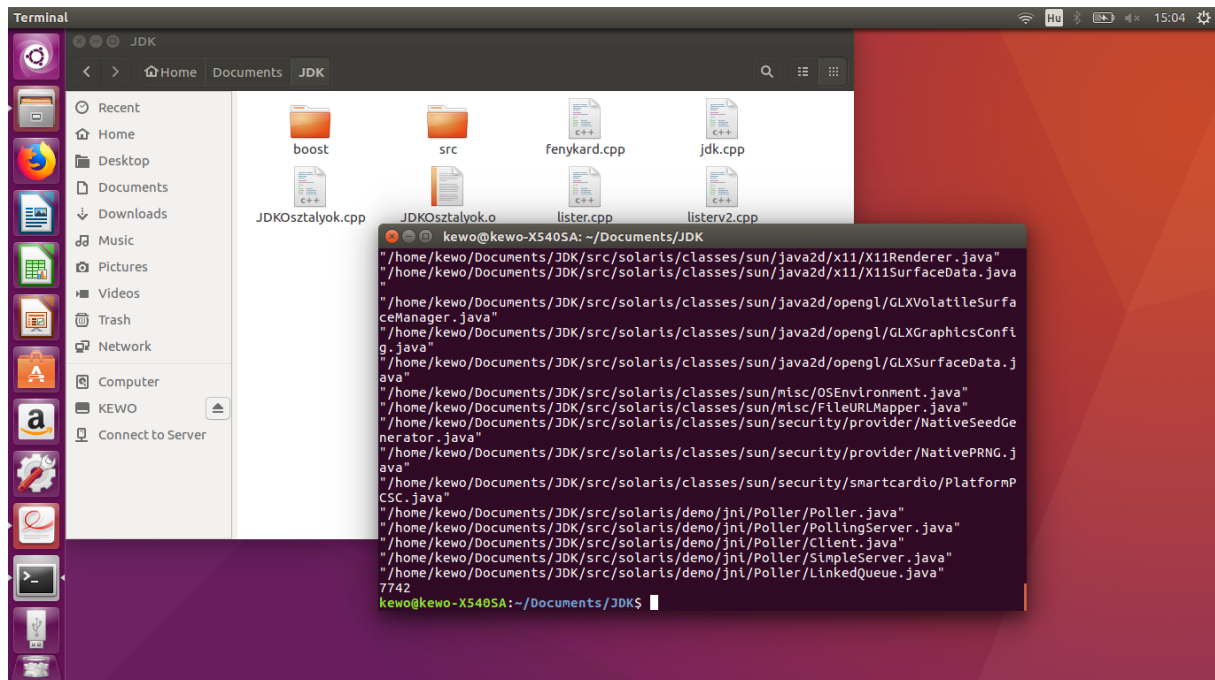
    public static void leetSpeak(Scanner input, PrintStream output) {
        while (input.hasNextLine()) {
            String line = input.nextLine();
            Scanner tokens = new Scanner(line);

            while (tokens.hasNext()) {
                String token = tokens.next();
                token = token.replace("a", "4");
                token = token.replace("o", "0");
                token = token.replace("l", "1");
                token = token.replace("e", "3");
                token = token.replace("t", "7");
                token = token.replace("b", "8");
                token = token.replace("d", "|");
                token = token.replace("f", "|=");
                token = token.replace("g", "6");
                token = token.replace("j", "_");
                token = token.replace("x", "|2");

                output.println(token);
            }
        }
    }
}
```

5.hét

5.1 JDK



```
#include <iostream>
#include <boost/filesystem.hpp>

using namespace std;
using namespace boost::filesystem;

void bejaro(std::string root);
int counter = 0;

int main(int argc, char* argv[]){

    //string root = "/usr/lib/jvm/java-11-openjdk-amd64";
    string root = "/home/kewo/Documents/JDK";
    bejaro(root);
    cout << counter << endl;

