A tananyag elkészítését az EFOP-3.4.3-16-2016-00021 számú projekt támogatta. A projekt az Európai Unió támogatásával, az Európai Szociális Alap társfinanszírozásával valósult meg.

Magasszintű programozási nyelvek 2 DEIK PTI BSc 4 órás laborok

Feladatgyűjtemény a mérési jegyzőkönyv kidolgozásához

A Prog2 tematika tartalmi megtöltése

A jelen dokumentumban konkrét labor és otthoni feladatokat rendelünk a (számunkra hivatalból előírt, betartandó) heti bontású tematika tételeihez. A heti labormunka az adott héthez rendelt feladatok teljesítésén alapszik. A hallgatók a laboron önállóan dolgoznak, az oktató rövid indító iránymutatása (például a feladatok pontosítása) után, ám a feladatokon otthon is dolgozni kell.

- Egy „feladatcsokor” (egy csokorból 5 feladatot kell választani, lásd például később a „Helló, Arroway!” egy csokor) feladatok teljesítésére időben az adott hét áll rendelkezésre.
    - Az induláskor alkalmazunk egy 2 hetes időbeli puffert, a szeptember 21-i heti laborodon a „Helló, Berners-Lee!” csokor megkezdését kell bemutatnod, a következő héttől pedig folytatólagosan a „Helló, Arroway!”-től kezdve. Ha adott laborközösségnek laborja elmarad, akkor ez a „hol tartunk számlálót” sem léptetjük.
    - Az a feladat fogadható el megkezdettnek, amelyről részletesen tudósít a hallgató a jegyzőkönyv pdf-jében (vagy magáról a feladatról, ha kész, vagy a nehézségről, amibe beleütközött és gátolja a megoldásban).
    - A jegyzőkönyvet DocBook-ben kell megvalósítani.

Közös munka

Minden labormérést lehet egy a félév során fix, két fős csoportban is végezni. Ennek előkészülete, hogy a két hallgató a prog1-es jegyzőkönyvéből minden prog1-es feladatból kiválasztja a jobb megoldást, ezt teszi be az új mérési jegyzőkönyvbe és a jelen prog2 részt ebben az anyagban valósítja meg. Az új feladatoknál szerepeltetni kell egy bekezdést, ami rögzíti, ki mit csinált a feladatban. Védeni egyénileg kell.

Értékelési szempontok

A jegyzőkönyvnek nyilvános repóban (és DocBook XML 5 forrásokban is) elérhetőnek kell lennie. Ha a DocBook források nem validak, vagy plágium van bennük, akkor az értékelés automatikusan elégtelen.

A gyakorlati jegyet az utolsó laborokon a laborjegyzőkönyvre és a védésre adjuk.

- A jeles szükséges (de nem elégséges) feltétele, hogy
    - minden héten mind az 5 feladatra legyen megoldásunk bemutatva a jegyzőkönyvben. Legalább egy feladat megoldása, annak elmagyarázása, bemutatása legyen kistreamve vagy legalább YB videóban kitéve.
    - Legalább 5 hallgatótárs feltünteti a jegyzőkönyvében, hogy a szóban forgó hallgató tutorja volt vagy volt legalább 50 megnézés videón.
- A jó szükséges (de nem elégséges) feltétele, hogy minden héten az 5 feladatból legyen 4 feladatra megoldásunk bemutatva a jegyzőkönyvben.
- A közepes szükséges (de nem elégséges) feltétele, hogy minden héten az 5 feladatból legyen 3 feladatra megoldásunk bemutatva a jegyzőkönyvben.
- A 9 heti bontásból 1-nél lehet három feladatnál kevesebb megoldás, ha 2 vagy több olyan hét van, ahol három feladatnál kevesebb megoldás van, az automatikusan elégtelen gyakorlati jegyet eredményez.

A tananyag elkészítését az EFOP-3.4.3-16-2016-00021 számú projekt támogatta. A projekt az Európai Unió támogatásával, az Európai Szociális Alap társfinanszírozásával valósult meg.

- Ha bármely hétnél 1 megoldás van vagy egyetlen megoldás sincs, az elégtelen gyakorlati jegyet eredményez.

A félév utolsó laborjait a védésnek szenteljük, amely a jegyzőkönyvből az oktató által kiválasztott feladat gép melletti bemutatásából áll. A jegyzőkönyv és a védés alapján adja a laborvezető a gyakorlati jegyet. Ezt időben támogatandó a tematika néhány párját összevontuk az alábbiak szerint.

Háttér

Minden feladatot megcsináltam már, ezeket eléred az UDPROG közösségben (vagy a repóban, vagy az évkönyvben vagy a fészes csoportban) vagy egyéb célrepókban.

- A zöld, piros és kék (lásd később) feladatok nem kötelezőek.
  - A „deprecated” (zöld) feladatokat is lehet választani, de azok mivel régiek, kevés a támogatottság, tipikusan nehézséget okozhat a megoldásuk… (mert például a felhasznált API-k sokat változtak és már nem gondoztam a kódokat…). De olyan is van, melyet már meghaladtál, például az első részben is feldolgozhattál…
  - A piros feladat kidolgozásával az egész csokor kiváltható, ezek tipikusan kutatási feladatok, beszállhatsz velük akár kutatási kéziratokba társszerzőként, akár TDK-zhatsz, szakdolgozhatsz belőlük.
  - A kék olyan mint a piros, de a tartalmazó és egy másik csokor is kiváltható vele.

