

Programozási nyelvek Java

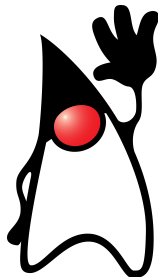
Alapok

Kozsik Tamás

A Java nyelv

- C-alapú szintaxis
- Objektumelvű (object-oriented)
 - ◇ Osztályalapú (class-based)
- Imperatív
 - ◇ Újabban kis FP-beütés
- Fordítás bájt kódra (a Java VM gépi kódjára)
- Erősen típusos
- Statikus + dinamikus típusrendszer
- Generikus, konkurens nyelvi eszközök

Java Language Specification



Jellemzői

- Könnyű/olcsó szoftverfejlesztés
- Gazdag infrastruktúra
 - ◇ Szabványos és egyéb programkönyvtárak
 - ◇ Eszközök
 - ◇ Kiterjesztések
 - ◇ Dokumentáció
- Platformfüggetlenség (JVM)
 - ◇ Write once, run everywhere
 - ◇ **Compile** once, run everywhere
- Erőforrásintenzív

JavaZone videó

Történelem

James Gosling és mások, 1991 (Sun Microsystems)

Java version history

- 1991: Oak → Green → **Java**
- 1996: Java 1.0 (SE, Standard Edition)
- 1999: Enterprise Edition (J2EE, Jakarta EE)
- 2010: a Java az Oracle-höz kerül
- SE LTS kiadások
 - ◊ Java 11 (2018)
 - ◊ Java 17 (2021)
 - ◊ Java 21 (2023)

Java Virtual Machine

- Alacsony szintű nyelv: bájtkód
- Sok nyelv fordítható rá (Ada, Closure, Eiffel, Jython, Kotlin, Scala...)
- Továbbfordítható
 - ◇ Just In Time compilation
- Dinamikus szerkesztés
- Kódmobilitás

Java Virtual Machine Specification

C és Java hasonlósága

```
// legnagyobb közös osztó  
int lnko(int a, int b) {  
    while (b != 0) {  
        int c = a % b;  
        a = b;  
        b = c;  
    }  
    return a;  
}
```

C és Java különbsége

```
double sum(double array[]) {  
    double s = 0.0;  
    for (int i = 0; i < array.length; ++i) {  
        s += array[i];  
    }  
    return s;  
}
```

C és Java különbsége - hangsúlyosabban

```
double sum(double[] array) {  
    double s = 0.0;  
    for (double item: array) {  
        s += item;  
    }  
    return s;  
}
```


Java programok felépítése

(első blikkre)

- [modul (module)]
- csomag (package)
- osztály (class)
 - ◇ adattag (mező, field)
 - ◇ metódus (method) vagy kicsit pontatlanul *függvény*
 - ▶ utasítás (statement)
 - kifejezés (expression)
 - ◇ literál

Tag (member): adattagok és metódusok összefoglaló neve.

Literál: érték megjelenése a forráskódban, pl. **123** vagy **"abc"**.

Java forrásfájl

- Osztálynévvel
- .java kiterjesztés
- Fordítási egység
- Csomagjának megfelelő könyvtárban
- Karakterkódolás

Hello World!

```
public class HelloWorld {  
    public static void main(String[] args) {  
        System.out.println("Hello world!");  
    }  
}
```

Parancssorban

```
$ ls
HelloWorld.java

$ javac HelloWorld.java

$ ls
HelloWorld.class  HelloWorld.java

$ java HelloWorld
Hello world!
```

Fordítás, futtatás

- A „tárgykód” a JVM bájtkód (.class)
- Nem szerkesztjük statikusan
- Futtatás: bájtkód interpretálása + JIT

Egyszerűsítések

Memóriába fordítás és onnan futtatás:

```
$ java HelloWorld.java
```

Hello world!

- Nem keletkezik .class fájl
- Csak a legegyszerűbb programokhoz javasolt

Egyszerűsítések

Memóriába fordítás és onnan futtatás:

```
$ java HelloWorld.java  
Hello world!
```

- Nem keletkezik .class fájl
- Csak a legegyszerűbb programokhoz javasolt

Bizonyos részek kihagyhatók (HelloWorld ilyenkor ún. *unnamed class*)

```
void main() {  
    System.out.println("Hello world!");  
}
```

Ehhez jelenleg extra opciókat kell megadni

```
$ javac.exe --source 21 --enable-preview HelloWorld.java  
$ java --enable-preview HelloWorld  
Hello world!
```

Java programok futása

- Végrehajtási verem (execution stack)
 - ◇ Aktivációs rekordok
 - ▶ Lokális változók
 - ▶ Paraméterátadás
- Dinamikus tárhely (heap)
 - ◇ Objektumok tárolása

