



Java - please (Netbeans)

by Szabó Dávid (2016)



Figyelem!

Ez az előadás csak erős idegzetűeknek ajánlott.
A prezentáció elkészítése során senkinek nem sérült meg (eddig).

Igen, lesz elmélet, bocsi.

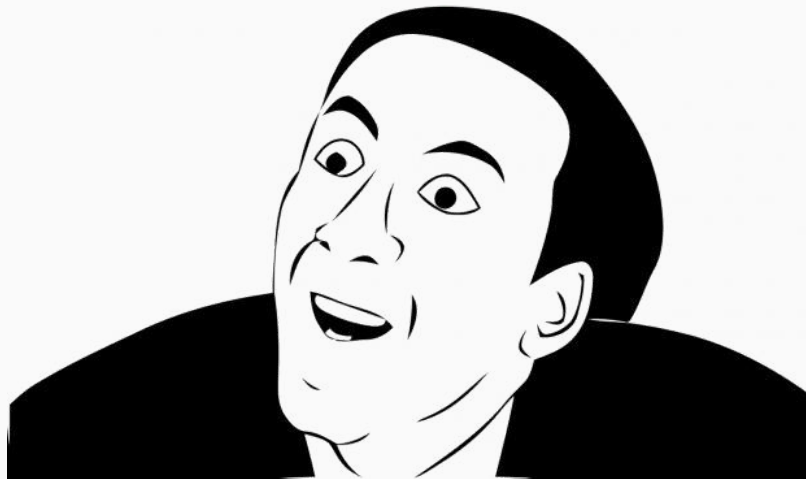
U.I. Az előadás 100%-os megértéséhez angol nyelvismeret szükséges.

Mi is az a Java?



(osztály alapú)
objektum-orientált **programozási nyelv**

YOU DON'T SAY?

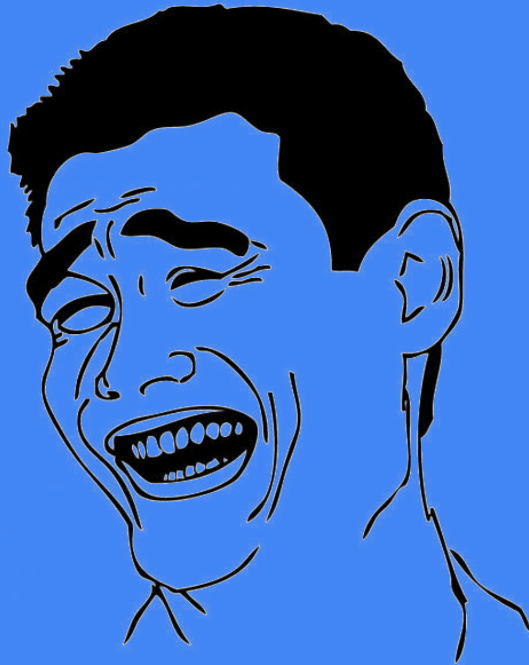


Mi az az **objektum**?

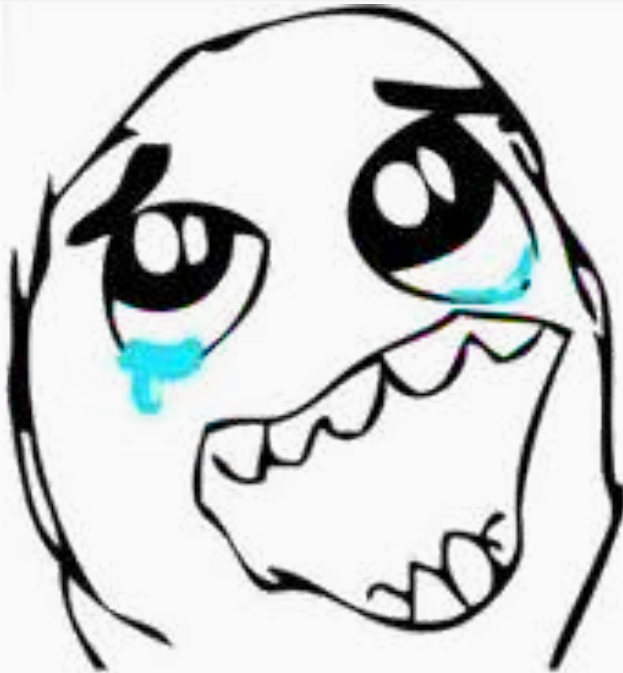
Nézz körül, rengeteg példát fogsz rá találni. Minden objektumnak van **állapota** és **viselkedése**.