A tananyag elkészítését az EFOP-3.4.3-16-2016-00021 számú projekt támogatta. A projekt az Európai Unió támogatásával, az Európai Szociális Alap társfinanszírozásával valósult meg.

A feladatcsokrok

Minden csokor alapértelmezésben kiegészül egy opcionálisan választható előre egyeztetett Red Flower Hell[1] (Python vagy C++ megvalósítású Mineacraft MALMÖ MI ágens) fejlesztéssel és egy nyelvi csokorral, ahol a tematika egyszerű elemeinek (pl. az első héten: Osztály, objektum, példányosítás) megfelelő rövid kis kódcsipeteket mutatunk be.

EPAM

Az idei kurzus kvintesszenciája, hogy a Java platformra koncentrálva maga az EPAM is visz egy labort. Minden csokorban az "**EPAM:** " prefix után találjátok az "EPAM-os" feladatokat. Ezek ugyanúgy választhatók feladatok bármely más kurzusbeli laborcsopornak. (Ezeknek a feladatoknak a listája és a referencia megoldásukra adott tippek, segítség, megoldás-forrás linkek az első pár hétben még bővülni, frissülni fognak.)

A referencia megoldások: https://github.com/epam-deik-cooperation/epam-deik-prog2

0. hét - „Helló, Berners-Lee!”

A szokásos olvasónapló feladat:

- C++: Benedek Zoltán, Levendovszky Tihamér Szoftverfejlesztés C++ nyelven
- Java: Nyékyné Dr. Gaizler Judit et al. Java 2 útikalauz programozóknak 5.0 I-II.
    - Ebből a két könyvből pár oldalas esszé jellegű kidolgozást kérek, Java és C++ összehasonlítás mentén, pl. kb.: kifejezés fogalom ua., Javában minden objektum referencia, mindig dinamikus a kötés, minden függvény virtuális, klónozás stb.
- Python: Forstner Bertalan, Ekler Péter, Kelényi Imre: Bevezetés a mobilprogramozásba. Gyors prototípus-fejlesztés Python és Java nyelven (35-51 oldal)
    - Itt a kijelölt oldalakból egy 1 oldalas élmény-olvasónaplóra gondoltam.

1. hét - „Helló, Arroway!”

1. hét Az objektumorientált paradigma alapfoglamai. Osztály, objektum, példányosítás.

OO szemlélet

A módosított polártranszformációs normális generátor beprogramozása Java nyelven. Mutassunk rá, hogy a mi természetes saját megoldásunk (az algoritmus egyszerre két normálist állít elő, kell egy példánytag, amely a nem visszaadottat tárolja és egy logikai tag, hogy van-e tárolt vagy futtatni kell az algot.) és az OpenJDK, Oracle JDK-ban a Sun által adott OO szervezés ua.!

Lásd még fóliák!

Ismétlés: https://arato.inf.unideb.hu/batfai.norbert/UDPROG/deprecated/Prog1_5.pdf (16-22 fólia)

Ugyanezt írjuk meg C++ nyelven is! (lásd még UDPROG repó: source/labor/polargen)

Homokózó

Írjuk át az első védési programot (LZW binfa) C++ nyelvről Java nyelvre, ugyanúgy működjön! Mutassunk rá, hogy gyakorlatilag a pointereket és referenciákat kell kiirtani és minden máris működik

[1] https://github.com/nbatfai/RedFlowerHell

A tananyag elkészítését az EFOP-3.4.3-16-2016-00021 számú projekt támogatta. A projekt az Európai Unió támogatásával, az Európai Szociális Alap társfinanszírozásával valósult meg.

(erre utal a feladat neve, hogy Java-ban minden referencia, nincs választás, hogy mondjuk egy attribútum pointer, referencia vagy tagként tartalmazott legyen).

Miután már áttettük Java nyelvre, tegyük be egy Java Servletbe és a böngészőből GET-es kéréssel (például a böngésző címsorából) kapja meg azt a mintát, amelynek kiszámolja az LZW binfáját![2]

„Gagyi"

Az ismert formális[3] „while (x <= t && x >= t && t != x);" tesztkérdéstípusra adj a szokásosnál (miszerint x, t az egyik esetben az objektum által hordozott érték, a másikban meg az objektum referenciája) „mélyebb" választ, írj Java példaprogramot mely egyszer végtelen ciklus, más x, t értékekkel meg nem! A példát építsd a JDK Integer.java forrására[4], hogy a 128-nál inkluzív objektum példányokat poolozza!

Yoda

Írjunk olyan Java programot, ami java.lang.NullPointerEx-el leáll, ha nem követjük a Yoda conditions-t! https://en.wikipedia.org/wiki/Yoda_conditions

Kódolás from scratch

Induljunk ki ebből a tudományos közleményből: http://crd-legacy.lbl.gov/~dhbailey/dhbpapers/bbp-alg.pdf és csak ezt tanulmányozva írjuk meg Java nyelven a BBP algoritmus megvalósítását!

Ha megakadsz, de csak végső esetben: https://www.tankonyvtar.hu/hu/tartalom/tkt/javat-tanitok-javat/apbs02.html#pi_jegyei (mert ha csak lemásolod, akkor pont az a fejlesztői élmény marad ki, melyet szeretném, ha átélnél).