Karakterkódolási szabványok

Karakterkódolások

- Bacon's cipher, 1605 (Francis Bacon)
- Baudot-code, 1874
- BCDIC, 1928 (Binary Coded Decimal Interchange Code)
- EBCDIC, 1963 (Extended ...)
- **ASCII**, 1963 (American Standard Code for Information Interchange)
- ISO/IEC 8859 (Latin-1, Latin-2, ...)
- Windows 1250 (Cp1250)
- **Unicode** (**UTF-8**, UTF-16, UTF-32)

Kapcsolódó program: `iconv` (Unix/Linux)

Karakterkódolás: fordítás

- Tfh a `Main.java` a régi magyar Windows-1250 kódolással íródott, de a rendszer a modern UTF-8 szabványt használja.

Karakterkódolás: fordítás

- Tfh a Main.java a régi magyar Windows-1250 kódolással íródott, de a rendszer a modern UTF-8 szabványt használja.

A rendszerben alapértelmezett UTF-8 kódolással

```
$ javac Main.java
Main.java:2: error: unmappable character (0xE1)
               for encoding UTF-8
String hib?s;
```

Karakterkódolás: fordítás

- Tfh a Main.java a régi magyar Windows-1250 kódolással íródott, de a rendszer a modern UTF-8 szabványt használja.

A rendszerben alapértelmezett UTF-8 kódolással

```
$ javac Main.java
Main.java:2: error: unmappable character (0xE1)
                for encoding UTF-8
String hib?s;
```

A kódolás kikényszerítése

```
$ javac -encoding Cp1250 Main.java
```

Lexikai elemek

- Kulcsszavak
- Azonosítók
- Operátorok
- Literálok
- Zárójelek: (.) [.] {.} <.>
- Speciális jelek: . , : ; -> | ... :: @
- Megjegyzések
 - ◇ Egysoros
 - ◇ Többsoros
 - ▶ Dokumentációs

Tárgvált kulcsszavak fenntartott szavak

Utasítás: `if else switch case default while do for`
`break continue return try catch finally throw assert yield`

Programszerkezet:

`package import class enum interface extends implements`

Típus: `boolean char byte short int long float double void`

Deklaráció:

`public protected private`
`abstract static final throws`

Hivatkozás: `this super`

Operátor: `instanceof new`

Literál: `true false null`

Nem tárgyalt kulcsszavak, fenntartott szavak

Típuskövetkeztetés: `var`

Deklaráció: `synchronized volatile transient strictfp native`

Nem használt, fenntartott szavak: `_ const goto`

Moduldeklaráció:

`module exports open opens provides requires
uses with to transitive`

Azonosítók

- Unicode betűk, számjegyek, _ és \$
 - ◊ Számjeggyel nem kezdődhet
- Néhány ritka, de lehetséges példa: 先生, ε, árvíztűrőTükörfúrógép

Konvenciók

```
package java.lang;  
public final class Integer ... {  
    ...  
    public static final int MAX_VALUE = 2147483647;  
    public int intValue() { ... }  
    ...  
}
```


Literálok

- Logikai ~: **true** és **false**
- Karakter~: `'c'`, `'\t'`, `'\''`, `'\\'`, `'\uBABA'`
 - ◇ Megjegyzés: `'0'` kódja 48, `'\0'` kódja 0
- Szöveg~: `"this string\nhas \u0032 lines"`
 - ◇ Megjegyzés: $32_{10} = 50$, ami a 2 karakterkódja
- Egész ~
 - ◇ **int**: 1984, 9_772_756, 0123, 0XBee, 0xCAFE_BABE, 0b1010101
 - ◇ **long**: 1984L, 1984l, 0xDEAD_BEEF_ADDED_COOL
- Lebegőpontos ~
 - ◇ **double**: 3.14159, .000_001, 1E-6, 6.022140857e23, 3., 3D, 3.14d, $0x1.Bp-2 = (1+11./16)/4$, 0X1DE.1POD
 - ◇ **float**: 3.14159F, .000_001f...



Kifejezés

- szintaxis: számít az operátorok aritása (paraméterszáma) és fixitása (sorrendje)
- kiértékelés
 - ◇ precedencia: $A + B * C$
 - ◇ asszociativitás: $A-B-C$ jelentése $(A-B)-C$, nem $A-(B-C)$
 - ◇ operandusok/paraméterek kiértékelésének sorrendje: $A + B$,
 $f(A, B)$
 - ◇ mohó/lusta (eager/lazy) kiértékelés: $A ? B : C$
 - ◇ mellékhatások (side effect): $++x$

Mellékhatás: példa: *olvasás a fájl végéig* (idióma)

Tíh in egy fájl olvasó `InputStream`.

```
int v;  
while ((v=in.read()) != -1) {  
    ...  
}
```

A ciklusfeltétel két mellékhatást tartalmaz.

