01.02. Java és C# nyelvi jellemzői

Monday, February 8, 2021 1:08 PM

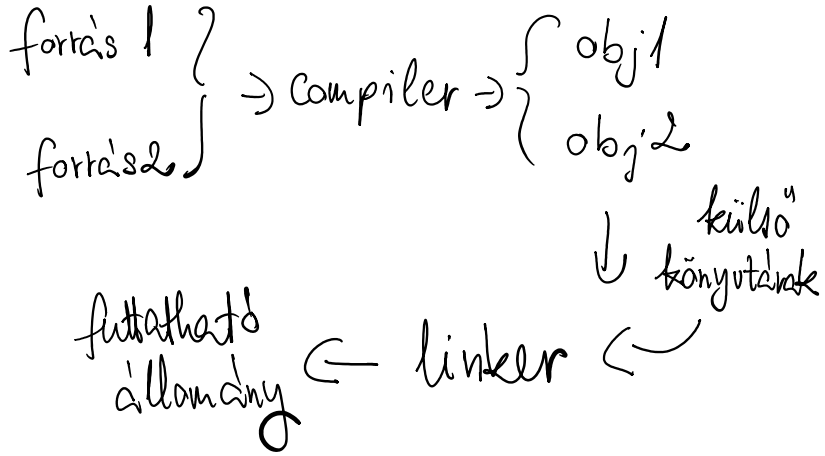
C++ alapú szintaktika

de

- mutatók helyett referencia-változók (C#-ban mindkettő)
- nincs többszörös öröklés
- Java: nincs operátor overloading

Fordítás:

- hagyományos: C, C++

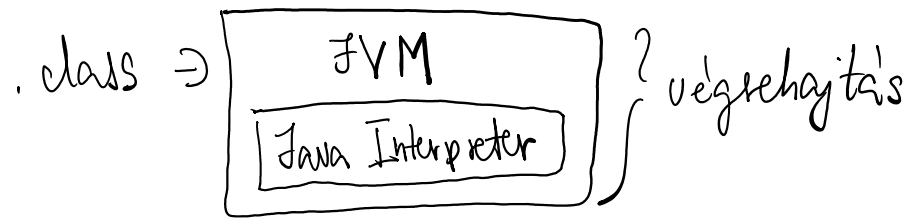


- Java interpretálás

.java → Java compiler → .class

platformfüggetlen bytecode

.java → JVM



↳ a futtató gépen telepítve kell, hogy legyen a JVM

- Just in Time (JIT) fordítás (C#)

forrás ⇒ MSIL (intermediate language)

↓ JIT

futtatás

közvetlenül használat előtt fordul

Környezet:

Java:

JDK + min. editor, általában IDE

JDK részei:

javac, java, jdb (debugger),
javadoc, jar

+JRE:

JVM + platform könyvtárak

Program felépítése:

Java:

```
public class Hello {
    pub. st. void main(String[] args) {
        System.out.println("...");
    }
}
```

C#:

```
using System;
namespace ProjectNeve {
    class Test {
        p. s. v. Main(string[] args) {
            Console.WriteLine("...");
        }
    }
}
```

Output:

Java:

- A `System` osztály a `java.lang` csomagban van definiálva. Ezt a csomagot nem kell importálni, a java automatikusan megteszi (alapértelmezett csomag). 21

Nyelvi elemek:

- azonosítók
  - lehetőleg angol nyelvűek
  - betűk, számok, _, $
    - nem az elején
  - C# - metódus is nagybetűvel kezdődik
    - interface elején „I" betű
- megjegyzések

  // egysoros

  /* többsoros */

  Java: /** dokumentációs */

  C#: /// dokumentációs
- egyszerű típusok

  Java: kisbetűsek,

  mindegyikhez wrapper (csomagoló) osztály

  C#: szintén kisbetűsek
  - itt vannak előjelesek, nem előjelesek (Java-ban nincs)

- literálok:
  C-vel teljesen megegyezőek
  + true, false, null
    logikai referencia literál
    └ nem egész típusúak!

- operátorok:
  Java-ban eddig nem tanultak:
  new: dinamikus memóriában foglal!
  instanceof: típus vizsgálat

  C#:
  new, typeof, is

- típuskonverzió:
  - numerikusoknál bővebb-be automatikus
  - long -> float nem automatikus
    mert adatvesztés lehet
  - numerikus és boolean között
    egyik irányban sincs automatikus konverzió

- tömb:
  - primitív vagy objektum elemek

is tárolhatók (tömbök is)

- típus[] tömbnév = new típus[méret];

- többdim: (kb.)
  - így kell elképzelni, mint C-ben a mutatótömbök
  - "tömbök tömbje"
  - csak egy dim. méretét kötelező megadni

- Tömb létrehozási lehetőségek C#-ban:

```
// Create an array of four elements, and add values later
string[] cars = new string[4];

// Create an array of four elements and add values right away
string[] cars = new string[4] {"Volvo", "BMW", "Ford", "Mazda"};

// Create an array of four elements without specifying the size
string[] cars = new string[] {"Volvo", "BMW", "Ford", "Mazda"};

// Create an array of four elements, omitting the new keyword, and
string[] cars = {"Volvo", "BMW", "Ford", "Mazda"};
```

- Java-ban az 1., 3. és 4. szintaktika megengedett

Tömbelemek elérése C# 8.0-tól:

- Hátulról, 1-től indexelve méretig: `cars[^1]`

  Pl. Tömb másolása fordítva, egy indexet használva:

```
int[] n = {1, 2, 3, 4, 5};
int[] m = new int[5];
for ( int i = 0; i < 5; i++ ) {
  m[i] = n[^(i+1)];
}
```

- Több elem elérése egy utasításban. Az index intervallum alulról zárt, felülről nyitott.
  - `cars[1..3]` → BMV, Ford
  - `cars[2..]` → Ford, Mazda
  - `cars[..3]` → Volvo, BMV, Ford

(+) struct csak C#-ban van!