Ütemterv

# Egyszerű típusok

Alapok ismétlése típusok területén:

* számábrázolás: egész számok (előjeles, előjel nélküli), lebegőpontos számok
* boolean/logikai típus: más nyelvek esetén hogy van, és hogy Javában
* módosítók: final, const, static, volatile

# Összetett típusok

Alapok ismétlése továbbra is:

* enum
* tömbök
* sztringek: karakterkódolások, sztring más nyelvekben
* bitműveletek

# Program

Avagy mi az, amit csinálunk tulajdonképpen

* Java program felépítése
* Manifest
* külső könyvtárak
* Fordítás
* Java Virtual Machine működése (elővetítés a memóriakezelésre – következő lecke)
* Fordítást segítő eszközök (Ant, Maven, Gradle)

# Memória

Hogyan tárolódik el, amit írunk, és meddig él?

* Garbage collection
* Heap
* Stack
* Statikus memória
* Globális memória más nyelvekben – megjelenése Java nyelvben (data segment)
* Referenciák – hasonlóság/különbség más nyelvekből a pointerekkel

# Vezérlési szerkezetek

Hogyan lehet ugrálni a programban? Mikor mit használjunk?

* if-else
* kiértékelési pontok
* goto
* switch-case
* return
* ciklusok: for, while, foreach, do-while
  + break, continue
* Kivételek: mire jó, mit csinál, hogyan használjuk?

# Kollekciók

Hogyan érdemes adatokat eltárolni? Mikor mit érdemes használni?

* Ismétlés: tömbök
* Vektor
* Lista: ArrayList, LinkedList
* ((bináris) fa)
* Map
* Gyakran használt szerkezetek: FIFO, Ringbuffer, Stack

# Általános osztályok és függvények

Mi az, ami mindig hasznos lehet, és mit ad a Java ehhez?

* Object ősosztály
* Hashcode, equals, toString
* Matekos osztályok
* Comparable (sort)
* IO osztályok alapszinten (streamről meséljek)
* Erőforrás kezelési szabályok (fájl, network,…)

# Kommentezés

Mit, hogyan és miért kommentezzünk?

* Emlékeztető kommentek
* Rövid magyarázatos kommentek
* Függvényt leíró kommentek: javadoc
* Fájl szerkezetét adó vizuális komment
* History: legyen/ne legyen?
* Komment mennyisége, kinézete

# Hibakeresési módszerek, Tesztelés

Mit tegyünk, ha beütött a kraft?

* Manuális: println
* Debug line to line
* Debug: watch variables
* Tesztelési módszerek, unit teszt
* statikus kódanalízis (SonarQube)

# Rekurzió

Rekurzió, rekurzió

* Működése
* Néhány hasznos use-case: például fibonacci
* Hibái: stack overflow, integer overflow
* Sebesség: cache-elés
* Előző lecke (debug) gyakorlása rekurziós példán

# Szoftverfejlesztési modellek

Hogyan csesszük fel a programozók agyát managerként

* Vízesés
* V-modell
* Agilitás: Scrum
* Kanban

# Többszálú program

Az egész processzort kifizettem, az egészet használni is akarom

* Szekvenciális program
* Szálak létrehozása, életciklusa
* Közös erőforrások: kölcsönös kizárás
* Deadlock elkerülése
* Synchronized

# Hálózati ismeretek

* OSI 7
* Ethernet réteg
* Wi-fi
* IP (v4, v6)
* TCP vs. UDP
* HTTP, HTTPS, FTP,
* DNS kezelés,
* DHCP
* SSH
* REST API
* Websocket
* Kliens oldal, szerver oldal…
* Levelezés
* Titkosítás, aláírás: publikus és privát kulcsok

# Advanced fájlkezelés

Milyen fájlok vannak, és mit használjunk?

* Szerializálás
* Sima szövegfájl
* XML
* SQL
* Tömörített fájlok
* Képek
* …

# Állapotgép

Ha a programunk tudja, mit csinál éppen

* Felépítése, működése
* UML diagram
* Gyakorlati példa

**OOP**

# OOP alapok

Mi az, amit egy óvodás is tud?

* Osztályok láthatósága
* Konstruktor (destruktor)
* Láthatósági szokások/elvárások/alapevek
* Függvény overload

# OOP alapok folytatás

Ami szinten minden programban előfordul

* Interfészek
* Öröklés: Javában, többszörös öröklés más nyelvben
* Absztrakt osztályok

# Heterogén kollekciók

Miért jó, ha a kutya állat?

* Heterogén kollekció alapok
* Gyakorlati példák
* Template osztályok (ez nem heterogén kollekció része, de ide passzol leginkább időben)

# UML

Nem szeretjük, de minden programozás alapja

* Szöveges specifikációk
* UML class diagram
* UML sequence diagram
* UML állapotdiagram ismétlés

# Verziókezelés

Projektmunka sose vesszen el, osszuk meg egymással

* Don’t do it: mappákba rendezés dátummal
* SVN
* Git
* Git gyakorlati használat, hasznos dolgok

# Tervezés

Milyen tervezési minták és architektúrák vannak gyakran használva? (Példákkal, gyakorlatban)

* MVC
* MVVM
* Singleton, és ami még jól jöhet

# OOP elvek

Hogyan írjunk normális OO programot?

* OO elvek
* Clean code elvek
* Refaktorálás
* API tervezési elvek

# Közös projekt

* Pair programming
* Verziókezelés gyakorlás
* Stabil környezet létrehozása
* Programozás specifikáció alapján
* Dokumentálás