- `read()`: a fájl olvasása halad
- `v`: új értéket kap

Bitmanipuláció

- Bitenkénti és és *vagy*: $A \& B$, $A | B$ (A és B típusa **int** vagy **long**)

Bitmanipuláció

- Bitenkénti és és *vagy*: $A \& B$, $A | B$ (A és B típusa **int** vagy **long**)
- Kizáró vagy (XOR): $A \wedge B$

Bitmanipuláció

- Bitenkénti és és *vagy*: $A \& B$, $A | B$ (A és B típusa **int** vagy **long**)
- Kizáró vagy (XOR): $A \wedge B$
- Negáció: $\sim A$

Bitmanipuláció

- Bitenkénti és és vagy: $A \& B$, $A \mid B$ (A és B típusa **int** vagy **long**)
- Kizáró vagy (XOR): $A \wedge B$
- Negáció: $\sim A$
- Eltolás (shift): $A \ll B$, $A \gg B$, $A \ggg B$

Új operátor készítése, operátortúlterhelés

- Nem készíthető új operátor
- Az előre adott operátorok jelentése nem változtatható meg
 - ◇ A programozó nem definiálhat operátortúlterhelést
 - ◇ A beépített túlterhelések köre (pl. + vagy &) korlátozott

Logikai operátorok kiértékelése

A és B (rész)kifejezések típusa boolean

- Lusta: A && B, A || B
- Mohó: A & B, A | B

Lusta és mohó kiértékelési tábla

A logikai kifejezések kiértékelésének négy lehetséges eredménye:

true (\uparrow), **false** (\downarrow), *kivétel* (\perp) és *végtelen ciklus* (∞).

A mellékhatásoktól eltekintve a $\alpha \wedge \beta$ kifejezés értéke:

$\alpha \ \&\& \ \beta$	$\beta = \uparrow$	$\beta = \downarrow$	$\beta = \perp$	$\beta = \infty$
$\alpha = \uparrow$	\uparrow	\downarrow	\perp	∞
$\alpha = \downarrow$	\downarrow	\downarrow	\downarrow	\downarrow
$\alpha = \perp$	\perp	\perp	\perp	\perp
$\alpha = \infty$	∞	∞	∞	∞

$\alpha \ \& \ \beta$	$\beta = \uparrow$	$\beta = \downarrow$	$\beta = \perp$	$\beta = \infty$
$\alpha = \uparrow$	\uparrow	\downarrow	\perp	∞
$\alpha = \downarrow$	\downarrow	\downarrow	\perp	∞
$\alpha = \perp$	\perp	\perp	\perp	\perp
$\alpha = \infty$	∞	∞	∞	∞

Példa: mohó kiértékelésű operátor

```
int v1, v2;
while (((v1 = in1.read()) != -1) | ((v2 = in2.read()) != -1))
    if (v1 == -1) {
        out.write(v2);
    } else if (v2 == -1) {
        out.write(v1);
    } else {
        out.write(v1+v2);
    }
}
```

Utasítások

- Kifejezéskiértékelő utasítás
 - ◇ Értékadások
 - ◇ Metódushívás
- **return**-utasítás és **yield**-utasítás
- Elágazások (**if**, **switch**)
- Ciklusok (**while**, **do-while**, **for**)
- Nem strukturált: **break**, **continue**
- Blokk-utasítás
- Deklaráció (pl. változó~)
- Kivételkezelő és -kiváltó utasítások
- **assert**-utasítás

Puzzle 22: Dupe of URL (Bloch, Gafter: Java Puzzlers)

Címkézett utasítás

```
class Main {  
    public static void main(String[] args) {  
        https://jdk.java.net/  
        System.out.println();  
    }  
}
```

switch-utasítás

```
switch (day) {                                // enumeration type
    case SUN: case SAT: return 0;
    case FRI:      return 6;
    default:       return 8;
}
```

```
switch (day.toString()) { // String
    case "SUN": case "SAT": return 0;
    case "FRI":      return 6;
    default:       return 8;
}
```

```
switch (day.ordinal()) { // int
    case 6: case 5:      return 0;
    case 4:      return 6;
    default:      return 8;
}
```