Például a **kutyának** van **állapota** (név, szín, fajtája, éhség) és **viselkedése** (ugatás, labda visszahozása, farok csóválás).

Azt hitted könnyen megúszod mi?
Ez még csak a kezdet...



De először....
jöjjön egy kis gyakorlat



Netbeans projekt

File -> New Project
(Ctrl + Shift + N)



Program
belépőpontja



New Java Application

Steps

1. Choose Project
2. **Name and Location**

Name and Location

Project Name:

Project Location:

Project Folder:

☐ Use Dedicated Folder for Storing Libraries

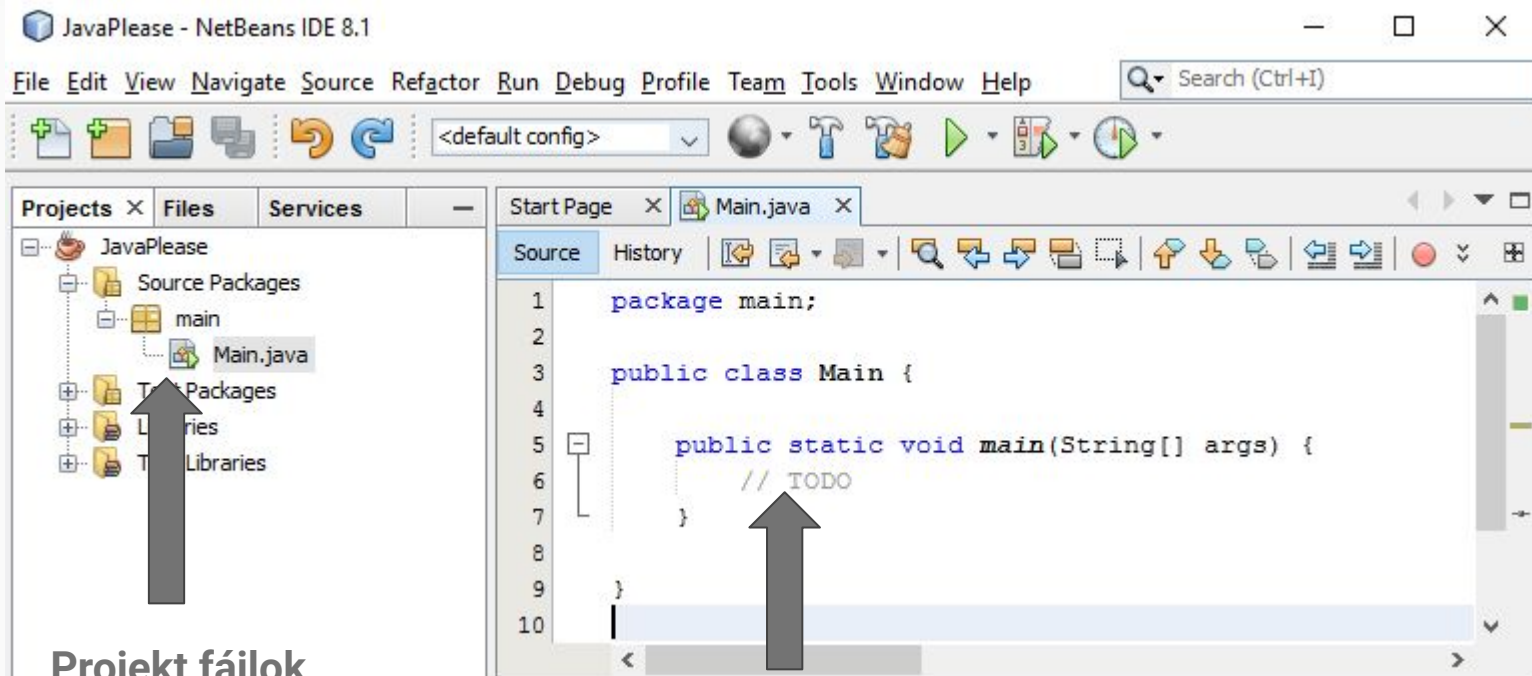
Libraries Folder:

Different users and projects can share the same compilation libraries (see Help for details).

