



Java alapok

Java programozás

1. gyakorlat



Fontos feladat

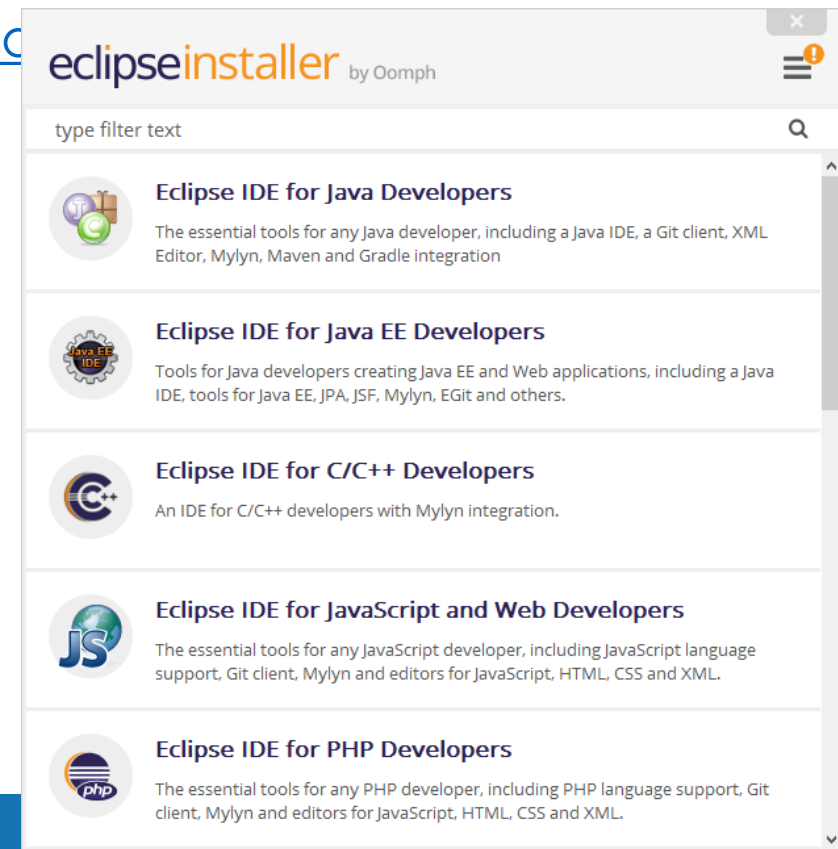
- Kérjük, hogy az alábbi linken elérhető formot mindenki egyszer töltsse ki!
 - <https://goo.gl/forms/wcpaJXwCE6DWhMc62>
 - Azért van erre szükség, hogy a félév során kötelezően házi feladatok beadásához szükséges repository elkészüljön!
- Kitöltési határidő:
 - Február 18., vasárnap 24.00
- Repositoryk elkészülnek
 - Február 19., hétfő 24.00
 - Erről lesz tájékoztatás külön!



Követelmények

- A részletes követelményrendszer a Neptunban, valamint a tárgy Wiki oldalán található meg.
- A Neptunban megadott az elsődleges.

- Amit a félév során használni fogunk:
 - Eclipse fejlesztőkörnyezet (Oxygen) 64 bit
 - Fontos, hogy 64 bites Java kell hozzá
 - Ha valakinek nem ez van otthon, akkor külön le kell töltenie a www.eclipse.org oldalról
 - <https://www.eclipse.org/download>





Tulajdonságok

- Alaptulajdonságok
 - Objektum Orientált
 - Típusos
 - Egyszeres öröklődés
 - Hordozható – amit egyszer megírunk bárhol fut
 - A VM többféle módon kezelheti a megírt JAVA kódot
 - Interpretált módszer, bájt kód sorról-sorra történő futtatásával
 - JIT (Just in Time), futtatáskor nagyobb egységből natív kódot hoz létre (tehát nem sorról-sorra interpretál)
 - Hot Spot Detection, kombinált módszer. A gyakran használt kódrészleteket fordítja natív kóddá. Gyorsabb, mivel adaptívan használ natív kódot, de nem szöszöl túl sokat a fordítással.
- Unicode
- Kivételkezelés, többszálú programozás, eseménykezelés.
- Dinamikus memóriakezelés garbage collectorral
 - Minden memória művelet ellenőrzött
 - Nincs (hibás) pointeraritmetika, alul-, vagy túlcsordulás
- Minden objektum – kivéve pár alaptípust
 - A main is egy osztályban található



Építőkövek

- Az OOP elemei:
 - osztályok (típusok)
 - adattagok
 - műveletek
- Tagolás magasabb szinten (csomagok)
 - Modularitás ...
- Párhuzamosság (végrehajtási szálak)
- Végrehajtás: programok és appletek
- Kivételek – kivételkezelés és hierarchia
- Sablonok – Generikusok



Amit használunk

- JRE – Java Runtime Environment
 - Futtatási környezet, virtuális gép
- JDK – Java Development Kit
 - Fejlesztőkörnyezet, ami tartalmaz egy JRE-t is
 - Dokumentációk, példaprogramok, fordító, stb.
- JavaDoc
- Mindezekből a SE 8 verzió
- Fejlesztőkörnyezet: Eclipse Oxygen
 - Ezen környezet használata javasolt – ZH-n azt kell használni és házit is Eclipse projektben kérjük
 - A gyakorlatok során támogatott eszköz
 - Amit a géptermekekbe letöltöttünk az 64 bites, akinek 32 bites kell, otthon töltsse le innen:
 - <https://www.eclipse.org/downloads/>
 - Eclipse IDE for Java Developers



Java virtuális gép architektúra

- A Java nyelven megírt kódot (forrás) egy fordító bájtkóddá alakítja át
- A bájtkódot csak a VM érti meg
- Az API tartalmaz előre elkészített csomagokat, amelyekben sok gyakori algoritmus és adatszerkezet implementálva van

Java Runtime Environment

- Java API
- Java VM

Operating System

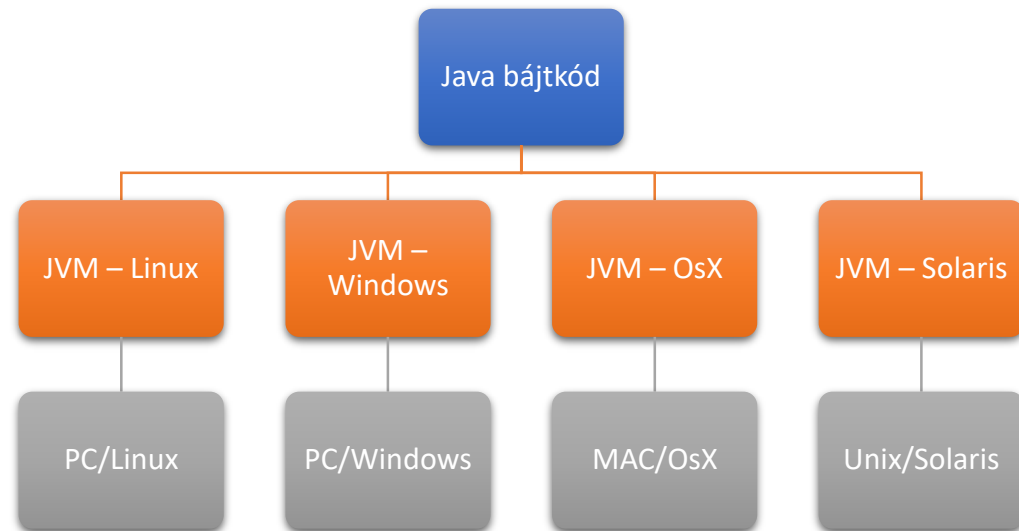
- Windows
- Linux
- Unix
- Os/X

Hardware

- Intel
- IBM
- Alpha
- ARM

Java virtuális gép architektúra

- A Java nyelven megírt kódot (forrás) egy fordító bájtkóddá alakítja át
- A bájtkódot csak a VM érti meg
- Az API tartalmaz előre elkészített csomagokat, amelyekben sok gyakori algoritmus és adatszerkezet implementálva van

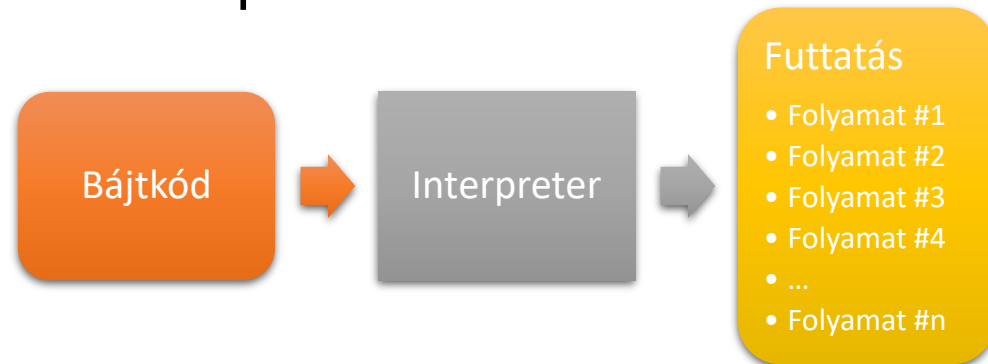


- Java forráskódból ...
 - *.java fájl
- ... a Java fordító készít ...
 - javac (Java compiler)
- ... bájtkódot, amit ...
 - *.class fájl
- ... a JVM futtat, vagy ...
 - java (konzollal)
 - javaw (konzol nélkül)
- ... vagy elemmez
 - javap