Hagyományos switch-utasítás

```
String name;    // szándékosan nem inicializált
switch (dayOf(new java.util.Date())) {
    case 0: name = "Sunday"; break;
    case 1: name = "Monday"; break;
    case 2: name = "Tuesday"; break;
    case 3: name = "Wednesday"; break;
    case 4: name = "Thursday"; break;
    case 5: name = "Friday"; break;
    case 6: name = "Saturday"; break;
    default: throw new Exception("illegal value");
}
```

Biztonságosabb switch-utasítás

```
String name;    // szándékosan nem inicializált
switch (dayOf(new java.util.Date())) {
    case 0 -> name = "Sunday";
    case 1 -> name = "Monday";
    case 2 -> name = "Tuesday";
    case 3 -> name = "Wednesday";
    case 4 -> name = "Thursday";
    case 5 -> name = "Friday";
    case 6 -> name = "Saturday";
    default -> throw new Exception("illegal value");
}
```


switch kifejezés

```
String name =  
    switch (dayOf(new java.util.Date())) {  
        case 0 -> "Sunday";  
        case 1 -> "Monday";  
        case 2 -> "Tuesday";  
        case 3 -> "Wednesday";  
        case 4 -> "Thursday";  
        case 5 -> "Friday";  
        case 6 -> "Saturday";  
        default -> throw new Exception("illegal value");  
    };
```

Túlcsorgás

```
switch (month) {  
    case 4:  
    case 6:  
    case 9:  
    case 11: days = 30;  
             break;  
    case 2: days = 28 + leap;  
             break;  
    default: days = 31;  
}
```

```
days = switch (month) {  
    case 4,6,9,11 -> 30;  
    case 2 -> 28 + leap;  
    default -> 31;  
};
```

yield-utasítás

```
int days =  
    switch (month) {  
        case 4, 6, 9, 11 -> 30;  
        case 2 -> { int leap = 0;  
                    if (year % 4 == 0)    leap = 1;  
                    if (year % 100 == 0) leap = 0;  
                    if (year % 400 == 0) leap = 1;  
                    yield 28 + leap;  
                }  
        default -> 31;  
    };
```

Nem triviális túlcsorgás

```
enum States {RED, AMBER, GREEN};  
...  
switch (trafficLight) {  
    case RED:    stop();  
                break;  
    case AMBER:  if (canSafelyStop()) {  
                    stop();  
                    break;  
                }  
    case GREEN:  go();  
}
```

Javában nem, de C-ben ilyen is írható

```
switch (trafficLight) {  
    case AMBER:    if (canSafelyStop()) {  
    case RED:      stop();  
                  break;  
    }  
    case GREEN:   go();  
}
```

Csomag

- Program tagolása
- Összetartozó osztályok összefogása
- Programkönyvtárak
 - ◇ Szabványos programkönyvtár

A package utasítás

```
package geometry;  
  
public class Point {    // geometry.Point  
    int x, y;  
    void move(int dx, int dy) {  
        x += dx;  
        y += dy;  
    }  
}
```

- Osztály (teljes) neve: geometry.Point
- Osztály rövid neve: Point

Hierarchikus névtér

```
package geometry.basics;
```

```
public class Point {    // geometry.basics.Point
    int x, y;
    void move(int dx, int dy) {
        x += dx;
        y += dy;
    }
}
```

- Szabványos programkönyvtár, pl. `java.net.ServerSocket`
- `hu.elte.kto.teaching.javabsc.geometry.basics.Point`

Compilation and execution

- Munkakönyvtár
(working directory)
- Hierarchikus csomagszerkezet
→ könyvtárszerkezet
- Fordítás a munkakönyvtárból
 - ◇ Fájlnev teljes elérési úttal
- Futtatás a munkakönyvtárból
 - ◇ Teljes osztálynév

```
$ ls -R
.:
geometry

./geometry:
basics

./geometry/basics:
Main.java  Point.java
$ javac geometry/basics/*.java
$ ls geometry/basics
Main.class  Main.java
Point.class Point.java
$ java geometry.basics.Main
$
```

Fordítás: Java és C

```
$ ls geometry/basics
Main.java  Point.java
$ javac geometry/basics/Point.java
$ ls geometry/basics
Main.java  Point.class  Point.java
$ javac geometry/basics/Main.java
$ ls geometry/basics
Main.class  Main.java  Point.class  Point.java
$ java geometry.basics.Main
$
```

Rekurzív fordítás

```
$ ls geometry/basics
Main.java  Point.java
$ javac geometry/basics/Main.java
$ ls geometry/basics
Main.class Main.java Point.class Point.java
$ java geometry.basics.Main
$
```

Névtelen csomag

Default/anonymous package

- Ha nem írunk package utasítást
- Forrásfájl közvetlenül a munkakönyvtárba
- Kis kódbázis esetén rendben van