☒ Create Main Class

< Back Next > **Finish** Cancel Help

A FORRÁSKÓÓD



Projekt fájlok
csomagok, osztályok,
interfészek

Program
Belépőpont

Az első programunk

```
package main;

public class Main {

    public static void main(String[] args) {
        System.out.println("Ti nem unjátok már ezt a 'Helló Világ' cuccot?");
    }
}
```

out - JavaPlease (run) X

```
run:
Ti nem unjátok már ezt a 'Helló Világ' cuccot?
BUILD SUCCESSFUL (total time: 0 seconds)
```

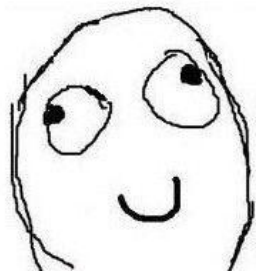


Forráskód elemzése



Osztályaidat, interfészeidet csomagokba rendezheted. Valójában mappáként funkcionálnak, de \ elválasztó helyett . (pont) az elválasztó.

Csomagok segítségével könnyen elkülönítheted különböző részeit a programodnak, így könnyebben karbantartható.



dafuq did i just read

```
package main;

public class Main {

    public static void main(String[] args) {
        // TODO
    }
}
```



Pl.:

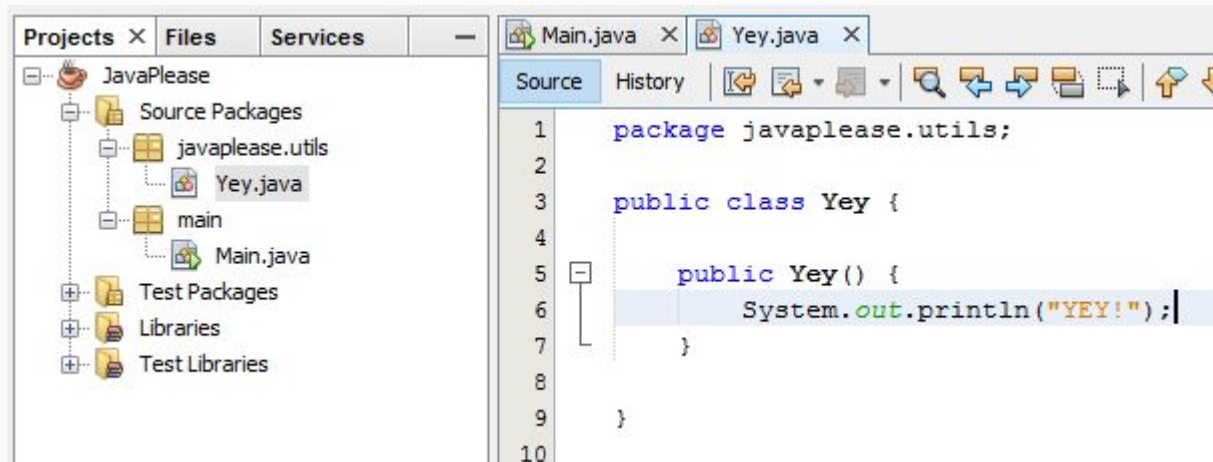
- flappybird.engine -> játékmotor
- flappybird.gui -> felület
- flappybird.utils -> segédosztályok

Más csomagban lévő osztályok, interfészek nem lesznek elérhetőek, ezért importálni kell vagy teljes elérési útvonalat megadni.

Tételezzük fel van egy **Yey** osztályunk a **javaplease.utils** csomagban, amely példányosítás esetén kiírja hogy **YEY!**.

A Yey osztályt a main csomagból akarjuk elérni a Main osztályból, tehát a **main.Main** -ből. Ezért vagy hivatkozunk rá teljesen: **javaplease.utils.Yey** vagy importáljuk.