EPAM: Java Object metódusok

Mutasd be a Java Object metódusait és mutass rá mely metódusokat érdemes egy saját osztályunkban felüldefiniálni és miért. (Lásd még Object class forráskódja)

EPAM: Eljárásorientál vs Objektumorientált

Írj egy 1 oldalas értekező esszé szöveget, amiben összehasonlítod az eljárásorientált és az objektumorientált paradigmát, igyekezve kiemelni az objektumorientált paradigma előnyeit!

EPAM: Objektum példányosítás programozási mintákkal

Hozz példát mindegyik "creational design pattern"-re és mutasd be mikor érdemes használni őket!

[2]Tavalyi prog2 első védés volt.

[3]https://www.facebook.com/groups/udprog/permalink/437825193072042/?comment_id=437862206401674&reply_comment_id=437863669734861&comment_tracking=%7B%22tn%22%3A%22R3%22%7D

[4]A JDK telepítési könyvtárában az src.zip-ben találod.

A tananyag elkészítését az EFOP-3.4.3-16-2016-00021 számú projekt támogatta. A projekt az Európai Unió támogatásával, az Európai Szociális Alap társfinanszírozásával valósult meg.

2. hét - „Helló, Liskov!”

2. hét Öröklődés, osztályhierarchia. Polimorfizmus, metódustúlterhelés. Hatáskörkezelés. A bezárási eszközrendszer, láthatósági szintek. Absztrakt osztályok és interfészek.

Liskov helyettesítés sértése

Írjunk olyan OO, leforduló Java és C++ kódcsipetet, amely megsérti a Liskov elvet! Mutassunk rá a megoldásra: jobb OO tervezés.

https://arato.inf.unideb.hu/batfai.norbert/PROG2/Prog2_1.pdf (93-99 fólia)
(számos példa szerepel az elv megsértésére az UDPROG repóban, lásd pl. source/binom/Batfai-Barki/madarak/)

Szülő-gyerek

Írjunk Szülő-gyerek Java és C++ osztálydefiníciót, amelyben demonstrálni tudjuk, hogy az ősön keresztül csak az ős üzenetei küldhetőek!

Lásd fóliák! https://arato.inf.unideb.hu/batfai.norbert/PROG2/Prog2_1.pdf (98. fólia)

Anti OO

A BBP algoritmussal[5] a Pi hexadecimális kifejtésének a 0. pozíciótól számított 10^6, 10^7, 10^8 darab jegyét határozzuk meg C, C++, Java és C# nyelveken és vessük össze a futási időket!
https://www.tankonyvtar.hu/hu/tartalom/tkt/javat-tanitok-javat/apas03.html#id561066

deprecated - Hello, Android!

Élesszük fel a https://github.com/nbatfai/SamuEntropy/tree/master/cs projektjeit és vessünk össze néhány egymásra következőt, hogy hogyan változtak a források!

Hello, Android!

Élesszük fel az SMNIST for Humans projektet!
https://gitlab.com/nbatfai/smnist/tree/master/forHumans/SMNISTforHumansExp3/app/src/main
Apró módosításokat eszközölj benne, pl. színvilág.

Hello, SMNIST for Humans!

Fejleszd tovább az SMNIST for Humans projektet SMNIST for Anyone emberre szánt appá! Lásd az smnist2_kutatasi_jegyzokonyv.pdf-ben a részletesebb hátteret!

Ciklomatikus komplexitás

Számoljuk ki valamelyik programunk függvényeinek ciklomatikus komplexitását! Lásd a fogalom tekintetében a https://arato.inf.unideb.hu/batfai.norbert/UDPROG/deprecated/Prog2_2.pdf (77-79 fóliát)!

[5] https://www.tankonyvtar.hu/hu/tartalom/tkt/javat-tanitok-javat/apbs02.html#pi_jegyei

A tananyag elkészítését az EFOP-3.4.3-16-2016-00021 számú projekt támogatta. A projekt az Európai Unió támogatásával, az Európai Szociális Alap társfinanszírozásával valósult meg.

EPAM: Interfész evolúció Java-ban

Mutasd be milyen változások történtek Java 7 és Java 8 között az interfészekben. Miért volt erre szükség, milyen problémát vezetett ez be?

EPAM: Liskov féle helyettesíthetőség elve, öröklődés

Adott az alábbi osztály hierarchia.
`class Vehicle, class Car extends Vehicle, class Supercar extends Car`
Mindegyik osztály konstruktorában történik egy kiíratás, valamint a `Vehicle` osztályban szereplő `start()` metódus mindegyik alosztályban felül van definiálva.
Mi történik ezen kódok futtatása esetén, és miért?

```
Vehicle firstVehicle = new Supercar();
firstVehicle.start();
System.out.println(firstVehicle instanceof Car);
Car secondVehicle = (Car) firstVehicle;
secondVehicle.start();
System.out.println(secondVehicle instanceof Supercar);
Supercar thirdVehicle = new Vehicle();
thirdVehicle.start();
```

EPAM: Interfész, Osztály, Absztrak Osztály

Mi a különbség Java-ban a Class, Abstract Class és az Interface között? Egy tetszőleges példával / példa kódon keresztül mutasd be őket és hogy mikor melyik koncepciót célszerű használni.

A tananyag elkészítését az EFOP-3.4.3-16-2016-00021 számú projekt támogatta. A projekt az Európai Unió támogatásával, az Európai Szociális Alap társfinanszírozásával valósult meg.