Láthatósági kategóriák

- **private** (privát, rejtett)
 - ◇ csak az osztálydefiníción belül
- semmi (félnyilvános, package-private)
 - ◇ csak az ugyanabban a csomagban lévő osztálydefiníciókban
- **public** (publikus, nyilvános)
 - ◇ osztály is
 - ◇ tagok, konstruktor is

Nyilvános és rejtett tagokat tartalmazó nyilvános osztály

```
package hu.elte.kto.javabsc.eloadas;

public class Time {
    private int hour;           // 0 <= hour < 24
    private int minute;        // 0 <= minute < 60
    public Time(int hour, int minute) { ... }
    public int getHour() { return hour; }
    public int getMinute() { return minute; }
    public void setHour(int hour) { ... }
    public void setMinute(int minute) { ... }
    public void aMinutePassed() { ... }
}
```

Több csomagból álló program

hu/elte/kto/javabsc/eloadas/Time.java

```
package hu.elte.kto.javabsc.eloadas;  
  
public class Time {  
    ...  
}
```

Main.java

// a névtelen csomagban

```
public class Main {  
    public static void main(String[] args) {  
        hu.elte.kto.javabsc.eloadas.Time morning = new Time(6,10);  
        // fordítási hiba: nincs Time a névtelenben ↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑  
    }  
}
```

Egy csomagon belül

```
hu/elte/kto/javabsc/eloadas/Time.java
```

```
package hu.elte.kto.javabsc.eloadas;
```

```
public class Time { ... }
```

```
hu/elte/kto/javabsc/eloadas/Main.java
```

```
package hu.elte.kto.javabsc.eloadas;
```

```
public class Main {  
    public static void main(String[] args) {  
        Time morning = new Time(6,10);  
        ...  
    }  
}
```


Egy forrásfájlban több típusdefiníció

```
hu/elte/kto/javabsc/eloadas/Time.java
```

```
package hu.elte.kto.javabsc.eloadas;

public class Time {
    ...
}

class Main {
    public static void main(String[] args) {
        Time morning = new Time(6,10);
        ...
    }
}
```

Az import utasítás

```
hu/elte/kto/javabsc/eloadas/Time.java
```

```
package hu.elte.kto.javabsc.eloadas;
```

```
public class Time {  
    ...  
}
```

Main.java

```
import hu.elte.kto.javabsc.eloadas.Time;
```

```
public class Main {  
    public static void main(String[] args) {  
        Time morning = new Time(6,10);  
        ...  
    }  
}
```

Minősített név feloldása

- Osztály teljes neve helyett a rövid neve
- `import hu.elte.kto.javabsc.eloadas.*;`
- Nem tranzitív
- A `java.lang` csomag típusait nem kell
- Névütközés: teljes név kell
 - ◇ `java.util.List`
 - ◇ `java.awt.List`

Fordítási egység szerkezete

- opcionális package utasítás
- 0, 1 vagy több import utasítás
- 1 vagy több típusdefiníció

javac kapcsolók

`-d <directory>`

Specify where to place generated class files

`--source-path <path>, -sourcepath <path>`

Specify where to find input source files

`--class-path <path>, -classpath <path>, -cp <path>`

Specify where to find user class files...

Classpath

```
javac -classpath ./usr/lib/java:/opt/java/myfiles.jar \\  
geometry/basics/Point.java
```

```
java -classpath ./usr/lib/java:/opt/java/myfiles.jar \\  
geometry.basics.Main
```

- Ha kell a colors.RGB osztály:
 - ◇ ./colors/RGB.class
 - ◇ /usr/lib/java/colors/RGB.class
 - ◇ /opt/java/myfiles.jar-ban colors/RGB.class
- Windows alatt (cmd): -cp
.;C:\Users\kto\mylib;D:\myfiles.jar
 - ◇ Powershell használatával: -cp
".;C:\Users\kto\mylib;D:\myfiles.jar"
 - ◇ CLASSPATH környezeti változó

jar fájlok

- Java Archive
- ZIP-tömörítésű fájl
- jar parancs az SDK-ban

Egységteszt (Unit test)

- A program legkisebb, önálló részeinek kipróbálása
 - ◇ Egység lehet: metódus, **osztály**, komponens/modul
 - ◇ Nem egységteszt, ha külső függőségei vannak
 - ▶ Ilyen pl.: fájlrendszer, adatbázis, hálózat használata
- Kis, gyorsan lefutó, független tesztek
 - ◇ Futási időben működik
 - ◇ Fekete dobozos: az egység belső szerkezete nem ismert
 - ▶ Csak az osztály publikus interfészét (metódusait) használja
- Funkcionális helyességet tesztel: a lefutás az elvárt eredményt adja-e
 - ◇ Nem cél: hatékonyság tesztelése