Csomag, importálás



Csomag, importálás

The screenshot shows an IDE interface with the following components:

- Projects Panel (Left):** Displays the project structure for 'JavaPleace'. It includes 'Source Packages' (javapleace.utils, main) and 'Test Packages'. The 'main' package contains 'Main.java'.
- Editor (Center):** Shows the code for 'Main.java'. The code is as follows:

```
1 package main;
2
3 import javapleace.utils.Yey;
4
5 public class Main {
6
7     public static void main(String[] args) {
8         Yey y = new Yey();
9     }
10
11 }
```
- Output Panel (Bottom):** Shows the output of the run command. The output is:

```
run:
YEE!
BUILD SUCCESSFUL (total time: 0 seconds)
```


Belépőpont, avagy entry point

A programod ettől a ponttól kezd el futni amikor futtatják.

```
package main;

public class Main {

    public static void main(String[] args) {
        // TODO
    }
}
```

New Java Application

Steps

1. Choose Project
2. Name and Location

Name and Location

Project Name:

Project Location:

Project Folder:

☐ Use Dedicated Folder for Storing Libraries

Library Folder:

Different users and projects can share the same compilation libraries (see Help for details).

☒ Create Main Class



Amennyiben nem volt teljesen világos minden eddig, az nem baj, később tisztázzuk ezeket a dolgokat.



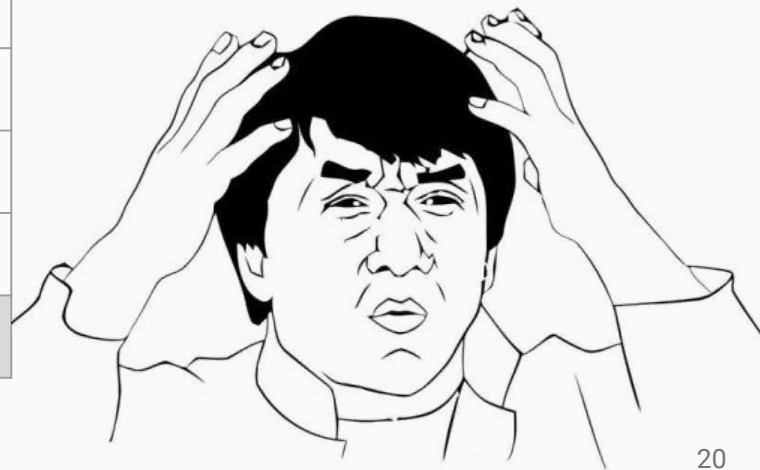
Okay

Primitív adattípusok



Primitív adattípusok

byte	előjeles egész szám, 8 bit
short	előjeles egész szám, 16 bit
int	előjeles egész szám, 32 bit
long	előjeles egész szám, 64 bit
float	egyszeres pontosságú lebegőpontos szám (32 bit)
double	kétszeres pontosságú lebegőpontos szám (64 bit)
char	16 bit unicode karakter
boolean	igaz / hamis
String	karakterlánc



Változók gyakorlatban

```
package main;

public class Main {

    public static void main(String[] args) {
        byte level = 125;
        short somewhatSmallNumber = 32766;
        int experiencePoint = 2147483645;
        long veryBigNumber = 9223372036854775407L;
        float somewhatPreciseValue = 5432.3432f;
        double preciseValue = 5432.342452345546;
        char characterForYou = 'a';
        String name = "Chuck Norris";
        boolean isJumping = false;
    }
}
```

L vagy l affix

F vagy f affix

false (hamis)
true (igaz)

Habár a karakterláncot primitív adattípusként láttuk, ez nem teljesen igaz rá. Amikor beírunk egy szöveget mint pl. `“cica”` ekkor a Java létrehoz nekünk egy `String` objektumot.

Karakterlánchoz könnyen hozzáfűzhetünk másik karakterláncot vagy más adatokat is a `+` operátor segítségével.