3. hét - „Helló, Mandelbrot!”

3. hét Modellező eszközök és nyelvek. AZ UML és az UML osztálydiagramja.

Reverse engineering UML osztálydiagram

UML osztálydiagram rajzolása az első védési C++ programhoz. Az osztálydiagramot a forrásokból generáljuk (pl. Argo UML, Umbrello, Eclipse UML) Mutassunk rá a kompozíció és aggregáció kapcsolatára a forráskódban és a diagramon, lásd még: https://youtu.be/Td_nlERlEOs.

Lásd fóliák!

Forward engineering UML osztálydiagram

UML-ben tervezzünk osztályokat és generáljunk belőle forrást!

Egy esettan

A BME-s C++ tankönyv 14. fejezetét (427-444 elmélet, 445-469 az esettan) dolgozzuk fel!

BPMN

Rajzoljunk le egy tevékenységet BPMN-ben!
https://arato.inf.unideb.hu/batfai.norbert/PROG2/Prog2_7.pdf (34-47 fólia)

BPEL Helló, Világ! - egy visszhang folyamat

Egy visszhang folyamat megvalósítása az alábbi teljes „videó tutoriál” alapján:
https://youtu.be/0OnlYWX2v_I

TeX UML

Valamilyen TeX-es csomag felhasználásával készíts szép diagramokat az OOCWC projektről (pl. use case és class diagramokat).

EPAM: Neptun tantárgyfelvétel modellezése UML-ben

Modellezd le a Neptun rendszer tárgyfelvételéhez szükséges objektumokat UML diagramm segítségével.

EPAM: Neptun tantárgyfelvétel UML diagram implementálása

Implementáld le az előző feladatban létrehozott diagrammot egy tetszőleges nyelven.

EPAM: OO modellezés

Írj egy 1 oldalas esszét arról, hogy OO modellezés során milyen elveket tudsz követni (pl.: SOLID, KISS, DRY, YAGNI).

A tananyag elkészítését az EFOP-3.4.3-16-2016-00021 számú projekt támogatta. A projekt az Európai Unió támogatásával, az Európai Szociális Alap társfinanszírozásával valósult meg.

4. hét - „Helló, Chomsky!”

4. hét Objektumorientált programozási nyelvek programnyelvi elemei: karakterkészlet, lexikális egységek, kifejezések, utasítások.

Encoding

Fordítsuk le és futtassuk a Javat tanítok könyv MandelbrotHalmazNagyító.java forrását úgy, hogy a fájl nevekben és a forrásokban is meghagyjuk az ékezetes betűket!
https://www.tankonyvtar.hu/hu/tartalom/tkt/javat-tanitok-javat/adatok.html

OOCWC lexer

Izzítsuk be az OOCWC-t és vázoljuk a https://github.com/nbatfai/robocar-emulator/blob/master/justine/rcemu/src/carlexer.ll lexert és kapcsolását a programunk OO struktúrájába!

l334d1c4[6]

Írj olyan OO Java vagy C++ osztályt, amely leet cipherként működik, azaz megvalósítja ezt a betű helyettesítést: https://simple.wikipedia.org/wiki/Leet (Ha ez első részben nem tetted meg, akkor írasd ki és magyarázd meg a használt struktúratömb memóriafoglalását!)

Full screen

Készítsünk egy teljes képernyős Java programot!
Tipp: https://www.tankonyvtar.hu/en/tartalom/tkt/javat-tanitok-javat/ch03.html#labirintus_jatek

Paszigráfia Rapszódia OpenGL full screen vizualizáció

Lásd vis_prel_para.pdf! Apró módosításokat eszközölj benne, pl. színvilág, textúrázás, a szintek jobb elkülönítése, kézreállóbb irányítás.

Paszigráfia Rapszódia LuaLaTeX vizualizáció

Lásd vis_prel_para.pdf! Apró módosításokat eszközölj benne, pl. színvilág, még erősebb 3D-s hatás.

Perceptron osztály

Dolgozzuk be egy külön projektbe a projekt Perceptron osztályát!
Lásd https://youtu.be/XpBnR31BRJY

EPAM: Order of everything

Collection-ok rendezése esetén jellemzően futási időben derül ki, ha olyan típusú objektumokat próbálunk rendezni, amelyeken az összehasonlítás nem értelmezett (azaz T típus esetén nem implementálják a `Comparable<T>` interface-t). Pl. ClassCastException a Collections.sort() esetében, vagy ClassCastException a Stream.sorted() esetében.

[6]Lásd a C+lex megoldásom itt:
https://www.facebook.com/groups/udprog/permalink/942314465956443/?comment_id=942571282597428&comment_tracking=%7B%22tn%22%3A%22R3%22%7D

A tananyag elkészítését az EFOP-3.4.3-16-2016-00021 számú projekt támogatta. A projekt az Európai Unió támogatásával, az Európai Szociális Alap társfinanszírozásával valósult meg.

Írj olyan metódust, amely tetszőleges Collection esetén vissza adja az elemeket egy List-ben növekvően rendezve, amennyiben az elemek összehasonlíthatóak velük azonos típusú objektumokkal. Ha ez a feltétel nem teljesül, az eredményezzen syntax error-t. Például:

```
List<Integer> actualOutput = createOrderedList(input);
```

Ahol az `input Collection<Integer>` típusú. Természetesen más típusokkal is működnie kell,
feltéve, hogy implementálják a Comparable interface-t.