- Fordítás folyamata

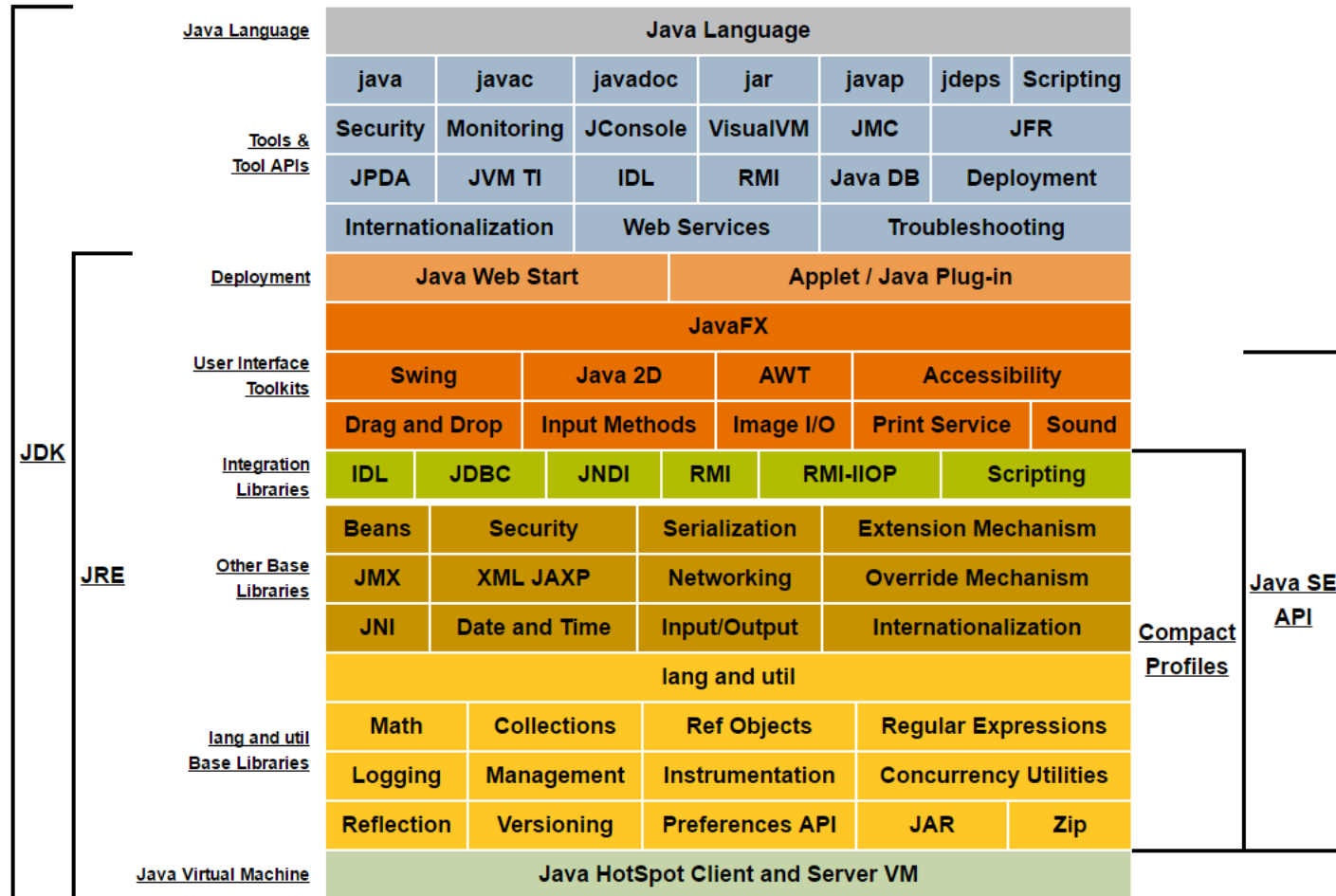


- Interpreter





Java



- A JVM nem egy konkrét implementáció, hanem szabvány
 - Oracle megoldása létezik
 - Létezik szabad szoftver megvalósítása (OpenJDK, Kaffe)
 - A 7-es Java-tól az OpenJDK az Oracle hivatalos, referencia implementációja
 - ART JVM az Androidnak
 - Bár ez kissé módosított
- Hasonlóan a .NET-hez nemcsak Java-ból lehet JVM bájtkódot generálni
 - Python – Jython
 - Scala
 - Groovy



Típusok

- Két fő fajta
 - Primitív
 - Referencia
- Primitív típusok esetében, ha egyiket egyenlővé tesszük a másikkal ($a = b;$), akkor érték szerint lesznek egyenlők, de különböző változók különböző memóriaterületen.
- Ugyanez esetben referencia típusok esetén nem lesz új változó, csak hivatkozás (a) (referencia) az eredetire (b).
- Cpp vs. Java
 - Ez az első legfontosabb különbség a C++-hoz képest



Primitív típusok

- Előjeles egészek:
 - `byte`, `short`, `int`, `long` (8, 16, 32, 64 bit)
- Lebegőpontos számok:
 - `float`, `double` (32, 64 bit)
- Unicode karakter:
 - `char`
- Logikai:
 - `boolean`
- Deklaráció:
 - `int a;`
`int b = 1;`



Referencia típusok

- Az összes többi objektum
- A tömbök is speciális referencia típusok, az indexelés nullával kezdődik
 - `int[] i = new int[10];`
 - A primitíveknek létezik előre definiált referencia változata
 - Ez általában nagy kezdőbetűvel írandó, például:
 - Double - `double`
 - De van, ami kicsit más:
 - Character - `char`, Integer - `int`
- Karakterlánc - Unicode alapú
 - Fontos, hogy ennek nincs primitív párja
 - `String`
- Deklaráció:
 - `Double a = 5.0;`
`String s = new String();`



Beépített referencia típusok

- A csomagoló osztályok használata során nem kell ügyelnünk arra, hogy ez osztály
 - A Java automatikusan kicsomagolja és becsomagolja nekünk
 - auto-boxing
 - auto-unboxing
 - Azaz értékül adhatunk egy Integert egy intnek és fordítva
- A beépített csomagoló osztályok, amik primitív típusokat reprezentálnak objektumként, értékei nem megváltoztathatóak
 - immutable osztályok
- Amikor látszólag megváltoztatjuk, új objektumpéldány jön létre az új értékkel, lecserélve az eredeti példányt
 - `Double d = 5.0;`
`d = 6;`



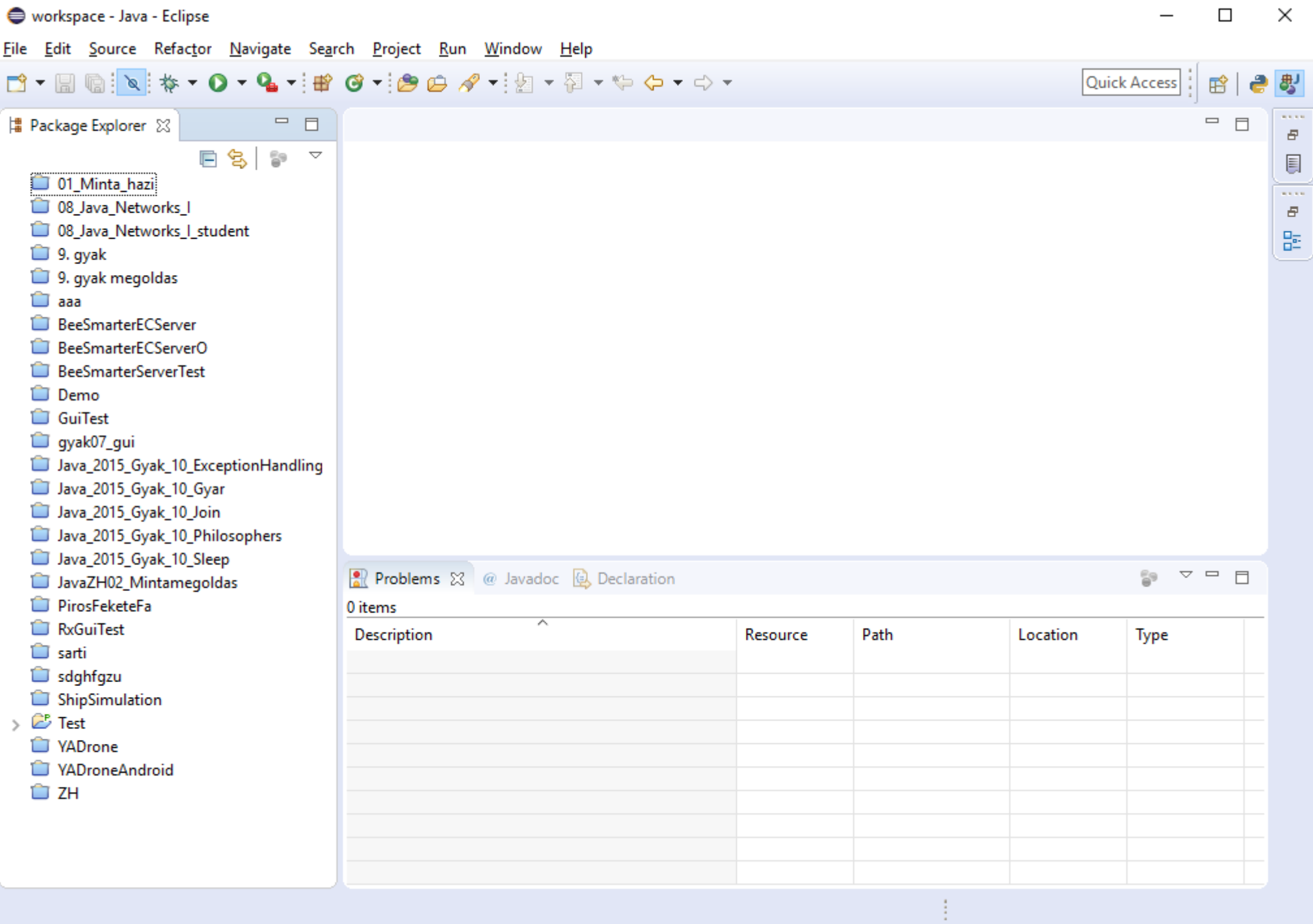
Típuskonverzió

- Kétféle konverziós módszer
 - Implicit – ekkor automatikusan megtörténik a konverzió
 - Például `int a = 5; double b = a;`
 - Explicit – ekkor saját magunk határozzuk meg a konverziót
 - Például `double a = 5.0; int b = (int) a;`
- Példa:
 - `(double) 11 / 3 = 3.666;`
 - `11 / 3 = 3;`
- A konverzió a csomagoló és a primitív között automatikus



Szia világ!