}

void bejaro(std::string root){
    for(recursive_directory_iterator end, dir(root); dir != end; dir++){
        if(extension(*dir) == ".java"){
            cout<<*dir<<endl;
            counter++;
        }
    }
}
```

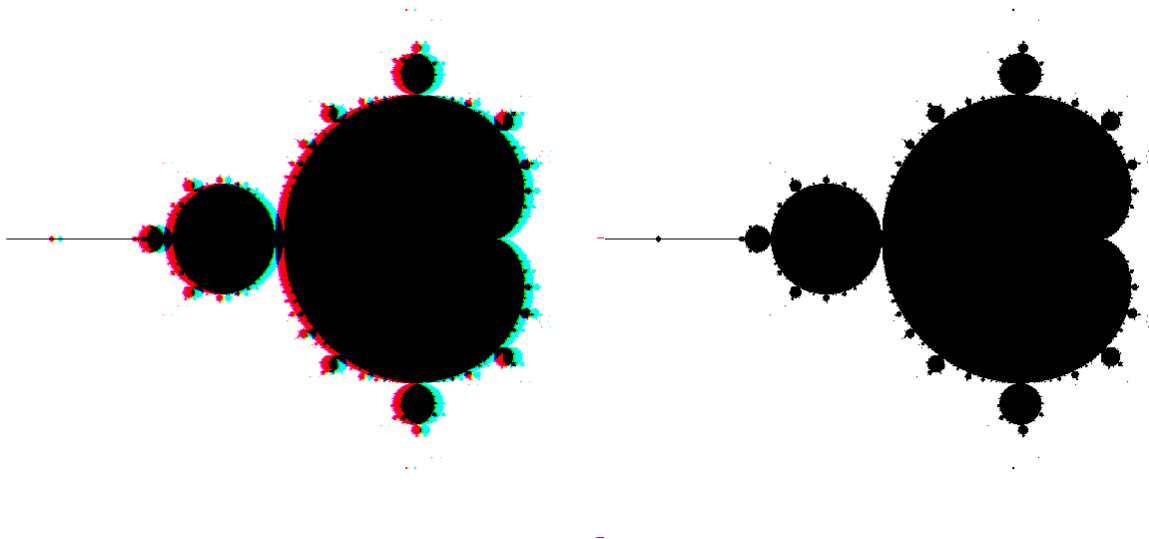
5.2 Másoló-mozgató szemantika

```
kewo@kewo-X540SA: ~/Documents/Másoló-mozgató
kewo@kewo-X540SA:~/Documents/Másoló-mozgató$ g++ wat.cpp -o wat.o -std=c++11
kewo@kewo-X540SA:~/Documents/Másoló-mozgató$ ./wat.o
pl1 értékei:
Az a érték: 12 A b érték 4 , és b 0x1fb0c20-re mutat
pl1 értékei:
Az a érték: 12 A b érték 4 , és b 0x1fb0c20-re mutat
pl2 értékei:
Az a érték: 12 A b érték 4 , és b 0x1fb1050-re mutat
pl3 értékei:
Az a érték: 20 A b érték 5 , és b 0x1fb1070-re mutat
pl3 értékei:
Az a érték: 20 A b érték 5 , és b 0x1fb1070-re mutat
pl2 értékei:
Az a érték: 20 A b érték 5 , és b 0x1fb1050-re mutat
pl1 értékei:
b nem létezik, a pedig 0
pl4 értékei:
Az a érték: 12 A b érték 4 , és b 0x1fb0c20-re mutat
```

```
4 int main(){
5     class asd{
6     public:
7         asd(int param1, int& param2) : a(param1), b(new int){           //konstruktor 2 paraméterrel
8             *b = param2;
9         }
10
11
12
13         asd(const asd &pl){                                           //másoló konstruktor
14             a = pl.a;
15             b = new int;
16             *b = *pl.b;
17         }
18
19         asd& operator= (asd &pl){                                     //másoló értékadás
20             a = pl.a;
21             *b = *pl.b;
22             return *this;
23         }
24
25         asd(asd && pl){                                              //mozgató konstruktor
26             a = 0;
27             b = nullptr;
28             *this = std::move(pl);
29         }
30
31         asd& operator= (asd && pl){                                   //mozgató értékadás
32             std::swap(b,pl.b);
33             std::swap(a,pl.a);
34             return *this;
35         }
36
37         void Print(){
38             if(b!=NULL)
39                 std::cout<<"Az a érték: "<<a<<" A b érték "<<b<<" , és b "<<b<<"-re mutat"<<std::endl;
40             else
41                 std::cout<<"b nem létezik, a pedig 0"<<std::endl;
42         }
43
44         ~asd(){                                                       //destruktor
45             delete b;
46         }
47     };
48
49     //...
50
51     //...
52
53     //...
54
55     //...
56 }
```

5.3 Perceptron

A feladat az volt, hogy a Perceptron osztályának bemenetére egy képet teszünk és egy ugyanakkora méretű képet adjon vissza.