Karakterlánc, String

```
package main;

public class Main {

    public static void main(String[] args) {
        int age = 18;
        System.out.println("Én "+age+" éves vagyok.");
        System.out.println(age+" éves vagyok.");
        System.out.println("Ennyi éves vagyok: "+age);

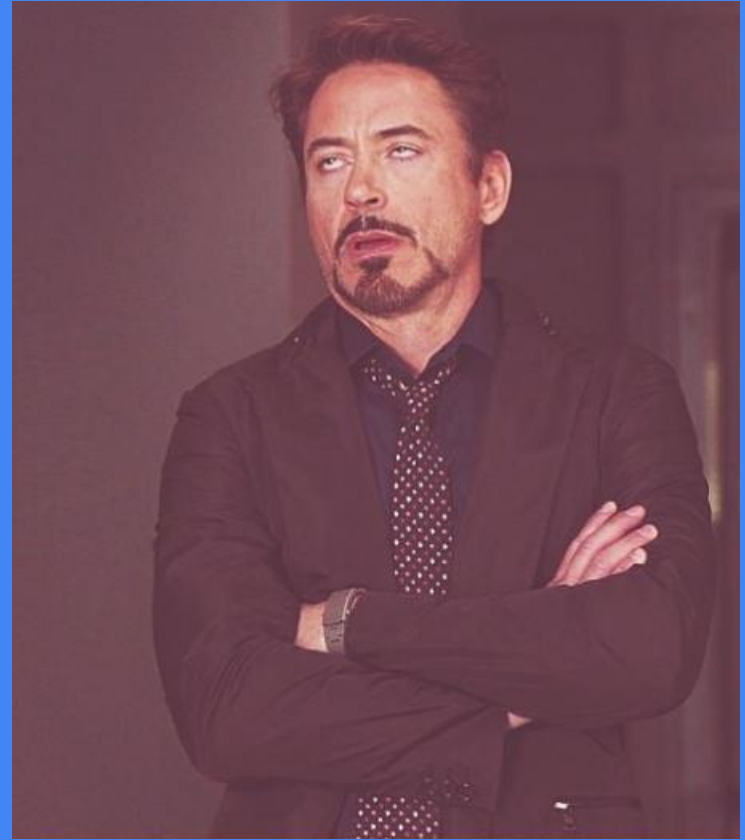
        String stringForYou = "kutya";
        stringForYou += "cica";
        System.out.println(stringForYou);
    }
}
```

out - JavaPlease (run) X

```
run:
Én 18 éves vagyok.
18 éves vagyok.
Ennyi éves vagyok: 18
kutyacica
BUILD SUCCESSFUL (total time: 0 seconds)
```



Beolvasás



```
package main;

import java.util.Scanner;





public class Main {

    public static void main(String[] args) {
        Scanner s = new Scanner(System.in);

        System.out.print("Írd be a születési évedet: ");
        int yearOfBirth = s.nextInt();

        int age = 2015 - yearOfBirth;
        System.out.println("Te "+age+" voltál 2015-ban.");
    }
}
```

Scanner osztállyal szinte bármilyen adatfolyamból képesek vagyunk beolvasni. Alapértelmezetten szóközig olvas.

<code>s.next()</code>		Karakterlánc beolvasása
<code>s.nextLine()</code>		Sor beolvasása
<code>s.nextInt()</code>		Egész szám beolvasása
<code>s.nextFloat()</code>		Szám beolvasása
....		stb.

Elágazások

Elágazások segítségével könnyen eldönthetjük hogy ha van egy feltételünk akkor annak megfelelően merre haladjunk tovább.

```
if (feltétel) {  
    ... utasítások  
    ....  
} else {  
    ....  
    ...ha nem teljesül  
    ...a feltétel  
}
```

```
if (f1 && f2) {  
    ... utasítások  
    ....  
}
```

```
if (f1 || f2) {  
    ... utasítások  
    ....  
}
```

```
if ((f1 && f2 && f3) || f4) {  
    ... utasítások  
    ....  
}
```

If-then-else

```
package main;

public class Main {

    public static void main(String[] args) {
        boolean isHuman = false;
        if (!isHuman) {
            System.out.println("You are a robot then.");
        } else {
            System.out.println("You are a human.");
        }

        int age = 18;
        if (age < 18) {
            System.out.println("You are way too young for this.");
        } else if (age > 18) {
            System.out.println("You are way too old for this.");
        } else {
            System.out.println("Your age doesn't matter anyway.");
        }
    }
}
```

Switch segítségével egy megadott érték alapján több ágra térhetünk ki.

```
switch (érték) {  
    case érték:  
        ....utasítások  
        break;  
    case másikeérték:  
        ....utasítások  
        break;  
}
```

That's a pretty good painting, right?