- Indítsuk el az Eclipse-t!
 - Illetve legelőször a számítógépet
- Először érdeklődni fog a workspace, azaz a munkakönyvtár felől.
 - Érdeemes lehet egy olyan könyvtárat adni meg amire teljes jogosultságotok van, illetve bárholnan elérhetitek majd.
(Például a medea-n legyen.)
 - Alapvetően a Dokumentumok könyvtárban legyen egy „workspace” mappa, és oda dolgozzatok (ez otthonról is elérhető).
- Állítsuk be a karakterkódolást!
 - Window/Preferences/General/Workspace/Text file encoding
 - UTF-8
 - Apply, OK
- Ezt majd otthon is tegyétek meg, és úgy dolgozzatok a házikon!!!





Workspace

Preferences

Workspace

- General
 - Startup and Shutdown
 - Workspaces**
 - Workspace
- Run/Debug
 - Launching

Workspace

See '[Startup and Shutdown](#)' for workspace startup and shutdown preferences.

☒ Build automatically

☐ Refresh using native hooks or polling

☒ Refresh on access

☐ Save automatically before build

☐ Always close unrelated projects without prompt

Workspace save interval (in minutes): 5

Workspace name (shown in window title): workspace

Workspace path: C:\Users\Kalman\workspace

☐ Show workspace path in window title

Open referenced projects when a project is opened

☐ Always ☐ Never ☒ Prompt

Command for launching system explorer: explorer /E,/select=\${selected_resource_loc}

Text file encoding

☐ Default (Cp1250)

☒ Other: UTF-8

New text file line delimiter

☒ Default (Windows)

☐ Other: Windows

Restore Defaults Apply

OK Cancel



Szia világ!

- A következő menüpontokra kattintsunk:
File | New ... | Project
- Azután a projekt típusát kell kiválasztani – jelen esetben ez a Java project.
- A harmadik ablakon további beállítások
 - Library-k
 - Futtató környezetek
 - Hivatkozó projektek

Select a wizard

Create a Java project



- Wizards:
- type filter text
- > General

> C/C++

> Git

> Gradle

▼ Java

Annotation

Class

Enum

Interface

Java Project

Java Project from Existing Ant Buildfile

Java Working Set

Package

Source Folder

> Java Run/Debug

> JUnit

> Launch Targets

> Maven

> Oomph

> PyDev

> Remote System Explorer

> RPM

> SVN

> Tasks

> Tracing

Create a Java Project

Create a Java project in the workspace or in an external location.



Project name: HelloWorld

☒ Use default location

Location: C:\Users\Kalman\workspace\HelloWorld

Browse...

JRE

☒ Use an execution environment JRE: JavaSE-1.8

☐ Use a project specific JRE: jre1.8.0_121

☐ Use default JRE (currently 'jre1.8.0_121') [Configure JREs...](#)

Project layout

☐ Use project folder as root for sources and class files

☒ Create separate folders for sources and class files [Configure default...](#)

Working sets

☐ Add project to working sets

New...

Working sets:

Select...

New Java Project

Java Settings

Define the Java build settings.

SourceProjectsLibrariesOrder and Export

▼ HelloWorld

> src

▼ Details

Create new source folder: use this if you want to add a new source folder to your project.

Link additional source: use this if you have a folder in the file system that should be used as additional source folder.

Add project 'HelloWorld' to build path: Add the project to the build path if the project is the root of packages and source files. Entries on the build path are visible to the compiler and used for building.

☐ Allow output folders for source folders

Default output folder:

HelloWorld/bin

Browse...

?

< Back

Next >

Finish

Cancel

New Java Project

Java Settings

Define the Java build settings.

SourceProjectsLibrariesOrder and Export

JARs and class folders on the build path:

▼ JRE System Library [JavaSE-1.8]

Access rules: No rules defined

External annotations: (None)

Native library location: (None)

> resources.jar - C:\Program Files\Java\jre1.8.0_121\rt.jar - C:\Program Files\Java\jre1.8.0_121\libjsse.jar - C:\Program Files\Java\jre1.8.0_121\libjce.jar - C:\Program Files\Java\jre1.8.0_121\libcharsets.jar - C:\Program Files\Java\jre1.8.0_121\libjfr.jar - C:\Program Files\Java\jre1.8.0_121\libaccess-bridge-64.jar - C:\Program Files\Java\jre1.8.0_121\libcldrdata.jar - C:\Program Files\Java\jre1.8.0_121\libdnsns.jar - C:\Program Files\Java\jre1.8.0_121\libjaccess.jar - C:\Program Files\Java\jre1.8.0_121\libjfxrt.jar - C:\Program Files\Java\jre1.8.0_121\liblocaledata.jar - C:\Program Files\Java\jre1.8.0_121\libnashorn.jar - C:\Program Files\Java\jre1.8.0_121\libsunec.jar - C:\Program Files\Java\jre1.8.0_121\libsunjce_provider.jar - C:\Program Files\Java\jre1.8.0_121\libsunmscapi.jar - C:\Program Files\Java\jre1.8.0_121\libsunpkcs11.jar - C:\Program Files\Java\jre1.8.0_121\libzipfs.jar - C:\Program Files\Java\jre1.8.0_121\lib\

Add JARs...

Add External JARs...

Add Variable...

Add Library...

Add Class Folder...

Add External Class Folder...

Edit...

Remove

Migrate JAR File...

?

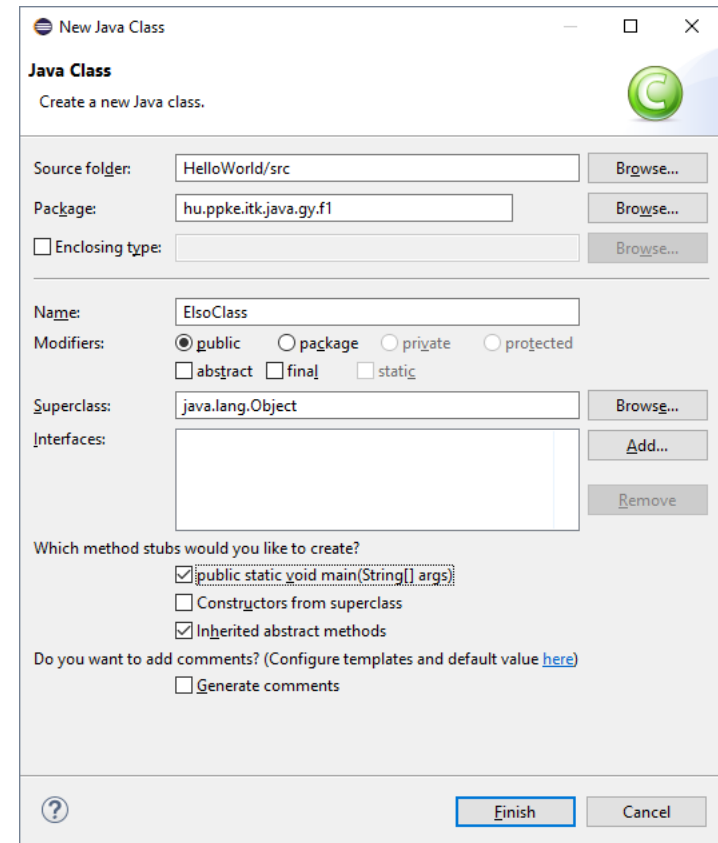
< Back

Next >

Finish

Cancel

- Ezután egy új osztályt fogunk létrehozni, a legegyszerűbb módon!
- Létrehozni a File | New | Class menüponttal
- Az osztály neve Hello legyen.
- A package legyen hu.ppke.itk.java.gy1.f1