```
kewo@kewo-X540SA:~/Documents/Perceptron/png++$ ./main mandel.png
```

```
Kimeneti kep mentve!
```

```
layers: 3
```

```
[1]: 15.8145 - 256
```

```
[2]: -0.00854808 - 360000
```

```
//Saját kép
```

```
for(int i{0}; i<png_image.get_width(); ++i)
    for(int j{0}; j<png_image.get_height(); ++j) {
        png_image[i][j].red = image_er[i*png_image.get_width()+j];
        png_image[i][j].blue = image_er[i*png_image.get_width()+j+7];
        png_image[i][j].green = image_er[i*png_image.get_width()+j+9];
    }
```

```
png_image.write("enyim.png");
cout << endl << "Kimeneti kep mentve! " << endl << endl;
```

6.hét

6.1 Custom Alloc

```
kewo@kewo-X540SA: ~/prog2/6.hét/CustomAlloc
kewo@kewo-X540SA:~/prog2/6.hét/CustomAlloc$ ./alloc
Allocating 1 object of 4 bytes. i=int hivasok szama 1 hivasok szama 1
q 0x7f110aa77064
Allocating 2 object of 8 bytes. i=int hivasok szama 2 hivasok szama 2
q 0x7f110aa77068
Allocating 4 object of 16 bytes. i=int hivasok szama 3 hivasok szama 3
q 0x7f110aa77070
v 0x7f110aa77000
o 0x7f110aa77000
0 0x7f110aa77000
42
43
44
42
43
44
```

```
Arena& arena;
int nn {0};

CustomAlloc (Arena& arena) : arena (arena) {}

pointer allocate(size_type n) {
    ++arena.nnn;
    ++nn;
    int s;
    char* p = abi::__cxa_demangle(typeid(T).name(), 0, 0, &s);

    cout << "Allocating " << n << " object of " << n*sizeof(T) << " bytes. " << typeid(T).name()
    << "==" << p << " hivasok szama " << nn << " hivasok szama " << arena.nnn << endl;

    free(p);

    char* q = arena.q;

    cout << "q " << static_cast<const void*>(q) << endl;

    arena.q += n*sizeof(T);

    return reinterpret_cast<T*>(q);
}

void deallocate (pointer p, size_type n) {}
};

int main(int argc, char **argv) {

    int osztott_memoria;
    char* osztott_memoria_terulet;