You just enjoyed one of
Adolf Hitler's artworks

Switch

```
public static void main(String[] args) {  
    int month = 2;  
  
    String monthName;  
    switch (month) {  
        case 1:  
            monthName = "január";  
            break;  
        case 2:  
            monthName = "február";  
            break;  
        default:  
            monthName = "hiba";  
    }  
    System.out.println(monthName);  
}
```

t - JavaPlease (run) X

```
run:  
február  
BUILD SUCCESSFUL (total time: 0 seconds)
```


Ciklusok




Ciklusok segítségével addig ismételhetjük az utasításokat amíg a feltétel igaz.

```
while (feltétel) {  
    ... utasítások  
    ....  
}
```

```
do {  
    ... utasítások  
    ....  
} while(feltétel);
```

```
for (inicializálás; feltétel; növelés) {  
    ... utasítások  
}
```

Példa



```
for (int i = 0; i < 10; i++) {  
    ... utasítások  
}
```

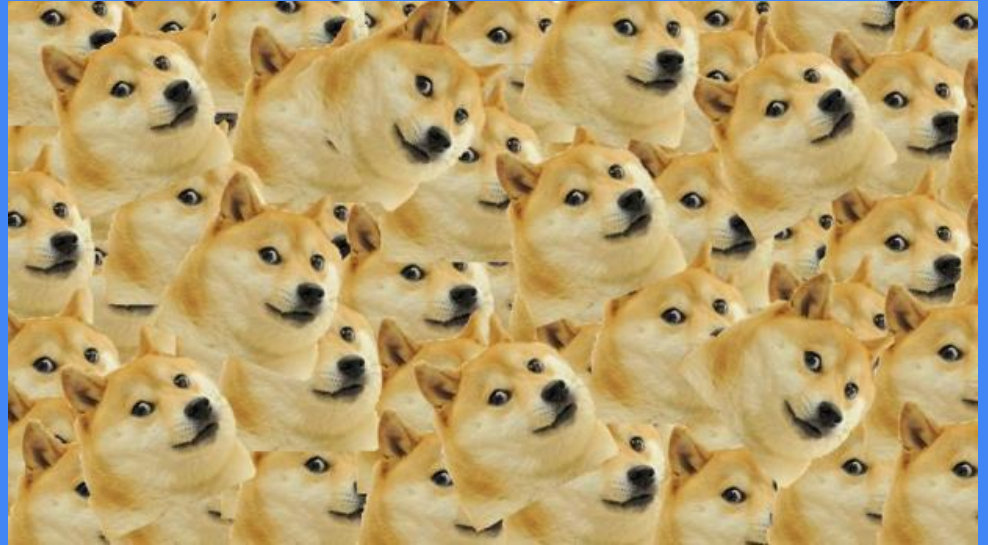
1 2 3 4 5 6 7 8 9 10

```
public static void main(String[] args) {  
    int i;  
  
    i = 1;  
    while (i < 11) {  
        System.out.print(" "+i+" ");  
        i++;  
    }  
  
    System.out.println();  
  
    i = 1;  
    do {  
        System.out.print(" "+i+" ");  
        i++;  
    } while (i < 11);  
  
    System.out.println();  
  
    for (i = 1; i < 11; i++) {  
        System.out.print(" "+i+" ");  
    }  
  
    System.out.println();  
}
```



```
run:  
1 2 3 4 5 6 7 8 9 10  
1 2 3 4 5 6 7 8 9 10  
1 2 3 4 5 6 7 8 9 10  
BUILD SUCCESSFUL (total time: 0 seconds)
```