EPAM: Bináris keresés és Buborék rendezés implementálása

Implementálj egy Java osztályt, amely képes egy előre definiált *n* darab Integer tárolására. Ennek az osztálynak az alábbi funkcionalitásokkal kell rendelkeznie:

- Elem hozzáadása a tárolt elemekhez
- Egy tetszőleges Integer értékről tudja eldönteni, hogy már tároljuk-e (ehhez egy bináris keresőt implementálj)
- A tárolt elemeket az osztályunk be tudja rendezni és a rendezett (pl növekvő sorrend) struktúrával vissza tud térni (ehhez egy buborék rendezőt implementálj)

EPAM: Saját HashMap implementáció

Írj egy saját java.util.Map implementációt, mely nem használja a Java Collection API-t.
Az implementáció meg kell feleljen az összes megadott unit tesztnek, nem kell tudjon kezelni null értékű kulcsokat és a “keySet”, “values”, “entrySet” metódusok nem kell támogassák az elem törlést.
Plusz feladatok:

1. az implementáció támogat null kulcsokat,

a “keySet”, “values”, “entrySet” metódusok támogatják az elem törlést.

A tananyag elkészítését az EFOP-3.4.3-16-2016-00021 számú projekt támogatta. A projekt az Európai Unió támogatásával, az Európai Szociális Alap társfinanszírozásával valósult meg.

5. hét - „Helló, Stroustrup!”

5. hét Objektumorientált programozási nyelvek típusrendszere (pl.: Java, C#) és 6. hét Típusok tagjai: mezők, (nevesített) konstansok, tulajdonságok, metódusok, események, operátorok, indexelők, konstruktorok, destruktorok, beágyazott típusok.

Összevonva.

JDK osztályok

Írjunk olyan Boost C++ programot (indulj ki például a fénykardból) amely kilistázza a JDK összes osztályát (miután kicsomagoltuk az src.zip állományt, arra ráengedve)!

Másoló-mozgató szemantika

Kódcsipeteken (copy és move ctor és assign) keresztül vesd össze a C++11 másoló és a mozgató szemantikáját, a mozgató konstruktort alapozd a mozgató értékadásra!

Hibásan implementált RSA törése

Készítsünk betű gyakoriság alapú törést egy hibásan implementált RSA kódoló: https://arato.inf.unideb.hu/batfai.norbert/UDPROG/deprecated/Prog2_3.pdf (71-73 fólia) által készített titkos szövegen.

Változó argumentumszámú ctor

Készítsünk olyan példát, amely egy képet tesz az alábbi projekt Perceptron osztályának bemenetére és a Perceptron ne egy értéket, hanem egy ugyanakkora méretű „képet” adjon vissza. (Lásd még a 4 hét/Perceptron osztály feladatot is.)

Összefoglaló

Az előző 4 feladat egyikéről írj egy 1 oldalas bemutató „”esszé szöveget!

EPAM: It's gone. Or is it?

Adott a következő osztály:

```
public class BugousStuffProducer {
    private final Writer writer;
    public BugousStuffProducer(String outputFileName) throws
IOException {
        writer = new FileWriter(outputFileName);
    }
    public void writeStuff() throws IOException {
        writer.write("Stuff");
    }
    @Override
    public void finalize() throws IOException {
        writer.close();
    }
}
```

A tananyag elkészítését az EFOP-3.4.3-16-2016-00021 számú projekt támogatta. A projekt az Európai Unió támogatásával, az Európai Szociális Alap társfinanszírozásával valósult meg.

Mutass példát arra az esetre, amikor előfordulhat, hogy bár a program futása során meghívtuk a `writeStuff()` metódust, a fájl, amibe írtunk még is üres.

Magyarázd meg, miért. Mutass alternatívát.

EPAM: Kind of equal

Adott az alábbi kódrészlet.

```
// Given
String first = "...";
String second = "...";
String third = "...";
// When
var firstMatchesSecondWithEquals = first.equals(second);
var firstMatchesSecondWithEqualToOperator = first == second;
var firstMatchesThirdWithEquals = first.equals(third);
var firstMatchesThirdWithEqualToOperator = first == third;
```

Változtasd meg a `String third = "...";` sort úgy, hogy a `firstMatchesSecondWithEquals`, `firstMatchesSecondWithEqualToOperator`, `firstMatchesThirdWithEquals` értéke `true`, a `firstMatchesThirdWithEqualToOperator` értéke pedig `false` legyen. Magyarázd meg, mi történik a háttérben.

EPAM: Java GC

Mutasd be nagy vonalakban hogyan működik Java-ban a GC (Garbage Collector). Lehetséges az OutOfMemoryError kezelése, ha igen milyen esetekben?
Források:

- https://medium.com/@hasithalgamge/seven-types-of-java-garbage-collectors-6297a1418e82
- https://stackoverflow.com/questions/2679330/catching-java-lang-outofmemoryerror

A tananyag elkészítését az EFOP-3.4.3-16-2016-00021 számú projekt támogatta. A projekt az Európai Unió támogatásával, az Európai Szociális Alap társfinanszírozásával valósult meg.

6. hét - „Helló, Gödel!”

7. hét Interfészek. Kollekciók. és 8. hét Funkcionális nyelvi elemek. Lambda kifejezések.