    if( (osztott_memoria = shmget(ftok(".", 44), 10*1024*1024, IPC_CREAT | S_IRUSR | S_IWUSR)) == -1 ) {
        perror( "shmget" );
        exit( EXIT_FAILURE );
    }
}
```

6.2 Alternatív Tabella

- A Csapat osztály nem önmagában egy értékkel fog rendelkezni hanem egyszerre kettővel is. A nev és az erteek változók”

kombinálása megy végbe azzal hogy az elején már megadjuk a programnak az osztály létrehozásakor, hogy mire is készüljön fel.

Innentol kezdve a Csapat típusú változók egyszerre két értéket fognak magukban hordozni, de nem mindegy hogy hogyan. A nev”

és erteek változókat össze kell hasonlítani egymással a compareTo meghívásával. A compareTo segítségével meg tudjuk szabni,

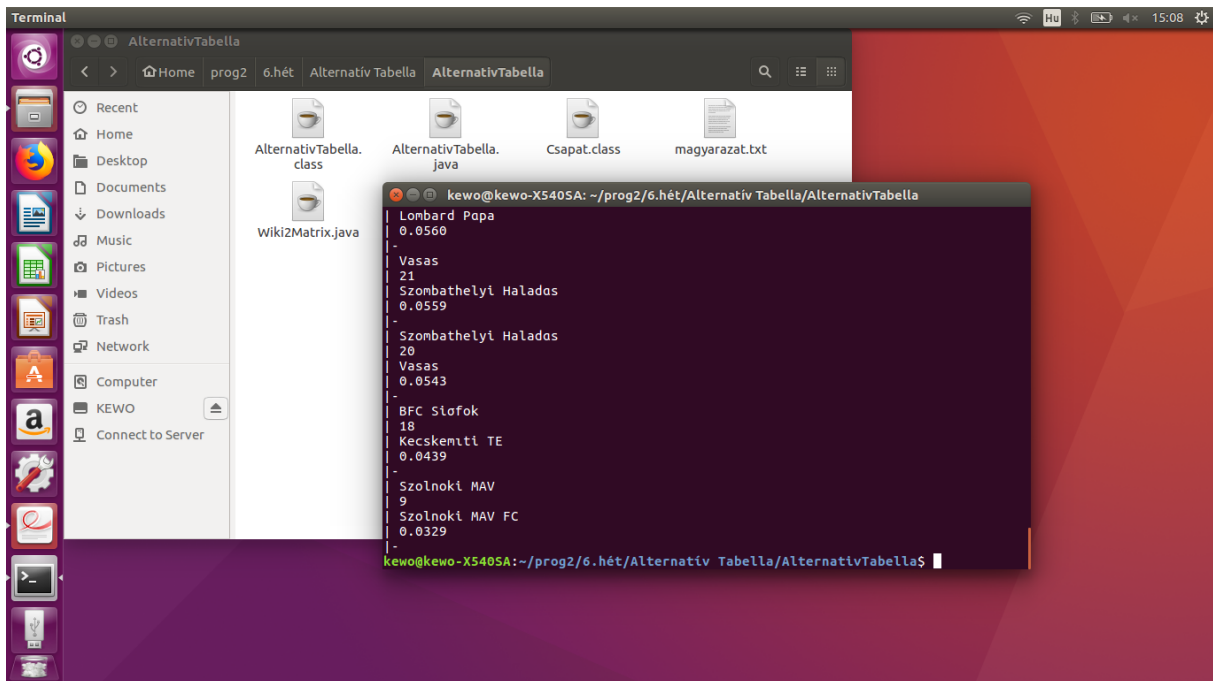
hogy melyik esetben mit szeretnénk milyen értékkel térjen vissza a compareTo függvényünk. Meg tudjuk neki mondani hogy

mit hasonlítson össze. Itt a példában ez az érték változó. A compareTo összehasonlítja az érték változót az éppen kiválasztott

elemét a többi nev változóhoz tartozó érték-vel és ez alapján tér vissza. -1 ha a kiválasztott érték kisebb mint a többi érték, +1 ha

nagyobb és 0 ha egyenlo. A compareTo függvény ez alapján visszaadja nekünk az összes elemünket, amit létrehozunk a Csapat”

osztályban méghozzá érték szerinti sorrendben.



The screenshot shows a Linux desktop environment with a purple and red background. A file manager window titled 'AlternativTabella' is open, displaying a directory structure with files like 'AlternativTabella.class', 'AlternativTabella.java', 'Csapat.class', 'magyarazat.txt', and 'Wiki2Matrix.java'. A terminal window is also open, showing the output of a program. The terminal output lists football teams and their scores, sorted by score in descending order.

```
kewo@kewo-X540SA: ~/prog2/6.hét/Alternativ Tabella/AlternativTabella
Lonbard Papa
0.0560
-
Vasas
21
Szombathelyi Haladas
0.0559
-
Szombathelyi Haladas
20
Vasas
0.0543
-
BFC Siófok
18
Kecskeméti TE
0.0439
-
Szolnoki MÁV
9
Szolnoki MÁV FC
0.0329
-
kewo@kewo-X540SA:~/prog2/6.hét/Alternativ Tabella/AlternativTabella$
```


6.3 STL map érték szerinti rendezése