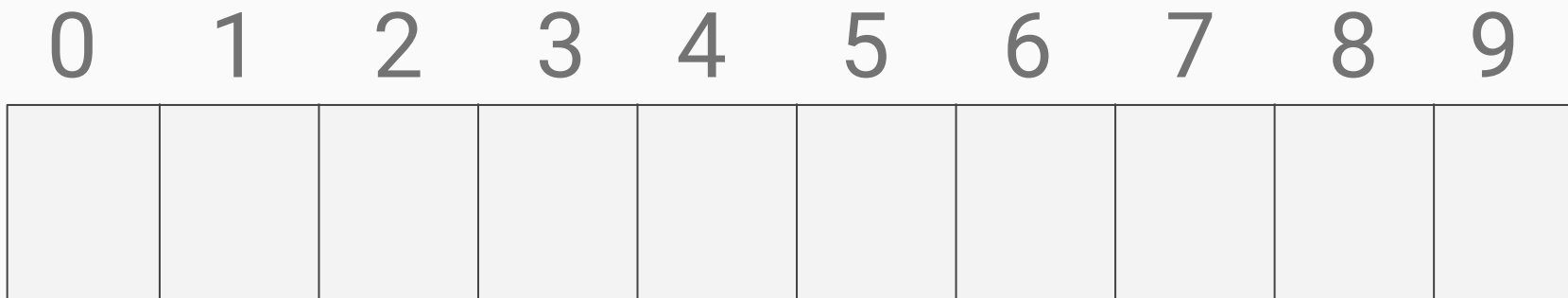
Tömbök



Tömbök segítségével **több adatot** tárolhatunk **egy változóban**.

Javában a tömbök **0-tól** indexelődnek.

Egy 10 elemű tömb



```
1 package main;
2
3 import java.util.Arrays;
4
5 public class Main {
6
7     public static void main(String[] args) {
8         int[] arrayForYou;
9         arrayForYou = new int[10];
10        arrayForYou[0] = 100;
11        arrayForYou[3] = 3;
12        arrayForYou[8] = 7;
13        System.out.println(Arrays.toString(arrayForYou));
14
15        int[] myGrades = { 1, 1, 1, 1, 1 };
16        myGrades[4] = 5;
17        System.out.println(Arrays.toString(myGrades));
18    }
19
20 }
```



```
run:
[100, 0, 0, 3, 0, 0, 0, 0, 7, 0]
[1, 1, 1, 1, 5]
BUILD SUCCESSFUL (total time: 0 seconds)
```

Osztály



A valóságban sokszor találkozol eltérő **tárgyakkal** de mind ugyanolyan **típusú**, **fajtájú**.

Például több ezer különféle bicikli van, de mind bicikli.
Több ezer **fajta** kutya van, de **mind** kutya.

Tehát egy **osztály** a **terve** a megadott **tárgynak**. Mind ugyanazon terv alapján készült és mind tartalmazza ugyanazokat az **összetevőket**.

Nem érted mi?

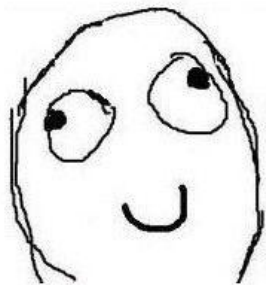
Akkor telefonos segítségkép
hívjuk fel Fintát



Just Kidding

Absztrakt osztály nem példányosítható de származtatható. Tehát önmaga csak egy tervként, alapként szolgál.

A **Pet** önmagában nem létezik mert nincs megadva mit tud, hiányos, absztrakt háziállat.



dafuq did i just read

Minden osztály, tulajdonság, interfész stb. rendelkezik egy láthatósággal, ez lehet: **semmi**, **public**, **protected**, **private**.

Modosító	Class	Package	Subclass	Világ
public				
protected				
semmi				
private				



A kenyér



JustVidman Életigazságai #2

**SOHA NE BÍZZ MEG A KENYÉRBEN,
INKÁBB FIGYELJ ODA A TANÁRRRA!**

Még akkor is, ha ő se nem házi jellegű, se nem szeletelt.
(Ha esetleg mégis szeletelt, gyorsan tárcsázd a mentőket!)

```
package javaplease.classtutorial;
```

```
public class Bread {
```

```
    public int price;
```

```
    public int weight;
```

```
    public boolean sliced;
```

```
    public Bread(int price, int weight, boolean sliced) {
```

```
        this.price = price;
```

```
        this.weight = weight;
```

```
        this.sliced = sliced;
```

```
    }
```

```
    public void slice() {
```

```
        this.sliced = true;
```

```
    }
```

```
}
```