Összevonva.

Gengszterek

Gengszterek rendezése lambdával a Robotautó Világbajnokságban
https://youtu.be/DL6iQwPx1Yw (8:05-től)

C++11 Custom Allocator

https://prezi.com/jvvbytkwgsxj/high-level-programming-languages-2-c11-allocators/ a CustomAlloc-os példa, lásd C forrást az UDPROG repóban!

STL map érték szerinti rendezése

Például: https://github.com/nbatfai/future/blob/master/cs/F9F2/fenykard.cpp#L180

Alternatív Tabella rendezése

Mutassuk be a https://progpater.blog.hu/2011/03/11/alternativ_tabella a programban a java.lang Interface Comparable<T> szerepét!

Prolog családfa

Ágyazd be a Prolog családfa programot C++ vagy Java programba! Lásd para_prog_guide.pdf!

GIMP Scheme hack

Ha az előző félévben nem dolgoztad fel a témát (például a mandalás vagy a króm szöveges dobozosat) akkor itt az alkalom!

EPAM: Mátrix szorzás Stream API-val

Implementáld le a mátrix szorzást Java-ban for és while ciklusok használata nélkül.

EPAM: LinkedList vs ArrayList

Mutass rá konkrét esetekre amikor a Java-beli LinkedList és ArrayList rosszabb performanciát eredményezhet a másikhoz képest. (Lásd még LinkedList és ArrayList forráskódja). Végezz méréseket is. (mit csinál az ArrayList amikor megtelik)

EPAM: Refactoring

Adott egy “legacy” kód mely tartalmaz anonymus interface implementációkat, ciklusokat és feltételes kifejezések. Ebben a feladatban ezt a “legacy” kódot szeretnénk átírni lambda kifejezések segítségével (metódus referencia használata előnyt jelent!)

A tananyag elkészítését az EFOP-3.4.3-16-2016-00021 számú projekt támogatta. A projekt az Európai Unió támogatásával, az Európai Szociális Alap társfinanszírozásával valósult meg.

7. hét - „Helló, !”

9. hét Adatfolyamok kezelése, streamek és 11. hét I/O, állománykezelés. Szerializáció.

Összevonva.

FUTURE tevékenység editor

Javítsunk valamit a ActivityEditor.java JavaFX programon!
https://github.com/nbatfai/future/tree/master/cs/F6
Itt láthatjuk működésben az alapot: https://www.twitch.tv/videos/222879467

OOCWC Boost ASIO hálózatkezelése

Mutassunk rá a scanf szerepére és használatára! https://github.com/nbatfai/robocar-emulator/blob/master/justine/rcemu/src/carlexer.ll

SamuCam

Mutassunk rá a webcam (pl. Androidos mobilod) kezelésére ebben a projektben:
https://github.com/nbatfai/SamuCam

BrainB

Mutassuk be a Qt slot-signal mechanizmust ebben a projektben: https://github.com/nbatfai/esport-talent-search

OSM térképre rajzolása[7]

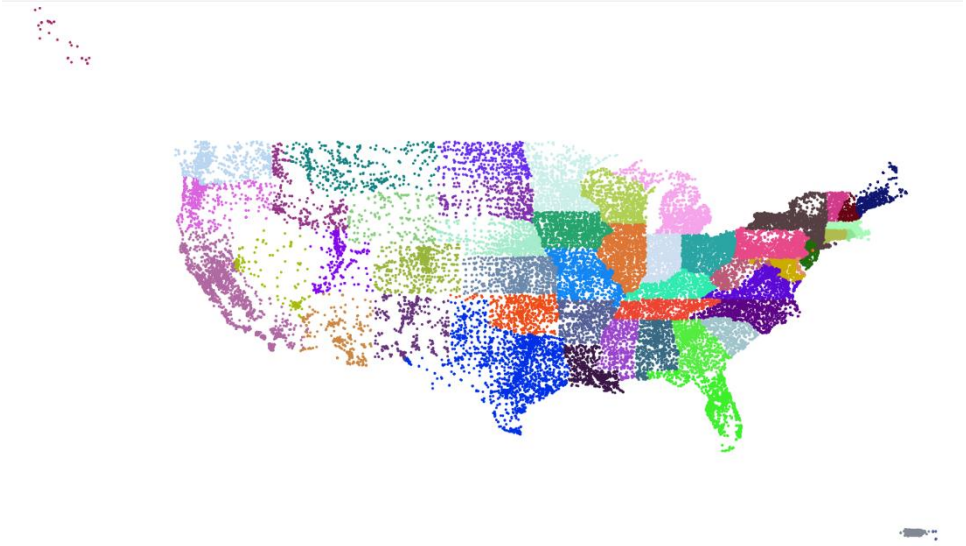
Debrecen térképre dobjunk rá cuccokat, ennek mintájára, ahol én az országba helyeztem el a DEAC hekkereket: https://www.twitch.tv/videos/182262537 (de az OOCWC Java Swinges megjelenítőjéből: https://github.com/nbatfai/robocar-emulator/tree/master/justine/rcwin is kiindulhatsz, mondjuk az komplexebb, mert ott időfejlődés is van...)