```
kewo@kewo-X540SA:~/prog2/6.hét/STL map$ ./stlmap.o
Rendezetlen map elemei:
wat0 194
wat1 821
wat2 200
wat3 986
wat4 129
wat5 298
wat6 621
wat7 434
wat8 801
wat9 604
Rendezett map elemei:
wat3 986
wat1 821
wat8 801
wat6 621
wat9 604
wat7 434
wat5 298
wat2 200
wat0 194
wat4 129
```

```
int randomNumber(int min, int max){
    return min+ rand()%(max-min);
}

void mapPrinter(std::map<std::string, int>& in){
    for(auto & it: in){
        std::cout << it.first << " " << it.second << std::endl;
    }
}

void mapFiller(std::map<std::string, int> &in){
    for(int i = 0; i<10; i++){
        std::string temp = "wat" + std::to_string(i);
        in[temp] = randomNumber(0, 1000);
    }
}

std::vector<std::pair<std::string, int>> mapSorter(std::map<std::string, int>& in){
    std::vector<std::pair<std::string, int>> ordered;

    for(auto &it :in){
        std::pair<std::string, int> temp{it.first, it.second};
        //std::cout << it.first << "\t" << it.second << std::endl;
        ordered.push_back(temp);
    }

    std::sort (
        std::begin ( ordered ), std::end ( ordered ),
        [ = ] ( auto && p1, auto && p2 ) {
            return p1.second > p2.second;
        }
    );
};
```

7.hét

7.1 OOCWC Boost ASIO hálózatkézelése

Az `sscanf` hasonlóan működik, mint egy sima `scanf`, tehát adatokat tudunk beolvasni vele, annyi különbséggel, hogy itt nem fileból vagy a standard inputról olvasunk be adatokat, hanem egy string bufferből. Az első paraméter a buffer neve, a második a formátuma. A példában `%n`-ben tároljuk el a beérkező string méretét.

```
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} );
}
```

7.2 BrainB

A program során a Samu Entropyt kell követnünk az egérrel, amely egyre nehezebb a megjelenő négyzetek miatt.

A program Qt slot-signal mechanizmust is használ. ami arra jó, hogy objektumok könnyedén tudjanak kommunikálni egymással. Tehát egy objektum signalokat tud küldeni egy másik objektum slotjaira. A példában az látszik, hogy a `brainBThread` objektum egy signalt küld a `BrainBWin` objektum slotjára.

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

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

Ebben az esetben ha a `brainBThread`-ben lefut a `heroesChanged`, akkor lefut ennek (`BrainBWin`) a `updateHeroes` függvénye. Az `endAndStats`-nél szintén ugyan ez történik.

7.3 SamuCam

A `videoCapture.open (videoStream);` sort kellett módosítani. -> `videoCapture.open (0);`

A mainben láthatjuk hogy a `videostream` a `samucam` konstruktorában van, ahol a `samucam` bekéri a `videostream` stringet.

Ezután az `openVideoStream`-ben a konstruktorba megadott adatok alapján beállítjuk a kép méreteit,

majd pedig a másodpercenkénti képszámot: 10

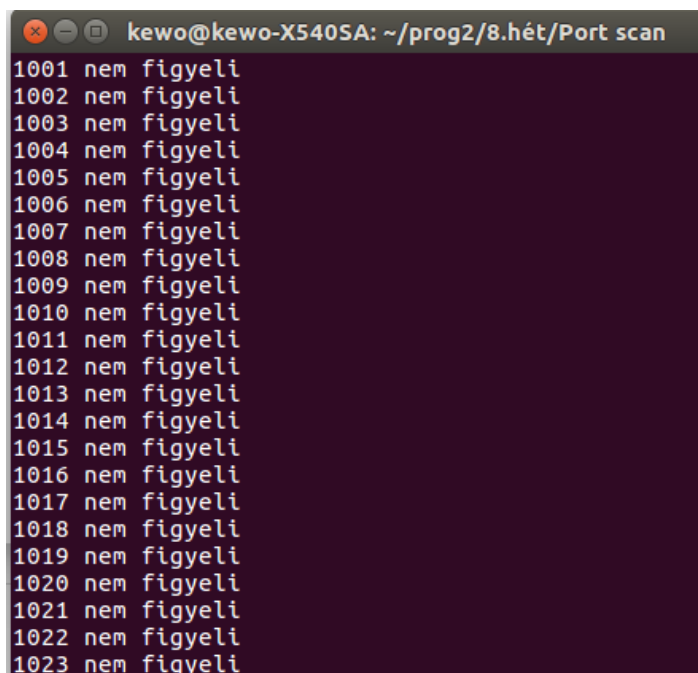
```
void SamuCam::openVideoStream()
{
    videoCapture.open ( videoStream );

