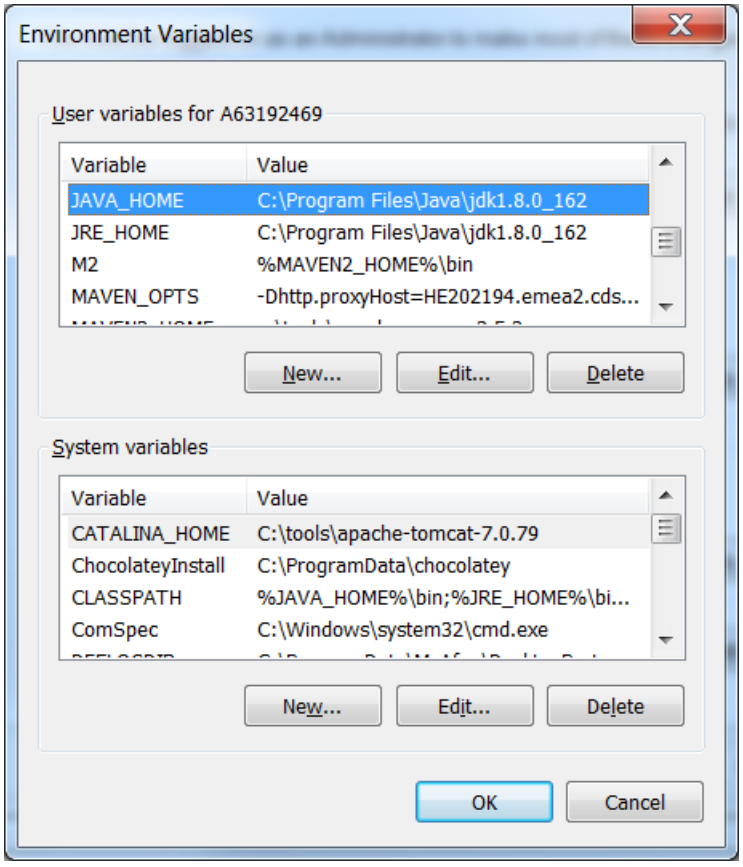
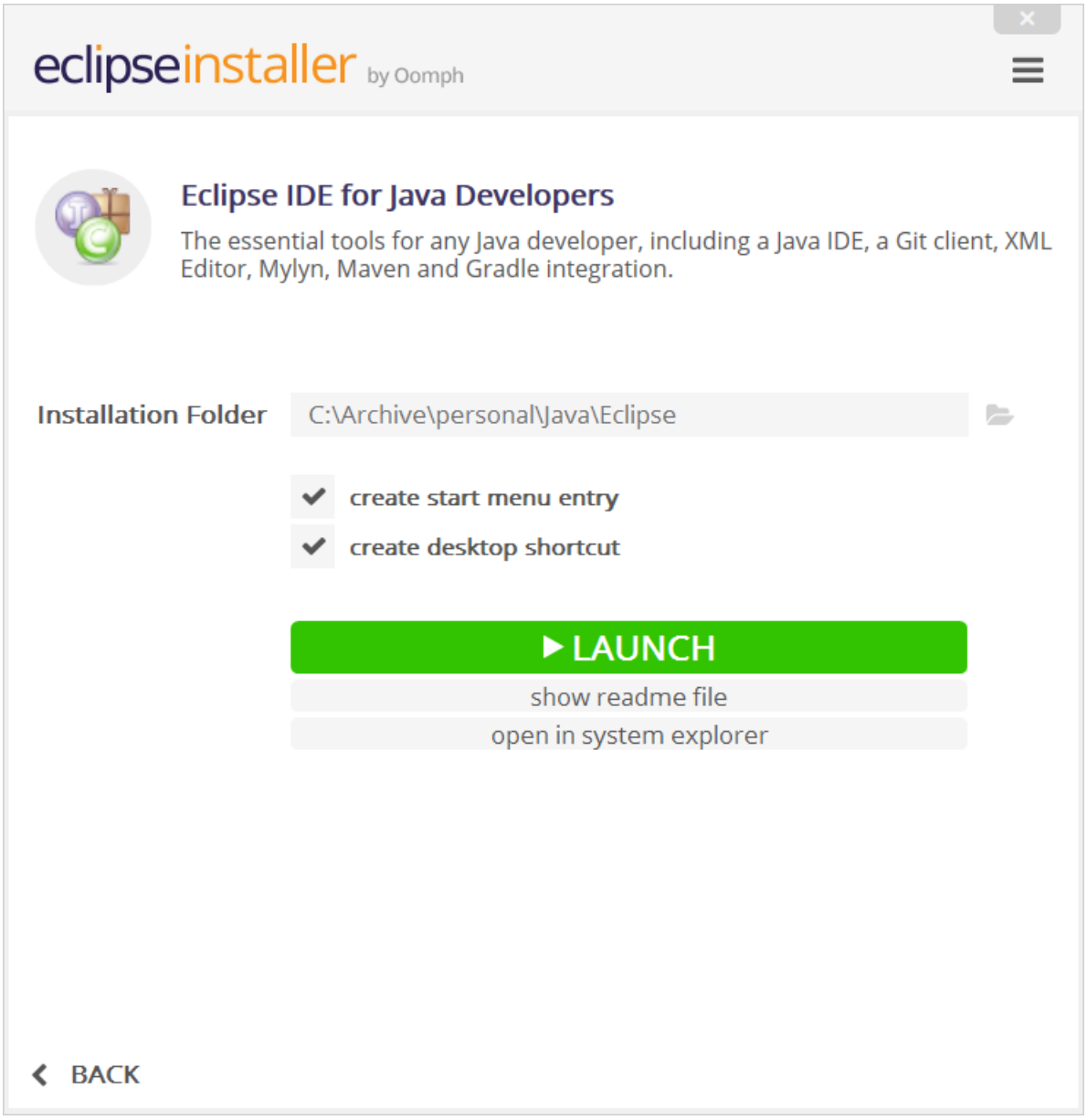
Bevezetés a Java programozásba 1.