Egységteszt: helyesség

- Nem *bizonyítja*, csak *alátámasztja* a helyességet
- Regressziók felfedése: hamar kiderül, ha hibás a kód
- Egyúttal dokumentálja, mi az elvárt működés
 - ◇ Együtt fejlődik a kóddal: ezt a fordítóprogram „érti” és ellenőrzi
 - ◇ A szöveges dokumentáció elavulhat
- Lefedettség (code coverage)
- Sok hibát megelőz még fejlesztés alatt
 - ◇ Nagyobb munkaigény kezdetben
 - ◇ Olcsóbb lehet az utólagos hibajavításnál
 - ▶ Az éles rendszer jobban működik

Egységteszt: módszerek

- Tesztvezérelt fejlesztés (test driven development, TDD)
 1. Új teszteset hozzáadása, ami még „piros” (sikertelen)
 2. Kód írása/fejlesztése: minden teszteset legyen „zöld” (sikeres)
 3. A kód minőségének javítása (refaktorálás): minden „zöld”
- Egyéb tesztelési megközelítések
 - ◇ Naplózás, kiírások használata
 - ◇ Hibakeresés (debugging)
 - ◇ Összetettebb: integrációs ~, teljesítmény~, stressz/terhelési ~, automatizált ~, véletlenített/tulajdonság alapú ~, mock ~, folyamatos ~ (CI/CD), ...
 - ◇ Felhasználói élmény: elfogadási ~, biztonsági ~, használati ~, lokalizációs ~, ...
 - ◇ Formális helyességbizonyítás

Egységtesztelő: így használandó

- Egy tesztelő metódus egyetlen vizsgálatot tartalmaz
- A lehető legegyszerűbb szerkezet: ciklus, elágazás, véletlen, ... nélkül
- Saját kódot teszteljük, ne könyvtárakat
- Lebegőpontos típusok tesztelése: az eredménynek lehet pontatlansága
 - ◇ Extra paraméter: tűréshatár (delta)
- Számítás adatainak struktúrája: egyszerűtől bonyolultig
 - ◇ `null`
 - ◇ üres szöveg, `0`
 - ◇ konstruktorhívás, majd getter
 - ◇ kis, pozitív értékek
 - ◇ egy-két lépéssel összeállított adatok
 - ◇ negatív/szokatlan/extrém értékek
 - ▶ pl. `Integer.MAX_VALUE` vagy `Double.MIN_VALUE`
 - ▶ kivételek
 - ◇ hosszabb „történet” , több hívással

Egységtesztelés: FIRST

- Fast: μs -ms
- Isolated: egymástól és külvilágtól elkülönülő
- Repeatable: megismételhető
 - ◇ Nincsenek mellékhatások
 - ◇ Nincs nemdeterminisztikus futás
- Self-verifying: önellenőrző
 - ◇ Minden teszt elbukhat
 - ◇ Minden bukásnak pontosan egy oka lehet
- Timely: a kóddal együtt bővülnek/fejlődnek a tesztek
- vagy Thorough: lásd előző fólia

JUnit

- Java nyelvű megvalósítások közül a legnépszerűbb
- A jelenleg legújabb kiadás: JUnit 5, 1.11.1 verzió
- Ide kattintva letölthető a jar fájl
 - ◇ A letöltött fájlnak adható rövidebb név, pl. junit5all.jar
- Tesztelendő osztály: system under test (SUT)
 - ◇ Tegyük fel, hogy a `time.Time` osztályt teszteljük
 - ◇ A SUT kódja a `time/Time.java` fájlban van
 - ◇ A tesztelő kód a `time/TimeTest.java` fájlba kerül

Fordítás és futtatás

```
javac -cp junit5all.jar time/TimeTest.java
java -jar junit5all.jar -cp . -c time.TimeTest
```

JUnit teszteset: Arrange-Act-Assert

```
package time;

import static org.junit.jupiter.api.Assertions.*;
import org.junit.jupiter.api.Test;

public class DemoTest {
    @Test
    void testHour00_00() {
        // Step 1: Arrange
        Time sut = new Time(0, 0);
        // Step 2: Act
        int hour = sut.getHour();
        // Step 3: Assert
        assertEquals(0, hour);
    }
}
```

JUnit teszteset: Arrange-Act-Assert röviden

```
package time;

import static org.junit.jupiter.api.Assertions.*;
import org.junit.jupiter.api.Test;

public class DemoTest {
    @Test
    void testHour00_00() {
        assertEquals(0, new Time(0, 0).getHour());
    }
}
```