EPAM: XML feldolgozás

Adott egy koordinátákat és államokat tartalmazó XML (kb 210ezer sor), ezt az XML-t feldolgozva szeretnék létrehozni egy SVG fájlt, melyben minden város megjelenik egy pont formájában az adott koordináták alapján (tetszőleges színnel)
Plusz feladat: A városokat csoportosíthatjuk államok szerint, és minden állam külön színnel jelenjen meg a térképen, így látszódni fognak a határok is.
Elvárt eredmény:

[7] Alternatívaként készíthetsz egy GoogleMaps alapú Androidos „GPS trackert”, 2007 óta csinálok ilyen példát: https://youtu.be/QStgBZ6JfAU az aktuális a Bátfai Haxor Stream keretében: https://bhaxor.blog.hu/2018/09/19/nandigps_ismerkedes_a_gps-el

A tananyag elkészítését az EFOP-3.4.3-16-2016-00021 számú projekt támogatta. A projekt az Európai Unió támogatásával, az Európai Szociális Alap társfinanszírozásával valósult meg.



EPAM: ASCII Art

ASCII Art in Java! Implementálj egy Java parancssori programot, ami beolvas egy képet és kirajzolja azt a parancssorba és / vagy egy szöveges fájlba is ASCII karakterekkel.

EPAM: Titkos üzenet, száll a gépben!

Implementájl egy olyan parancssori alkalmazást, amely a billentyűzetről olvas soronként ASCII karakterekből álló sorokat, és a beolvasott szöveget Caesar kódolással egy txt fájlba írja soronként.

A tananyag elkészítését az EFOP-3.4.3-16-2016-00021 számú projekt támogatta. A projekt az Európai Unió támogatásával, az Európai Szociális Alap társfinanszírozásával valósult meg.

8. hét - „Helló, Lauda!”

10. hét Kivételkezelés. és 12. hét Reflexió. A fordítást és a kódgenerálást támogató nyelvi elemek (annotációk, attribútumok).

Összevonva.

Port scan

Mutassunk rá ebben a port szkennelő forrásban a kivételkezelés szerepére!

https://www.tankonyvtar.hu/hu/tartalom/tkt/javat-tanitok-javat/ch01.html#id527287

AOP

Szőj bele egy átszövő vonatkozást az első védési programod Java átiratába! (Sztenderd védési feladat volt korábban.)

Android Játék

Írjunk egy egyszerű Androidos „játékot”! Építkezzünk például a 2. hét „Helló, Android!” feladatára!

Junit teszt

A https://progpater.blog.hu/2011/03/05/labormeres_otthon_avagy_hogyan_dolgozok_fel_egy_pedat poszt kézzel számított mélységét és szórását dolgozd be egy Junit tesztbe (sztenderd védési feladat volt korábban).

OSCI

Készíts egyszerű C++/OpenGL-es megjelenítőt, amiben egy kocsit irányítasz az úton.

OSCI2

Készíts egyszerű C++/OpenGL-es megjelenítőt, amiben egy kocsit irányítasz az úton. A kocsi állapotát minden pillanatban mentsd le. Ezeket add át egy Prolog programnak, ami egyszerű reflex ágensként adjon vezérlést a kocsinak, hasonlítsd össze a kézi és a Prolog-os vezérlést. Módosítsd úgy a programodat, hogy ne csak kézzel lehessen vezérelni a kocsit, hanem a Prolog reflex ágens vezérelje!

OSCI3

Készíts egy OSM utakat megjelenítő C++/OpenGL-es progit!

EPAM: DI

Implementálj egy alap DI (Dependency Injection) keretrendszert Java-ban annotációk és reflexió használatával megvalósítva az IoC-t (Inversion Of Control).

EPAM: JSON szerializáció

Implementálj egy JSON szerializációs könyvtárat, mely képes kezelni sztringeket, számokat, listákat és beágyazott objektumokat. A megoldás meg kell feleljen az összes adott unit tesztnek.
Plusz feladat:

1. a könyvtár tudjon deszerializálni

A tananyag elkészítését az EFOP-3.4.3-16-2016-00021 számú projekt támogatta. A projekt az Európai Unió támogatásával, az Európai Szociális Alap társfinanszírozásával valósult meg.

EPAM: Kivételkezelés

Adott az alábbi kódrészlet. Mi történik, ha az input változó 1F, “string” vagy pedig null? Meghívódik e minden esetben a finally ág? Válaszod indokold!

```
public void test(Object input) {
    try {
        System.out.println("Try!");
        if (input instanceof Float) {
            throw new ChildException();
        } else if (input instanceof String) {
            throw new ParentException();
        } else {
            throw new RuntimeException();
        }
    } catch (ChildException e) {
        System.out.println("Child Exception is caught!");
        if (e instanceof ParentException) {
            throw new ParentException();
        }
    } catch (ParentException e) {
        System.out.println("Parent Exception is caught!");
        System.exit( status: 1);
    } catch (Exception e) {
        System.out.println("Exception is caught!");
    } finally {
        System.out.println("Finally!");
    }
}
```

A tananyag elkészítését az EFOP-3.4.3-16-2016-00021 számú projekt támogatta. A projekt az Európai Unió támogatásával, az Európai Szociális Alap társfinanszírozásával valósult meg.

9. hét - „Helló, Calvin!”

13. hét Multiparadigmás nyelvek és 14. hét Programozás multiparadigmás nyelveken.

Összevonva.