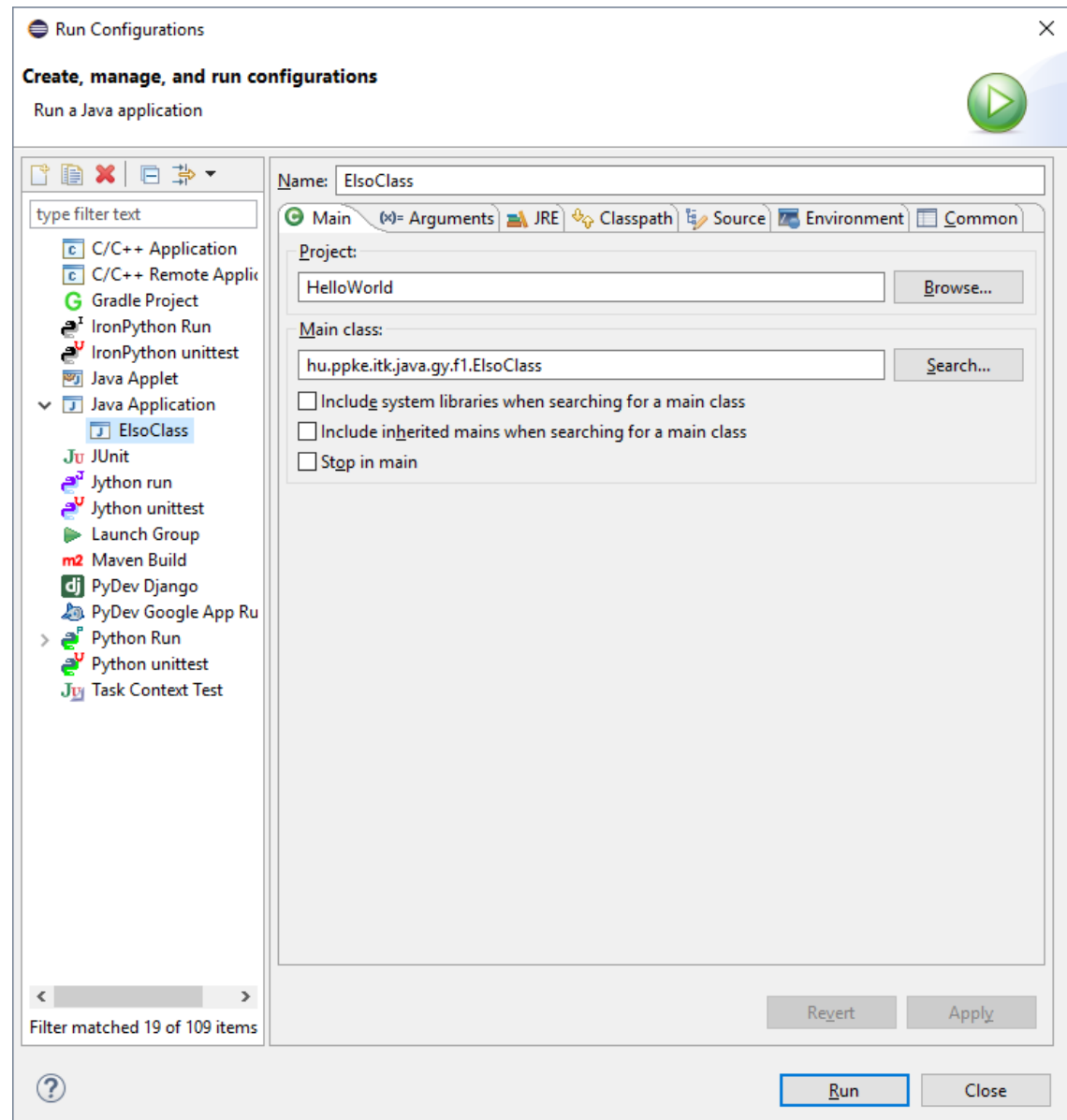
- Ezután megírjuk az első osztályunkat, benne a main függvényével.

*ElsClass.java

```
1 package hu.ppke.itk.java.gy.f1;
2
3 public class ElsClass {
4
5     public static void main(String[] args) {
6         System.out.println("Szervusz világ!");
7     }
8
9 }
10
```

Java – Futtatás

- Kattintsunk vagy a zöld nyilacskára vagy a Run | Run menüpontra.
- Ekkor konfigurálni kell a fordító és projekt típusát. Most nekünk a Java Application kell. Kattintsunk kettőt arra.
- Ha mindezzel készen vagyunk a Run gombra kattintsunk.

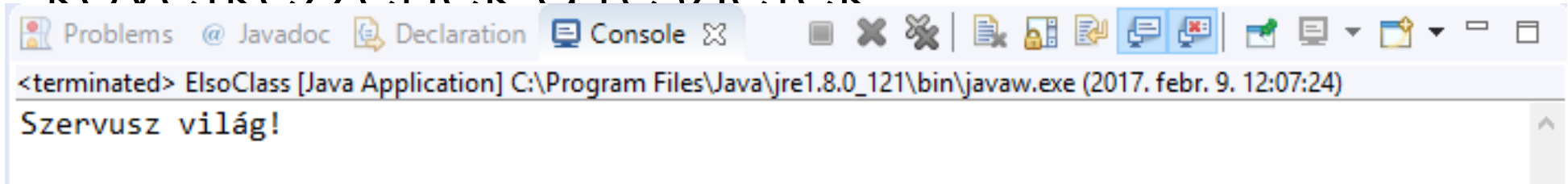


- Ha valaki előzőleg nem mentette



Java – Futtatás

- Siker esetén az alsó részen egy Console feliratú fülben megjelent a Szia világ szöveg!
- Az előbb definiáltunk egy osztályt, most következzenek a részletek





Osztály definiálása

```
public class Osztaly extends Szulo implements Interface
{
    // Mező
    private int szamlalo;

    public Osztaly()
    {
        /*
         * Ez itt az osztály konstruktor
         * Nem csinál mást csak kinullázza az egyetlen mezőt.
         */
        szamlalo = 0;
    }

    public void hozzaad(int mennyit) throws SajatException
    {
        this.szamlalo += mennyit;
    }
}
```



Kulcsszavak

- **Osztály**
 - **public** – „mindenki” elérheti
 - **final** – nem lehet tovább örökölni belőle
 - **abstract** – absztrakt osztály, absztrakt metódusa(i) van(nak)
 - **extends** – öröklésnél, a szülőosztályt adjuk meg
 - **implements** – után a megvalósított interfészeket.
- Ezek mind opcionális kulcsszavak. Ha nem adjuk meg az osztály láthatóságát alapértelmezésként csak a csomagon belül lesz elérhető
- **Mezők**
 - **public** – mindenki (ez az osztály, gyermekosztály, csomagon belül és kívül)
 - **protected** – ebben az osztályban, gyermekosztályban és a csomagon belül látható
 - alapértelmezett láthatóság – ebben az osztályban és csomagon belül
 - **private** – csak ebben az osztályban
 - **final** – nem változtatható meg az értéke - konstans
 - **volatile** – később (szálak közötti szinkronizációnál van szerepe)
 - **static** – Objektumpéldányosítás nélkül elérhető mező



Kulcsszavak

- **final** – még egyszer
 - Fontos megjegyezni, hogy itt a változó értéke ami nem megváltoztatható
 - Primitív esetén az értéke
 - Object esetén pedig az objektumra mutató referencia
 - A referencia mögött rejtőző objektum mezőire ez nincsen hatással!
- Tagfüggvények
 - **public protected private abstract static**
 - **synchronized** – később (szálak közötti szinkronizáció)
 - **void** – ha nincs visszatérési érték
 - **throws** – A függvény által dobható kivételek típusát kell utána felsorolni.



Csomagok

- Javában az osztályok csomagokba szervezhetőek. Ha nem adunk meg csomagot egy .java fájlban, akkor az „alapértelmezett” csomaghoz fog tartozni. (Nem javasolt!)
- Más csomagokhoz tartozó osztályokat csomagnévvel felsorolva érhetünk el, vagy a .java fájl elején jelezzük, hogy mely csomagokra van szükségünk.
- A csomagnév meghatározása a következőképpen történik:
 - `package csomagom;`
`package java;`
`package java.util;`
- „Betöltés” – Jelzem, hogy mely osztály kell:
 - (Analógia: a c++ linker (és nem compiler!))
`import java.util.*;`
`import java.util.LinkedList;`
- „Betöltés” nélkül:
 - `java.util.LinkedList ll =`
`new java.util.LinkedList();`



- Dokumentációs kommentek hozzáadása a forráskódhoz
 - Teljes html alapú dokumentáció generálható belőle
 - Ilyen például a Oracle (korábban Sun) hivatalos API dokumentációja is

```
/**  
 * Ez a függvény növeli a számláló értéket  
 * <p>Új bekezdésben lesz ez.  
 * @param mennyit Az értéke annak, hogy mennyit  
 * adjunk hozzá számlálóhoz  
 * @return void Nincs visszatérési érték (Ezért  
 * itt nem is használandó)  
 * @throws SajatException  
 * @author kami  
 */  
public void hozzaad(int mennyit) throws SajatException  
{  
    this.szamlalo += mennyit;  
}
```




JavaDoc – Exportálás

- Kattintsunk a File | Export menüpontra
- Eztán a Java / JavaDoc elemet kiválasztva, majd megadva a dokumentumkészítés paramétereit legenerálódik a html dokumentáció, melyet a projekt mappájában találhatunk meg.
- Ugyanitt tudunk exportálni .jar tömörítvénybe is

Konstruktorok

- Egy osztály példányosításakor a belső változókat inicializálni konstruktorral lehet
 - Javában a változók megszüntetése, memória felszabadítása automatikus, tehát destruktorok nincsenek
- Például:

```
• public class ConstructorExample
{
    private int i;
    private Double d;
    private String s;

    public ConstructorExample()
    {
        i = 0;
        d = new Double(5.1);
        // vagy!
        // d = 5.1;
        s = new String("Aladár");
    }
}
```

- Több konstruktor is lehet, különböző paraméterekkel:

- ```
public ConstructorExample()
{
 // mint az előbb
}
```

```
public ConstructorExample(String s)
{
 this.s = s;
 // ...
}
```

```
public ConstructorExample(Double d)
{
 this.d = d;
 // ...
}
```

- Hasonlóan lehet más függvényekből is több ...