osztály neve

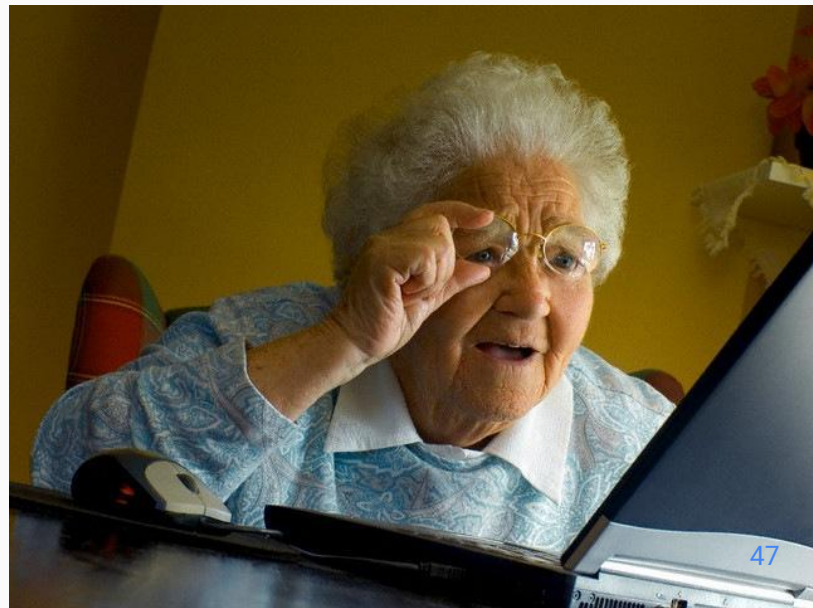
tulajdonságok, mezők

konstruktor

metódus

this

Ezzel a “változóval” elérhetjük a **jelenlegi példányt**. Tehát ha létrehoztunk egy **Bread példányt** és a **Bread osztályban** van **this** hivatkozás akkor az arra a **példányra** fog szólni amelyiket éppen **meghívtuk**.



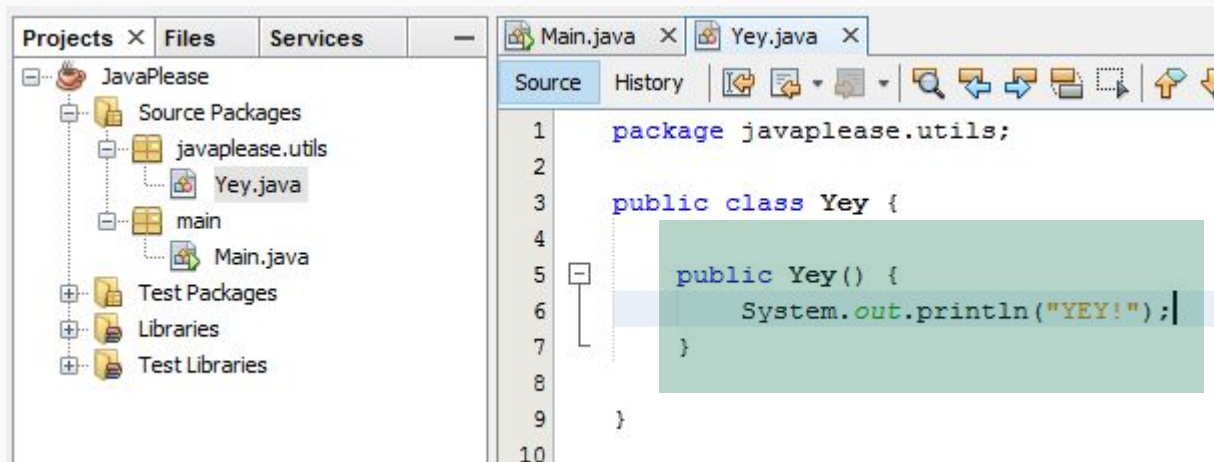
A metódusok rendelkeznek egy névvel, visszatérési értékkel és paraméterekkel.

```
public class Bread {  
  
    public int price;  
    public int weight;  
    public boolean sliced;  
  
    public Bread(int price, int weight, boolean sliced) {  
        this.price = price;  
        this.weight = weight;  
        this.sliced = sliced;  
    }  
  
    public void slice() {  
        this.sliced = true;  
    }  
}
```



láthatóság visszatérésiérték név(paraméterek)

Konstruktor akkor hívódik meg mikor létrehozunk egy **példányt** az adott **osztályból**. Ez a **metódus** mindig megegyezik az **osztály nevével** és **nincs visszatérési értéke**.



Kicsit mélyedjünk bele jobban....