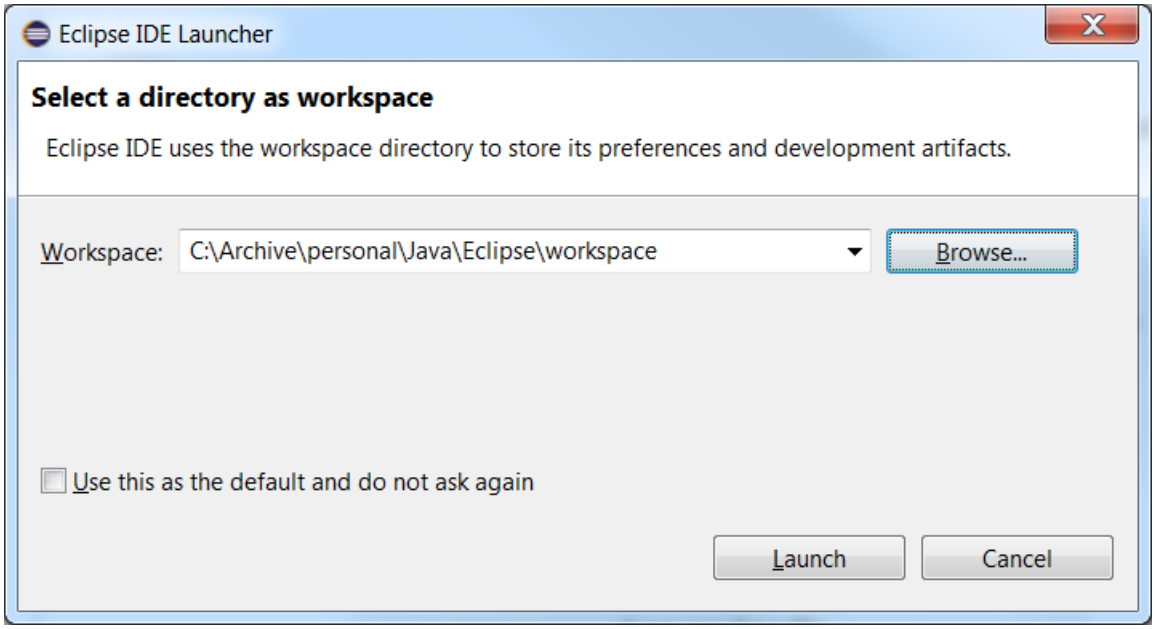
Java környezet bemutatása (2 óra)