# Konstruktorok

- Inicializálni a deklarációnál is lehet:

- **public class** ConstructorExample  
{  
    **private int** i = 0;  
    **private** Double d = 5.1;  
    **private** String s = "Béla";  
    // ...  
}



# Destruktorok

- Mivel a Java automatikusan kezeli memóriát nincsen destruktor a Java nyelvben
  - Van egy metódus, ami akkor fut le, ha a garbage collector törli az objektumot a memóriából, ez a `finalize()` metódus
  - Az, hogy a futása ennek mikor történik meg számos véletlen dologtól függ
    - JVM verziója és implementációja
    - GC állapota
- Nem garantált az, hogy egyáltalán lefut!



# Típusok és paraméterátadás

- Két fontos szabályt kell tudni:
  - Java esetében MINDIG érték szerinti paraméterátadás van! Azaz a változó értéke másolódik!
  - A referencia típusoknál az érték a memóriacím, azaz a cím másolódik tehát, a paraméterátadás látszólagosan referencia szerinti!
    - Mivel a beépített csomagoló típusok nem megváltoztatható értékűek, gyakorlatilag úgy látszik, mintha nem volna referencia.

```
• public class ParameterAtadas
{
 public static void param(int primitive,
 Integer reference)
 {
 System.out.println(">> " + primitive + " " +
 reference);
 primitive++;
 reference++;
 System.out.println(">> " + primitive + " " +
 reference);
 }
 public static void main(String[] arg)
 {
 int i1 = 1;
 Integer i2 = 5;
 System.out.println("1. " + i1 + " " + i2);
 param(i1, i2);
 System.out.println("2. " + i1 + " " + i2);
 }
}
```



# Típusok és paraméterátadás

```
public class Parameterek {
 private int a;

 public Parameterek (int a)
 {
 this.a = a;
 }

 void addOne() { a++; }

 int get() { return a; }

 public static void main(String[] arg)
 {
 Parameterek also = new Parameterek(1);
 Parameterek masodik = also;
 System.out.println(also.get());
 System.out.println(masodik.get());
 masodik.addOne();
 System.out.println(also.get());
 System.out.println(masodik.get());
 }
}
```



# Gondoljunk a következőkre ...

- Az `reference++` utasítás mögött a következők vannak
  - `reference = reference + 1;`
  - `reference = new Integer(reference + 1);`
  - Az első sor egyértelmű,
  - A második az immutable tulajdonság következménye
- Technikailag a tényleges megoldás közelebb van az alábbihoz
  - `reference = Integer.valueOf(reference.intValue() + 1);`
  - Ahol a statikus `valueOf()` függvény létrehoz egy új Integer példányt





# Vezérlési szerkezetek - IF

- **if** (logikai kifejezés)
  - Utasítás;
- 

- **if** (logikai kifejezés)
- Utasítás;
- **else**
- Utasítás2;



# Vezérlési szerkezetek - SWITCH

```
class SwitchDemo {
 public static void main(String[] args) {
 int month = 8;
 switch (month)
 {
 case 1: System.out.println("January"); break;
 case 2: System.out.println("February"); break;
 case 3: System.out.println("March"); break;
 case 4: System.out.println("April"); break;
 case 5: System.out.println("May"); break;
 case 6: System.out.println("June"); break;
 case 7: System.out.println("July"); break;
 case 8: System.out.println("August"); break;
 case 9: System.out.println("September"); break;
 case 10: System.out.println("October"); break;
 case 11: System.out.println("November"); break;
 case 12: System.out.println("December"); break;
 default: System.out.println("Invalid month."); break;
 }
 }
}
```



# Vezérlési szerkezetek - CIKLUSOK

- **while** (logikai kifejezés)
    - {
      - Utasítás1;
      - ...
      - Utasítás2;
    - }
- 
- **do**
    - {
      - Utasítás;
    - }
  - **while** (logikai kifejezés);



# Vezérlési szerkezetek - FOR

```
int sum(int[] a)
{
 int result = 0;
 for (int i = 0;
 i < a.length; i++)
 {
 result += a[i];
 }
 return result;
}
```

```
int sum(int[] a)
{
 int result = 0;
 for (int e : a)
 result += e;
 return result;
}
```



# Vezérlési szerkezetek – FOREACH

- Megoldhatjuk sima ciklussal azt hogy végig lépkedjünk egy tömb, vagy lista elemein de erre kínál alternatívát a Java a for-each ciklussal
- Bármilyen „bejárható” objektumra használható
  - Ami a Iterable interfészt megvalósítja, erről majd később



# Enum típus

- Az felsorolási típus egy olyan típus mely mezőit fix konstansokból állnak
- Ilyen lehet pl. a hét napjait reprezentáló típus:  

```
public enum Day {
 SUNDAY, MONDAY, TUESDAY, WEDNESDAY,
 THURSDAY, FRIDAY, SATURDAY
}
```
- Mint ahogyan az a példán is látható, a Java kulcsszó az ilyen típusokra az enum
- Javában az enum típus néhány extra funkcionálitással is rendelkezik
  - Például további függvények megadhatók, mint egy osztálynál
  - Használható **switch** és **for** utasításoknál
  - Ez a használatát teszi kényelmessé



# Enum típus

- A switch szerkezet használható int típuson kívül enum típusra is:

```
switch (day) {
 case MONDAY: System.out.println("Mondays are bad.");
 break;
 case FRIDAY: System.out.println("Fridays are better.");
 break;
 case SATURDAY:
 case SUNDAY: System.out.println("Weekends are best.");
 break;
 default: System.out.println("Midweek days are so-so.");
 break;
}
```

- Használható foreach ciklusban

```
for (Day day: Day.values()) {
 System.out.println(day);
}
```



# enum – További lehetőségek

- Egy **enum** típus létrehozása során valójában egy osztályt hozunk létre, így ezen belül definiálhatunk példány/osztályváltozókat és -metódusokat ill. konstruktort
- A fordító minden esetben néhány plusz metódust ad hozzá minden deklarált **enum** típushoz, ilyen pl. a *valueOf()* függvény
- További információk:
  - <http://download.oracle.com/javase/8/docs/api/java/lang/Enum.html>
  - <http://download.oracle.com/javase/tutorial/java/javaOO/enum.html>





# Önálló feladat

- Írjunk egy (külön) Java osztályt, ami képes statisztikai műveleteket végezni egy tömbön, amelynek értékei számok
- A műveletek:
  - Szélsőérték keresése, átlag, medián, módusz.
- Az egyes számításokat külön függvények végezzék.
- A tömböt a konstruktorban adjuk át az osztálynak
- A tömb értékeit a felhasználótól olvassuk be.



# Hasznos Eclipse gyorsbillentyűk

- Érdeemes a következő billentyűket megtanulni és használni

- Quick Fix Ctrl+1
- Open Call Hierarchy Ctrl+Alt+H
- Open Declaration F3
- Open Type Shift+Ctrl+T
- Change Method Signature Shift+Alt+C
- Convert Local Variable to Field Shift+Alt+F
- Extract Local Variable Shift+Alt+L
- Extract Method Shift+Alt+M
- Inline Shift+Alt+I
- Organize Imports Shift+Ctrl+O
- Debug Java Application Shift+Alt+D J
- Debug Last Launched F11
- Run Java Application Shift+Alt+V R

- Step Into F5
- Step Into Selection Ctrl+F5
- Step Over F6
- Step Return F7
- Toggle Line Breakpoint Shift+Ctrl+B
- Add Block Comment Shift+Ctrl+/  
Shift+Ctrl+M
- Add Import Shift+Ctrl+M
- Add Javadoc Comment Shift+Alt+J
- Format Shift+Ctrl+F
- Indent Line Ctrl+I
- Copy Lines Ctrl+Alt+Down
- Delete Next Ctrl+D
- Delete Next Word Alt+D
- Delete Previous Word Alt+Backspace
- **Code completion Ctrl+Space**



# System osztály, Runtime osztály

- Ezek adják a kapcsolatot a virtuális gép illetve a fizikai gép és a Java programunk között
  - Standard I/O eszközök elérése
  - GC kényszerített kezelése és beállítása
  - Fizikai gép információk
    - Platform, OS
  - VM információk
    - Memória (Heap)

- **System**

- `exit()`
  - Mivel a `main` függvény `void` típusú, amennyiben valamilyen visszatérési értékkel szeretnénk befejezni a programot ezt használjuk
- `arraycopy()`
  - Tömbök hatékony másolása
- `currentTimeMillis()`
  - Aktuális időpont milliszekundumban kifejezve
  - Algoritmusok futási idejének méréseihez használható
- ~~`gc()`~~
  - Garbage collector explicit futtatása
    - NE használjuk, a Java GC automatikája okosabb, mint mi ☺

- **Runtime**

- ```
public static void main(String[] args)
{
    System.out.println("Total Memory"
        + Runtime.getRuntime().totalMemory());
    System.out.println("Free Memory"
        + Runtime.getRuntime().freeMemory());
}
```