JUnit teszteset kimenete

- Fontos: az elvárt érték az első paraméter
 - ◇ Ez mindig egy konstans legyen, ne számított érték

@Test

```
void wrongResultTest() { assertEquals(5, 2+2); }
```

```
org.opentest4j.AssertionFailedError: expected:<5> but was:<4>
... (many unimportant lines)
at testing.DemoTest.wrongResultTest(DemoTest.java:9)
... (even more unimportant lines)
```


JUnit teszteset kimenete

- Fontos: az elvárt érték az első paraméter
 - ◇ Ez mindig egy konstans legyen, ne számított érték

@Test

```
void wrongResultTest() { assertEquals(5, 2+2); }
```

```
org.opentest4j.AssertionFailedError: expected:<5> but was:<4>
... (many unimportant lines)
at testing.DemoTest.wrongResultTest(DemoTest.java:9)
... (even more unimportant lines)
```

@Test

```
void wrongOrderTest() { assertEquals(2+2, 5); }
```

```
org.opentest4j.AssertionFailedError: expected:<4> but was:<5>
at testing.DemoTest.wrongOrderTest(DemoTest.java:9)
```

A JUnit szószátyár kimenetének visszafogása

Segítő eszköz: `check.cmd`

- A Teams-csoport Általános (General) csatornáján, a Fájlok között érhető el egy zip fájlban
- A `javac` és `java` parancsait futtatja
 - ◇ Windows (`cmd`, `powershell`), Linux, Mac környezetekben
 - ◇ Bevet néhány trükköt is ezen felül
 - ◇ Csak a kimenet érdemi részeit mutatja
- CheckThat szerkezetellenőrzést is futtat (erről bővebben később)
- További részletekért futtatható paraméterek nélkül
 - ◇ Valamit lásd a demót

JUnit: tömbök

Tömbök tesztelése: külön `assertArrayEquals` művelettel

- `assertEquals` nem jó
- Más adatszerkezetek jól működnek

```
@Test
```

```
public void testFibArray() {  
    int[] fibs = Fibonacci.fibsUpTo(6);  
    assertEquals(new int[] { 1, 1, 2, 3, 5, 8 }, fibs);  
}
```

JUnit: paraméterezett teszt: azonos működés több adaton

```
@CsvSource("this is some text,4")
@ParameterizedTest
public void testSplit(String text, int partCount) {
    assertEquals(partCount, text.split(" ").length);
}
```

```
@DisplayName("Computing the Fibonacci numbers")
@ParameterizedTest(name = "fib({0}) = {1}")
@CsvSource({"13,6", "21,7"})
public void testFib(int expected, int num) {
    assertEquals(expected, Fibonacci.fib(num));
}
```

```
'-- Computing the Fibonacci numbers [OK]
+-- fib(6) = 13 [OK]
+-- fib(7) = 21 [OK]
```

JUnit: paraméterezett tesztek szövegblokkokkal

Forrás: JUnit 5 dokumentációja

```
@ParameterizedTest(name = "[{index}] {arguments}")
@CsvSource(useHeadersInDisplayName = true, textBlock = """
    FRUIT,          RANK
    apple,          1
    strawberry,     700_000
    'lemon, lime',  0xF1
    """)

public void testWithCsvSource(String fruit, int rank) {
    // ...
}
```

Kimenet:

- [1] FRUIT = apple, RANK = 1
- [2] FRUIT = strawberry, RANK = 700_000
- [3] FRUIT = lemon, lime, RANK = 0xF1

JUnit: ritkábban használatos eszközök

```
fail();
```

```
assertEquals("y", "x", "expected to be y");
```

```
assertEquals("y", "x", () -> "Also expected to be y");
```

```
... AssertionError
```

```
    at time.JUnitDemoTest.testFail(JUnitDemoTest.java:19)
```

```
...: expected to be y ==> expected: <y> but was: <x>
```

```
    at time.JUnitDemoTest.testMessageV1(JUnitDemoTest.java:24)
```

```
...: Also expected to be y ==> expected: <y> but was: <x>
```

```
    at time.JUnitDemoTest.testMessageV2(JUnitDemoTest.java:29)
```

JUnit: ritkábban használatos eszközök

```
@Test
```

```
public void testTrue() {  
    assertTrue(2 + 2 == 4);  
}
```

```
@Test
```

```
public void testFalse() {  
    assertFalse("it's true" == "it's " + true);  
}
```

```
...: expected: <false> but was: <true>  
    at time.JUnitDemoTest.testFalse(JUnitDemoTest.java:14)
```

- Az assertEquals jobb: precízebb a hibaüzenet
- Figyelem: a == nem helyes egyenlőségvizsgálat a **String** típuson!
 - ◊ Az ellenpárja, != szintén rossz