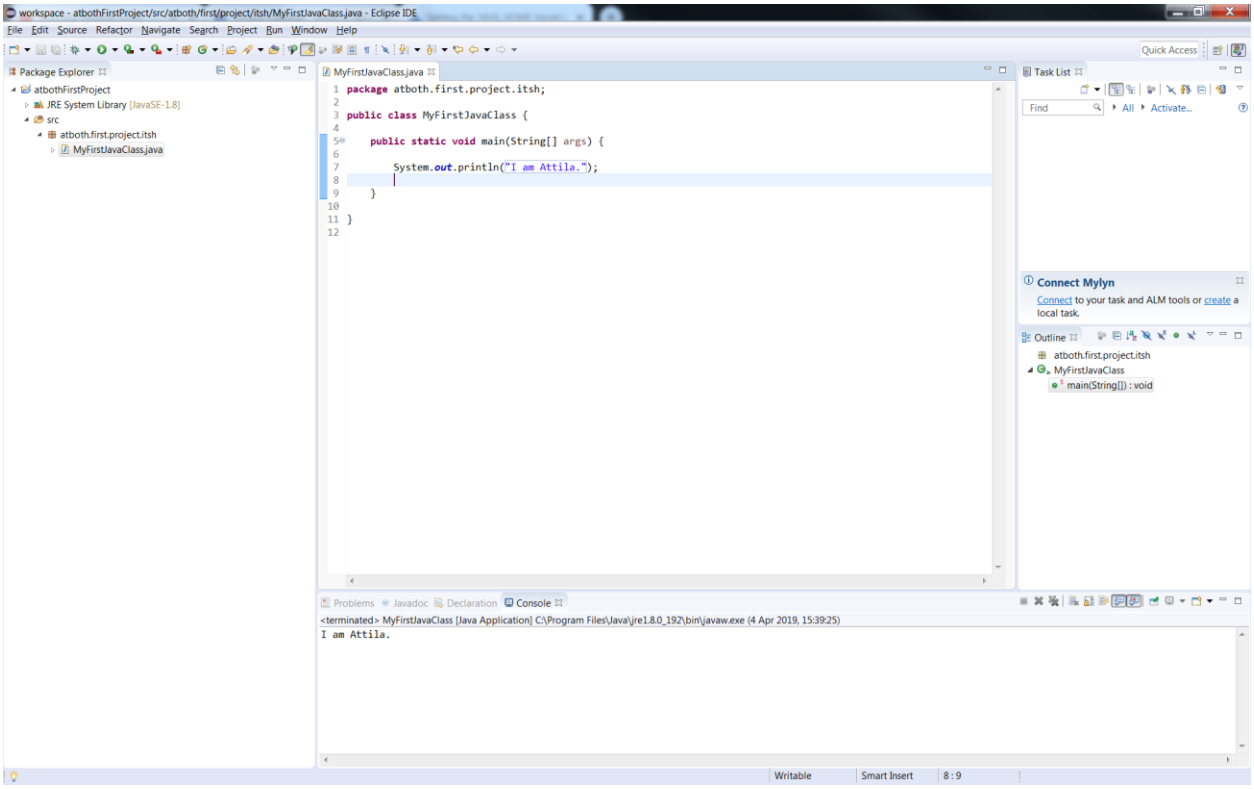
- A Java nyelv, JVM, JDK, JRE
- Eclipse IDE vagy IntelliJ fejlesztői környezet telepítése
    1. https://www.eclipse.org/downloads/
- JAVA_HOME környezeti változó beállítása
    1. https://www.java.com/en/download/installed.jsp
    2. `C:\Program Files\Java (for 64 bit architecture) or C:\Program Files (x86)\Java (for 32 bit architecture`
    3. Setting JAVA_HOME and JRE_HOME
    4. 