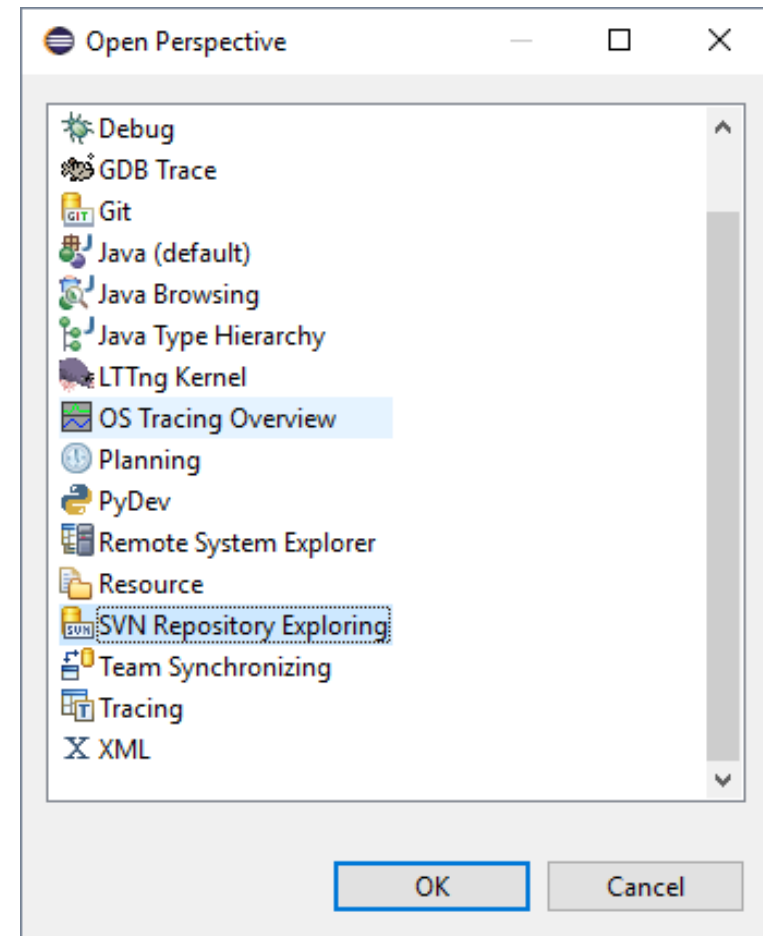
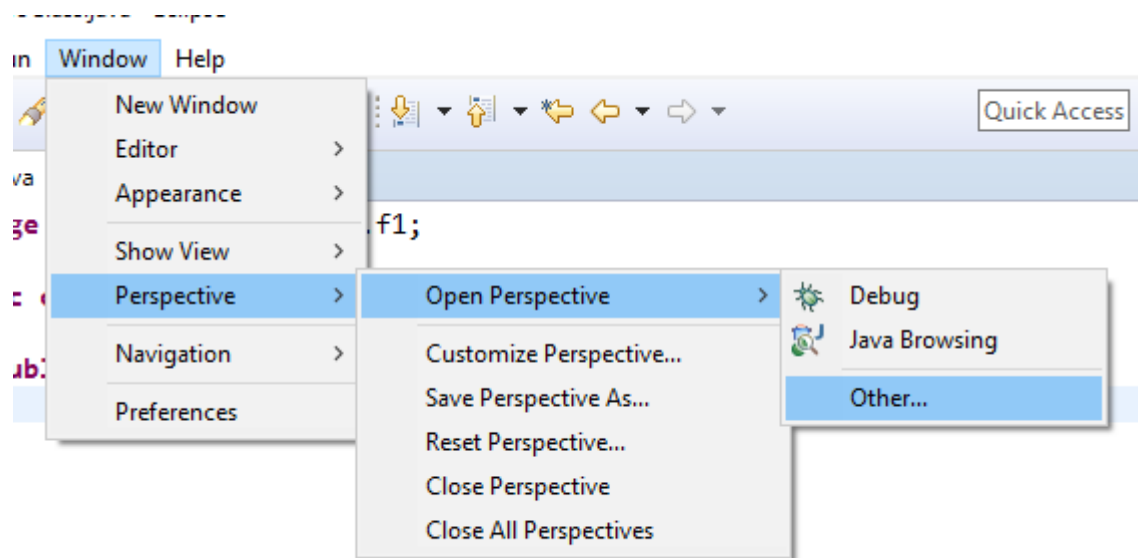


Subversive – Eclipse plugin

- Az Eclipse rendelkezik teljes SVN támogatással a subversive plugin segítségével.
- Telepítéshez link:
- <http://www.eclipse.org/subversive/documentation/gettingStarted/aboutSubversive/install.php>



SVN nézet



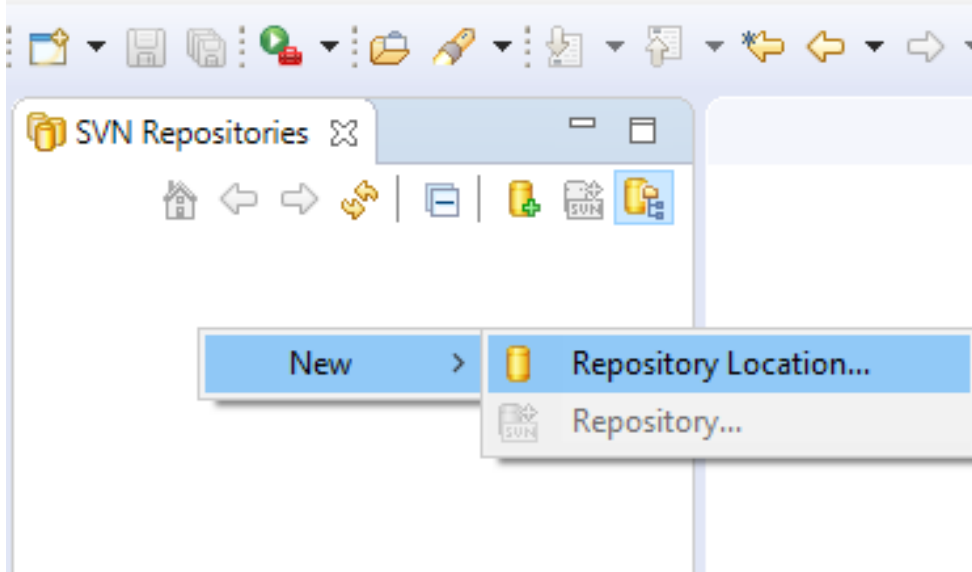


Checkout

https://wsn.itk.ppke.hu/java_repo/login_new

workspace - SVN Repository Exploring - Eclipse

File Edit Navigate Search Project Run Window Help



New Repository Location

Enter Repository Location Information

Define the SVN repository location information. You can specify additional settings for proxy and svn+ssh, https connections.

General Advanced SSH Settings SSL Settings

URL: Browse...

Label

☒ Use the repository URL as the label

☐ Use a custom label:

Authentication

User:

Password:

☐ Save authentication (could trigger secure storage login)

To manage your security data, please see ["Secure Storage"](#)

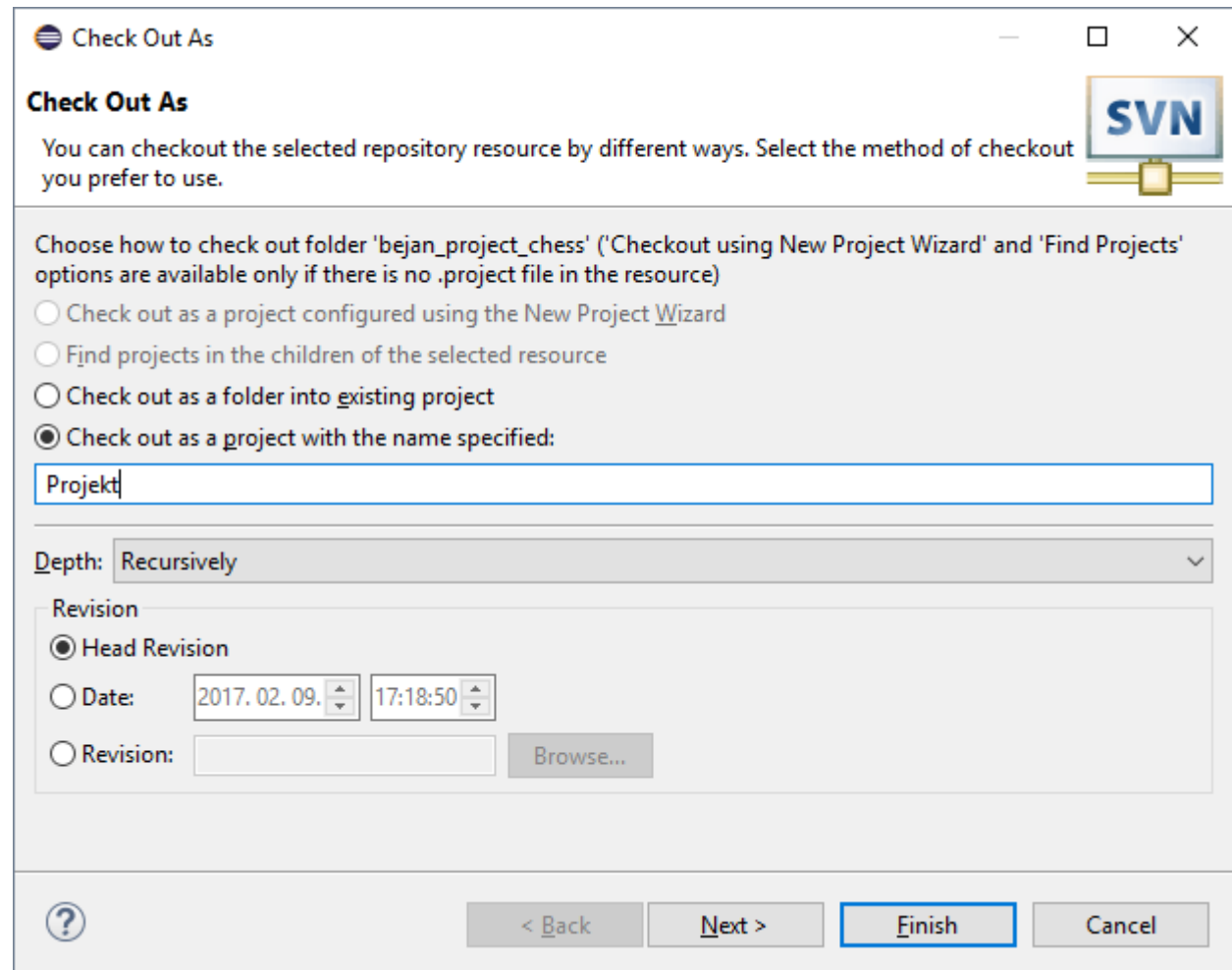
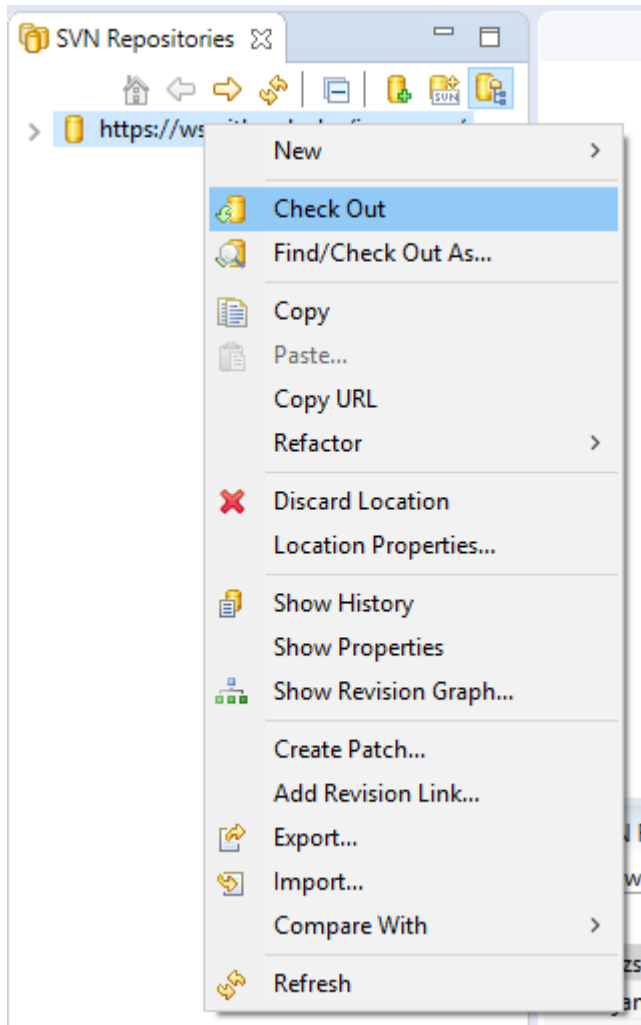
Show Credentials For:

☒ Validate Repository Location on finish









Reset Changes


Finish Cancel

Checkout



Checkout – eredmény

 Java_2015_Gyak_10_Sleep
 JavaZH02_Mintamegoldas
 PirosFeketeFa
>  Projekt 572 [https://wsn.itk.ppke.hu/java_repo/]
 RxGuiTest
 sarti
 sdghfgzu
 ShipSimulation


No



Update

Project 5/2 https://www.itk.nyu.edu/

- New
- Go Into
- Open in New Window
- Open Type Hierarchy F4
- Show In Alt+Shift+W >
- Copy Ctrl+C
- Copy Qualified Name
- Paste Ctrl+V
- Delete Delete
- Remove from Context Ctrl+Alt+Shift+Down
- Build Path >
- Source Alt+Shift+S >
- Refactor Alt+Shift+T >
- Import...
- Export...
- Refresh F5
- Close Project
- Close Unrelated Projects
- Assign Working Sets...
- Run As >
- Debug As >
- Profile As >
- Validate
- Profiling Tools >
- Restore from Local History...
- PyDev >
- Team >**
- Compare With >
- Replace With >
- Configure >
- Properties Alt+Enter

- Synchronize with Repository Ctrl+Alt+S
- Commit... Ctrl+Alt+C
- Update Ctrl+Alt+U**
- Update to Revision... Ctrl+Alt+D
- Revert...
- Add to Version Control...
- Add to svn:ignore... Ctrl+Alt+I
- Edit Conflicts
- Edit Tree Conflicts
- Mark as Merged
- Branch...
- Tag...
- Merge... Ctrl+Alt+E
- Switch...
- Add Revision Link...
- Show History
- Show Revision Graph... Ctrl+Alt+Y
- Lock... Ctrl+Alt+K
- Unlock...
- Scan Locks
- Show Properties
- Set Property...
- Set Keywords...
- Set External Definition...
- Create Patch...
- Apply Patch...
- Copy To...
- Export...
- Add to Index
- Ignore
- Cleanup
- Disconnect
- Upgrade Projects...

Megjelölés verziókezelésre

- Az SVN alapból csak azokat a fájlokat veszi figyelembe az adott könyvtárban, amik már verziókezelés alatt állnak vagy meglettek jelölve verziókezelésre.

▲ > delegateMAS 34

DelegateMASBehaviorStrategy.java 36

DelegateMASDeliveryVechicleAgent.java 36

DelegateMASDrivingStrategy.java 1

DelegateMASRoutingStrategy.java 36

DeliveryAgent.java 33

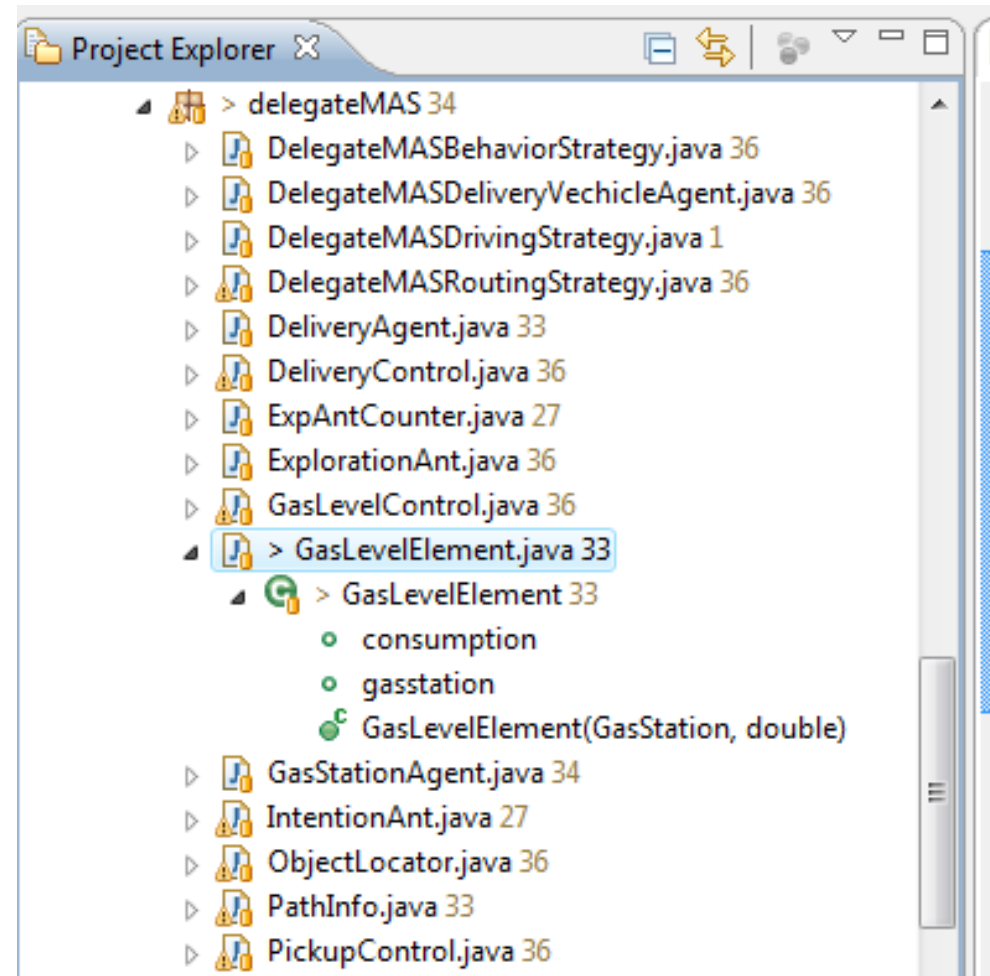
DeliveryControl.java 36

ehhe.java

ExpAntCounter.java 27

ExplorationAnt.java 36

- A fájl név előtti „>” jel jelzi, hogy változás történt a fájlban a legutóbbi update verzióhoz képest.
- A fájl név mögötti szám jelzi a revízió számot.