JUnit: kivételek

@Test

```
public void testInvalidTime() {  
    InvalidTimeException exception =  
        assertThrows(InvalidTimeException.class, () -> {  
            new Time(123, 456);  
        });  
    assertEquals("/ by zero", exception.getMessage());  
}
```

- `() -> { ... }`: a kivételt potenciálisan kiváltó kódrészlet ide kerül
- A `.class` tekinthető speciális adattagnak
- Itt megengedett két `assertX` írása is egy tesztelő metódusba
 - ◊ Sokszor nincs üzenet, akkor változó sem szükséges

JUnit: életciklus

```
public class TimeTest {  
    private Time time;  
  
    @BeforeEach  
    public void beforeEach() {  
        time = new Time(12, 34);  
    }  
  
    @Test void test1() { assertEquals(12, time.getHour()); }  
    @Test void test2() { assertEquals(34, time.getMin()); }  
    @Test void test3() { assertEquals(35, time.inc().getHour()); }  
}
```

- **@BeforeEach**: tesztesetek ismétlődő adatainak közös beállítása
 - ◇ A tesztesetek nem zavarják egymást, mert mindig újrainicializál
- **@AfterEach**: pl. átmeneti fájlok törlésére
- **@BeforeAll**, **@AfterAll**: ritkán használatos

CheckThat

- A szokásos JUnit tesztek a kód funkcionalitását vizsgálják
- Ez az eszköz a kód szerkezetét ellenőrzi
- Használata intuitív
- A megvalósító kód túlmutat a félév anyagán, nem kell megérteni

CheckThat példa

```
package time;

import static check.CheckThat.Condition.*;
import check.CheckThat;

import org.junit.jupiter.api.Test;

public class TimeStructureTest {
    @Test
    public void test1() {
        CheckThat...
    }

    ...
}
```

CheckThat példa

```
CheckThat.theClass("time.Time")
    .thatIs(FULLY_IMPLEMENTED, INSTANCE_LEVEL, VISIBLE_TO_ALL)
    .hasFieldOfType("hour", int.class)
        .thatIs(FULLY_IMPLEMENTED, INSTANCE_LEVEL, MODIFIABLE, VISI
        .thatHas(GETTER, SETTER);

it.hasConstructorWithParams(int.class, int.class)
    .thatIs(VISIBLE_TO_ALL);

it.hasMethodWithParams("getEarlier", "Time")
    .thatIs(FULLY_IMPLEMENTED, INSTANCE_LEVEL, VISIBLE_TO_ALL)
    .thatReturns("Time");
```

CheckThat hibaüzenetek

```
org.opentest4j.MultipleFailuresError: Multiple Failures (1 fail  
...: Nincsen megfelelő GETTER metódus  
           ehhez az adattaghoz: Time.hour
```

További üzenetek:

- ```
...: A Time.hour visszatérése nem megfelelő
...: A Time.hour láthatósága nem megfelelő
```
- Egy változóval angolra is állítható

## CheckThat használata

```
package time;
import org.junit.platform.suite.api.*;

@Suite
@SelectClasses({
 TimeStructureTest.class,
 WorldTimesStructureTest.class
 ,TimeTest.class
 ,WorldTimesTest.class // <-- ez a sor kikommentezendő
}) // ha WTT még nincs megvalósítva
public class TimeTestSuite {}
```

### Fordítás és futtatás

```
check.cmd time/TimeTestSuite.java time.TimeTestSuite
```

## CheckThat használata, elkülönülő tesztelő kód

| root                           | root            | Kimenet                    |
|--------------------------------|-----------------|----------------------------|
| + project                      | + junit5all.jar | '-- TimeTestSuite [OK]     |
| + src                          | + check*.jar    | '-- TimeStructureTest [OK] |
| + time                         |                 | +-- test1() [OK]           |
| + Time.java                    |                 | +-- test2() [OK]           |
| + test                         |                 | '-- test3() [OK]           |
| + time                         |                 |                            |
| + TimeStructureTest.java       |                 |                            |
| + WorldTimesStructureTest.java |                 |                            |
| + TimeTestSuite.java           |                 |                            |
| + TimeTest.java                |                 |                            |

- Továbbra is ugyanabban a csomagban van a SUT és a tesztelő

## Fordítás és futtatás

| root      | root            | Output                     |
|-----------|-----------------|----------------------------|
| + project | + junit5all.jar | '-- TimeTestSuite [OK]     |
| + src     | + check*.jar    | '-- TimeStructureTest [OK] |
| + time    |                 | +-- test1() [OK]           |