    videoCapture.set ( CV_CAP_PROP_FRAME_WIDTH, width );
    videoCapture.set ( CV_CAP_PROP_FRAME_HEIGHT, height );
    videoCapture.set ( CV_CAP_PROP_FPS, 10 );
}
```

8.hét

8.1 Portscan

A program 0-tól 1023-ig megvizsgálja a megadott nevű gép TCP kapuit. Ha sikerül létrehozni a kapcsolatot, akkor kiírja, hogy „figyeli” és bezárja a socketet, ha nem sikerül létrehozni akkor azt, hogy „nem figyeli”. Mind ez egy kivételkezelésben van megírva, így a program nem áll meg sikertelen kapcsolás esetén, hanem tovább lép, mivel a kivételkezelés futási időben keletkező hibákat kezel. Tehát, ha hiba is lép fel, a catch elkapja, így a program tovább tud működni leállítás nélkül.



```
kewo@kewo-X540SA: ~/prog2/8.hét/Port scan
1001 nem figyeli
1002 nem figyeli
1003 nem figyeli
1004 nem figyeli
1005 nem figyeli
1006 nem figyeli
1007 nem figyeli
1008 nem figyeli
1009 nem figyeli
1010 nem figyeli
1011 nem figyeli
1012 nem figyeli
1013 nem figyeli
1014 nem figyeli
1015 nem figyeli
1016 nem figyeli
1017 nem figyeli
1018 nem figyeli
1019 nem figyeli
1020 nem figyeli
1021 nem figyeli
1022 nem figyeli
1023 nem figyeli
```

8.2-3 Android játék + összefoglaló

Az asd nevezetű szimpla android játékot a <https://www.androidauthority.com/create-a-2d-platformer-for-android-in-unity-693550/> oldalon található egyszerű tutorial alapján készítettem.

Először is le kellett szedni és felrakni a Unity3d engine-t. Ezen kívül a JDK legújabb verziójára a java SDK-ra, android studio-ra volt szükség, mivel ezek elengedhetetlenek egy android játék fejlesztéséhez és később egy apk fájl felépítéséhez.

Miután elindítottuk a Unity-t létrehozunk egy 2d projektet. Majd hozzáadjuk a karakterünk és a platform képfájljait. Ezek mint objectek már felruházhatóak bizonyos tulajdonságokkal. A karakterünkhöz hozzárendelünk egy 2d rigidbody-t ami fizikális testet add neki így más fizikális testekkel kapcsolatba tud kerülni. Egy boxcollider-t is adunk hozzá és a platformhoz is, így tudnak egymással kapcsolatba kerülni. Ezután megírjuk a Controls nevű c# scriptet amivel karakterünk képes lesz mozogni jobbra és balra:

```
public class Controls : MonoBehaviour {
    public Rigidbody2D rb;
    public float movespeed;

    void Start () {
        rb = GetComponent<Rigidbody2D>();
    }

    void Update () {

        if (Input.GetKey(KeyCode.LeftArrow))
        {
            rb.velocity = new Vector2(-movespeed, rb.velocity.y);
        }
        if (Input.GetKey(KeyCode.RightArrow))
        {
            rb.velocity = new Vector2(movespeed, rb.velocity.y);
        }
    }
}
```

Ezt a scriptet hozzácsatoljuk a player-hez és máris életre kelt a karakterünk. Mivel telefonon szeretnénk futtatni játékunk ezért még ki kell egészíteni az Update részt:

```
if (moveright)
{
    rb.velocity = new Vector2(movespeed, rb.velocity.y);
}
if (moveleft)
{
    rb.velocity = new Vector2(-movespeed, rb.velocity.y);
}
```

Majd egy Touch nevű c# scriptet írunk, ami a nyilak lenyomására fog reagálni. Persze előtte be kell tennünk a nyilakat.