MNIST

Az alap feladat megoldása, +saját kézzel rajzolt képet is ismerjen fel,
https://progpater.blog.hu/2016/11/13/hello_samu_a_tensorflow-bol Háttérként ezt vetítsük le:
https://prezi.com/0u8ncvvoabcr/no-programming-programming/

Deep MNIST

Mint az előző, de a mély változattal. Segítő ábra, vesd össze a forráskóddal a
https://arato.inf.unideb.hu/batfai.norbert/NEMESPOR/DE/denbatfai2.pdf 8. fóliáját!

CIFAR-10

Az alap feladat megoldása, +saját fotót is ismerjen fel,
https://progpater.blog.hu/2016/12/10/hello_samu_a_cifar-10_tf_tutorial_peldabol

Android telefonra a TF objektum detektálója

Telepítsük fel, próbáljuk ki!

SMNIST for Machines

Készíts saját modellt, vagy használj meglévőt, lásd: https://arxiv.org/abs/1906.12213

Minecraft MALMO-s példa

A https://github.com/Microsoft/malmo felhasználásával egy ágens példa, lásd pl.:
https://youtu.be/bAPSu3Rndi8, https://bhaxor.blog.hu/2018/11/29/eddig_csaltunk_de_innentol_mi,
https://bhaxor.blog.hu/2018/10/28/minecraft_steve_szemuvege

EPAM: Reaktív programozás

Számoljuk ki az első 10 nem negatív egész szám összegét és általát.

1. Tegyük mindezt reaktív módon.
2. A számok előállítását végző komponensre "figyelhessenek" a különböző statisztikákat számító komponensek, az egyes számítások pedig párhuzamosan, külön szálon menjenek végbe.
3. Ügyeljünk arra, hogy a számok előállítása során ne küldjünk több számot az összeget és átlagot számoló szálaknak, mint amit azok fel tudnak dolgozni, bármilyen lassú is legyen a számítás. Az egyes számítások sebessége ne befolyásolja a számok előállításának és más számításoknak a sebességét.
4. Amennyiben egy számítást végző szál nem tudja fogadni a következő számot, azt mentsük el és kínáljuk fel a szálnak amint kész új számot fogadni. Az így elmentett számok mennyisége legyen limitálva, ha túl sok számot kellene elmentenünk, töröljük azt, amelyik a legrégebben érkezett. Ne blokkoljuk a számok előállítását, ha van olyan számítást végző szál, amely nem tudja feldolgozni az előállított számot.
5. A számokat a számítást végző szálak az előállításuknak megfelelő sorrendben dolgozzák fel.
6. Igyekezzünk minimálisra csökkenteni a blokkolt szálak számát.
7. A számítást végző szálak fejeződjenek be, ha nincs több feldolgozandó szám.

A tananyag elkészítését az EFOP-3.4.3-16-2016-00021 számú projekt támogatta. A projekt az Európai Unió támogatásával, az Európai Szociális Alap társfinanszírozásával valósult meg.

8. A megoldásunk legyen nyitott a bővítésre, de zárt a módosításra.
9. Legyen lehetőség új statisztikák bevezetésére úgy, hogy a meglévő osztályokat nem módosítjuk, illetve szükség esetén tudjunk hasonlóképpen új komponenseket létrehozni a számok előállítására is.

EPAM: Back To The Future

Adott az alábbi kódrészlet:

```
public class FutureChainingExercise {

  private static ExecutorService executorService = Executors.newFixedThreadPool(2);

  public static void main(String[] args) {
    CompletableFuture<String> longTask
        = CompletableFuture.supplyAsync(() -> {
      sleep(1000);
      return "Hello";
    }, executorService);

    CompletableFuture<String> shortTask
        = CompletableFuture.supplyAsync(() -> {
      sleep(500);
      return "Hi";
    }, executorService);

    CompletableFuture<String> mediumTask
        = CompletableFuture.supplyAsync(() -> {
      sleep(750);
      return "Hey";
    }, executorService);

    CompletableFuture<String> result
       = longTask.applyToEitherAsync(shortTask, String::toUpperCase, executorService);

    result = result.thenApply(s -> s + " World");

    CompletableFuture<Void> extraLongTask
        = CompletableFuture.supplyAsync(() -> {
      sleep(1500);
      return null;
    }, executorService);

    result = result.thenCombineAsync(mediumTask, (s1, s2) -> s2 + ", " +
s1, executorService);

    System.out.println(result.getNow("Bye"));
    sleep(1500);
    System.out.println(result.getNow("Bye"));

    result.runAfterBothAsync(extraLongTask, () -
> System.out.println("After both!"), executorService);
    result.whenCompleteAsync((s, throwable) -> System.out.println("Complete: " +
s), executorService);

    executorService.shutdown();
  }

  /**
   *
   * @param sleeptime sleep time in milliseconds
   */
  private static void sleep(int sleeptime) {...}

}
```

Mi lesz kiíratva a standard kimenetre és miért?

EPAM: AOP

A tananyag elkészítését az EFOP-3.4.3-16-2016-00021 számú projekt támogatta. A projekt az Európai Unió támogatásával, az Európai Szociális Alap társfinanszírozásával valósult meg.

Készíts két példa projektet, melyben egyes metódusok futási idejét méred majd kiíratod úgy, hogy a metódus futási idejének méréséhez AOP-t használsz. Az első projektben csak az AspectJ könyvtárat, a második esetében pedig Spring AOP-t használj.

Debrecen, 2020. szept. 14.
Dr. Bátfai Norbert, Lámfalusi Csaba (EPAM)