Commit

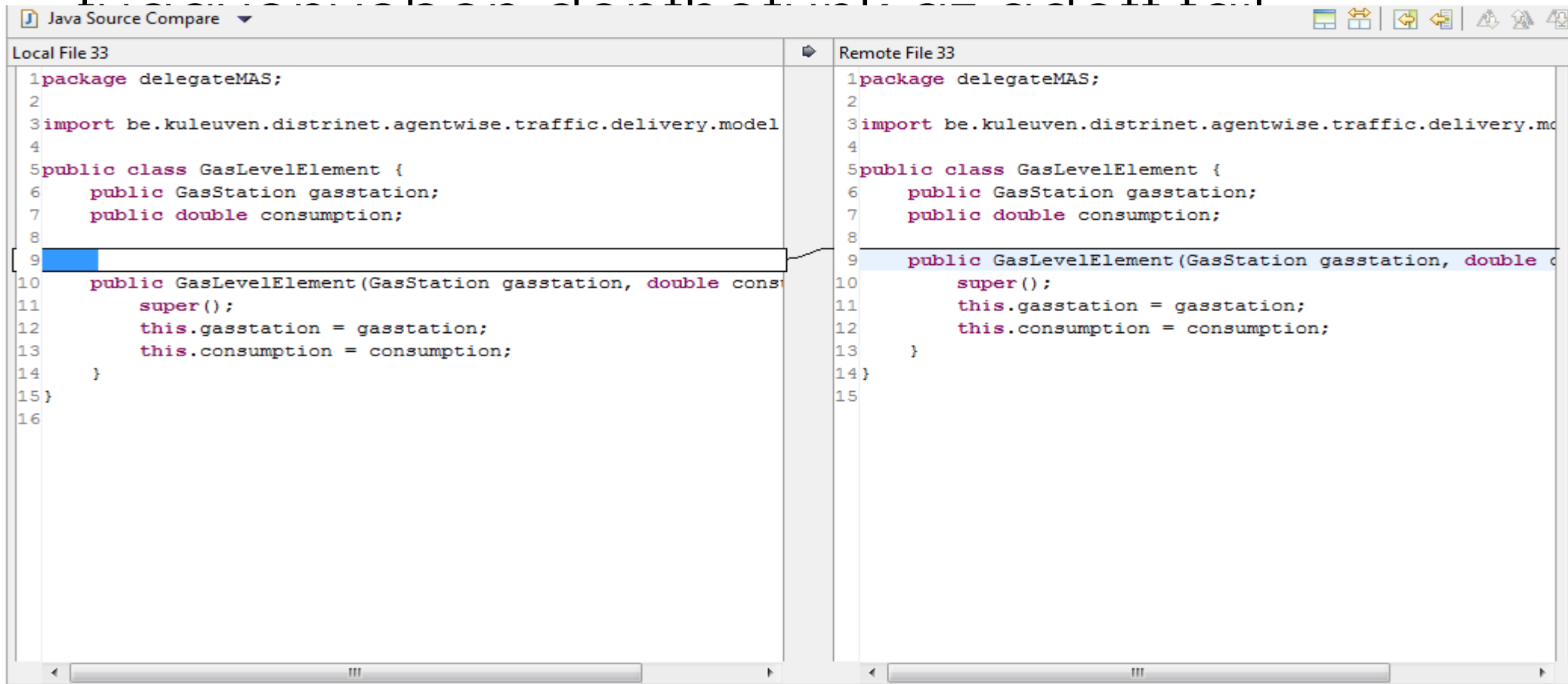
- Megjegyzés készítése nagyon fontos, a későbbi követhetőség érdekében, valamint a projekt többi résztvevőjét is tudjuk tájékoztatni ily módon.
- Egy commithoz természetesen több fájl is tartozhat. Pl. egy commit egy egész hibajavítás összefogása.

The screenshot shows a 'Commit' dialog box with the following elements:

- Title Bar:** 'Commit' with a close button (X).
- Section Header:** 'Enter a commit comment'.
- Text:** 'You can specify a new message or choose the previously entered one. Empty comments are allowed, but filling a comment message would help other people to understand the changes.'
- SVN Icon:** A small icon with the text 'SVN' and a yellow ribbon.
- Comment Field:** A large text area for entering a comment.
- Dropdown Menu:** 'Choose a previously entered comment or template:' with a dropdown arrow.
- Checkbox:** 'Keep Locks'.
- Button:** 'Paste selected names'.
- Table:** A table with columns: 'Resource', 'Content', 'Properties', and 'Treat as Edit'. The first row shows 'Project/ReadMe.txt' with a checked checkbox in the 'Resource' column and 'Modified' in the 'Content' column.
- Buttons:** 'Select All', 'Clear Selection', and 'Clear All'.
- Status:** 'Selected: 1 of 1'.
- Footer:** A question mark icon, an 'OK' button, and a 'Cancel' button.

Szikronizálás a repository-val

- Commit előtt megvizsgálhatjuk, hogy milyen változásokat is eszközöltünk az egyes fájlokon, ennek



```
Java Source Compare

Local File 33
1 package delegateMAS;
2
3 import be.kuleuven.distrinet.agentwise.traffic.delivery.model
4
5 public class GasLevelElement {
6     public GasStation gasstation;
7     public double consumption;
8
9
10    public GasLevelElement(GasStation gasstation, double const
11        super();
12        this.gasstation = gasstation;
13        this.consumption = consumption;
14    }
15 }
16

Remote File 33
1 package delegateMAS;
2
3 import be.kuleuven.distrinet.agentwise.traffic.delivery.mo
4
5 public class GasLevelElement {
6     public GasStation gasstation;
7     public double consumption;
8
9
10    public GasLevelElement(GasStation gasstation, double c
11        super();
12        this.gasstation = gasstation;
13        this.consumption = consumption;
14    }
15 }
```



Házi feladat beadási konvenciók

- Projektek elnevezése:
 - <shibboleth azonosító>_hf_<hf sorszáma>
- Beadás SVN repositoryba:
 - <http://repo.itk.ppke.hu/java/<shibboleth azonosító>>
- A projektből a következő fájlokat kell beadni (NEM tömörítve!)
 - .project
 - .classpath
 - src mappa tartalma (rekurzívan)
- Mindegyik házit a megfelelő (hf sorszám, wiki alapján) mappába kell tenni



Feladat G01F01

- Írj kettő osztályt és egy main függvényt az osztályaid tesztelésére
 - Írj egy kettő dimenziós **Vector** osztályt
 - Tudja tárolni az x és y koordinátákat (konstruktoron keresztül inicializálva)
 - Két metódus: hossz számítás, valamint skalár szorzat
 - Írj egy komplex számokat reprezentáló **Complex** osztályt
 - Imaginárius és képzetes tagot tárol (konstruktorban inicializálva)
 - Három metódus: összeadás, kivonás, szorzás
 - Main osztály **main** metódusa
 - Tesztelje a fenti két osztály lehető legtöbb funkcióját/esetét
 - NEM szükséges konzol olvasás, csak kiírás



Feladat G01F02

- Hozz létre egy n hosszú tömböt
 - Töltsd bele az első n prímszámot
- Másold le a tömböt
 - Emeld négyzetre a másolat elemeit, kivéve ha a szám háromjegyű
 - Ha háromjegyű, akkor legyen a tömbben az új érték nulla
- Ezt követően írd ki a Pascal-háromszög n -edik sorát

- Hozz létre egy tetszőleges méretű mátrixot tárolni képes osztályt az alábbi műveletekkel:
 - Mátrixok összeadása, szorzása
 - Skalárral szorzás, Transzponálás
 - Mátrix átformálása: meg lehessen adni két számot, amely a megadott méretűvé átalakítja a mátrixot
 - Ha nem egyenlő a régi és új mátrixban az elemek száma saját belátásod szerint cselekedj (Lehet kipótolni 0-val, illetve elhagyni értékeket.)
 - Részmatrix visszaadása: kezdősor és kezdőoszlop koordináták és méretinformációkkal
 - A kivágott mátrixot egy új példányban kell visszaadni
- Tetszőleges eltárolási módot használhatsz.

- Írj egy programot, amely ritkás mátrix tárolását teszi lehetővé.
 - A ritkás mátrix tárolási módját az adatszerkezetek és algoritmusok tárgyon tanult módszerek közül válaszd ki.
 - Legyen lehetőség (függvény), amellyel új elemet lehessen megadni a mátrix esetén, valamint törölni
 - Itt a törlés és a nem specifikált elemek 0 értékűek.
 - Példa
 - $$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$



Feladat G01F05

- Egy olyan program írása, amely képes meghatározni egy tetszőleges ország pénznemén (és érméin), hogy egy meghatározott összeget, hogyan lehet a lehető legkevesebb érmével felváltani.
 - Tehát legyenek az érméink:
 - $C = [1 \ 2 \ 5 \ 10 \ 20 \ 50 \ 100 \ 200]$
 - $S = 231$;
 - Ekkor $x = [1 \ 0 \ 0 \ 1 \ 1 \ 0 \ 0 \ 1]$
 - Összesen 4 érme kell.
- A feladatra egy osztály készíts el, amely a konstruktorában kapja meg az érmék paramétereit és egy `int[]` `szamol(int s)` függvényívás végzi el a számítást.
- A két legtriviálisabb megoldás
 - Kimerítő keresés: nem ér pontot
 - Mohó algoritmus: 15%-át éri a pontoknak, mivel nem teljesen jó megoldást ad.
- Gondolj az alábbi példára is:
 - $C = [1, 5, 20, 25]$
 - $S = 42$



Feladat G01F06

- Hozz létre egy tetszőleges méretű numerikus táblázatot tárolni képes osztályt az alábbi műveletekkel:
 - Sor beszúrása és törlése
 - Oszlop beszúrása és törlése
 - Sorátlag és oszlopátlag számítása
 - Sorok, oszlopok felcserélése
 - Résztáblázat visszaadása – másolattal új táblázatba
- Tetszőleges eltárolási módot használhatsz.

- Első feladatok
 - Ismételd át a tananyagot, állítsd be az Eclipse-t, az SVN plugint, stb.
 - Fontos, hogy a workspace UTF-8 karakterkódolást használjon!
 - Tartsd be a fenti névkonvenciókat és lehetőleg a Java névkonvencióit is!
- A konzolról bekért dátum alapján írasd ki a hét napját.
 - Készíts egy függvényt, ami meghatározza egy megadott dátumhoz a hét napját.
 - Egy külön main osztályban próbáld ki a függvényt, legalább három dátummal.
 - Add meg a függvénynek a dátumot év, hónap, nap formában, majd írasd ki a végeredményt a konzolra.
 - A hónapok, és a hét napjai legyenek enum-ok