```
using UnityEngine;
using System.Collections;
```

```

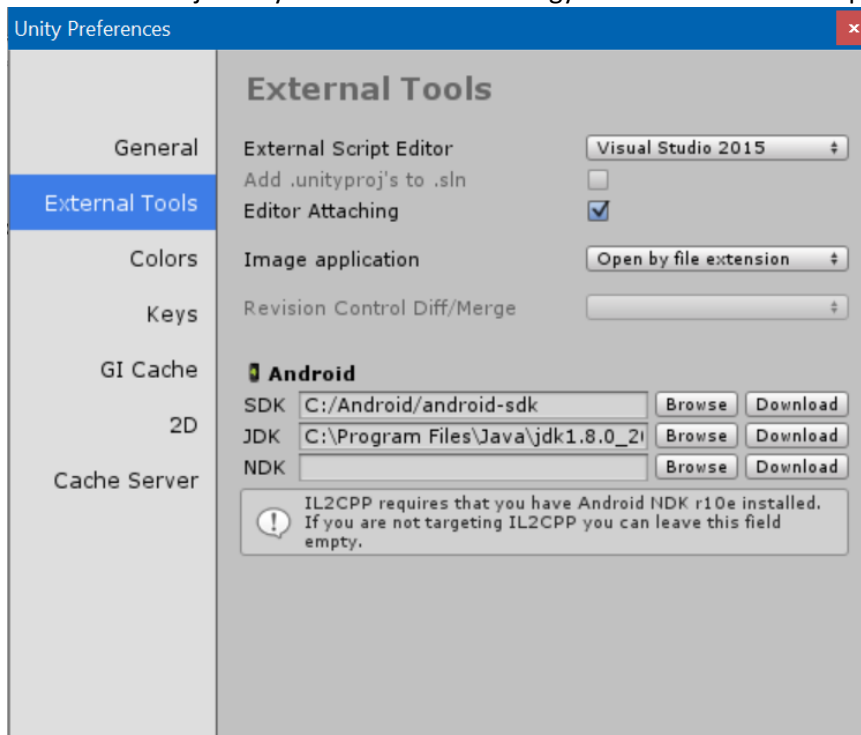
public class Touch : MonoBehaviour
{
    private Controls player;

    void Start()
    {
        player = FindObjectOfType<Controls>();
    }

    public void LeftArrow()
    {
        player.moveright = false;
        player.moveleft = true;
    }
    public void RightArrow()
    {
        player.moveright = true;
        player.moveleft = false;
    }
    public void ReleaseLeftArrow()
    {
        player.moveleft = false;
    }
    public void ReleaseRightArrow()
    {
        player.moveright = false;
    }
}
}

```

Ezt hozzárendeljük a nyilakhoz és készen is vagyunk. Ha feltettük a felépítéshez szükséges elemeket:



Akkor a Build menüpontra kattintva elkészíthetjük az apk-t amit telefonra feltelepíthető.

9.hét

9.1 Android Telefon TF

A TensorFlow Detect demo app bemutatja, hogy egy SSD-MobileNet modell amit a Tensorflow Object Detection API-val betanítva, fel tud ismerni, fizikális helyét meghatározni és követni jelen időben folyamatosan 80 különböző kategóriából a mobilunk kamerájának képén található objektumokat.

Az általam megkísérelt teszt alatt felismerte mindkét monitoromat, a billentyűzetemet és a laptopom billentyűzetét és monitorát is. Ezeket különböző színnel bekeretezte és azt is kiírta hogy mennyire biztos benne, hogy az az objektum amit felismert az abba a kategóriába tartozik ahova ő besorolta.

Mivel jelenleg 80 különböző kategória van ezért nem mindent tud megfelelően felismerni, de a megjelöl objektumot a felismert tulajdonságok alapján besorolja valahova, ez minden alkalommal más is lehet attól függően, hogy mik a domináns tulajdonságok.



<https://github.com/tensorflow/tensorflow/blob/master/tensorflow/examples/android/src/org/tensorflow/demo/> linken található a demo program kódja.

9.2-3 Malmo

Tutorial alapján feltettem és több példaprogramot is lefuttattam.