    5. Path: C:\Program Files\Java\jdk1.8.0_162\bin
    6. open cmd: java –version

- Gyakorlati feladat - Első Java projektem és programom



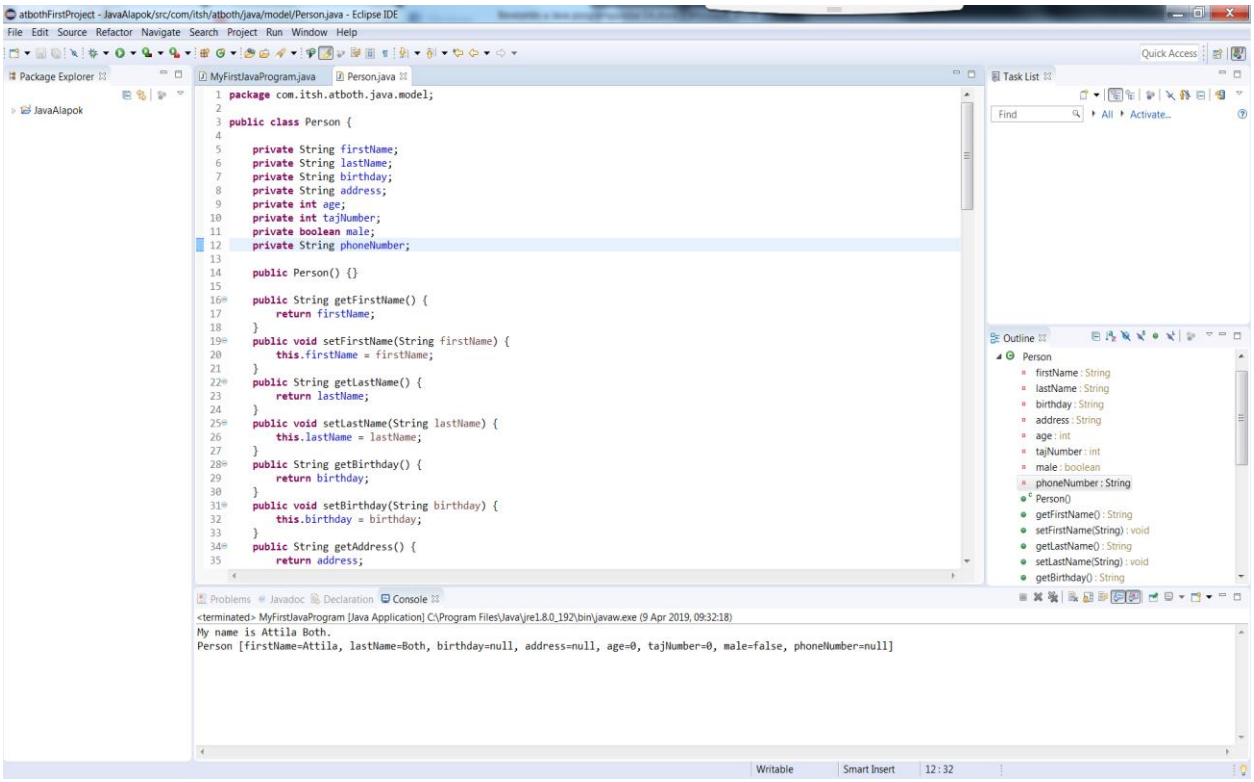

Eclipse IDE Launcher
Select a directory as workspace
Eclipse IDE uses the workspace directory to store its preferences and development artifacts.
Workspace:
C:\Archive\personal\Java\Eclipse\workspace
Browse...
Use this as the default and do not ask again
Launch
Cancel


workspace - atbothFirstProject/src/atboth/first/project/itsh/MyFirstJavaClass.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
atbothFirstProject
JRE System Library [JavaSE-1.8]
src
atboth.first.project.itsh
MyFirstJavaClass.java
MyFirstJavaClass.java
package atboth.first.project.itsh;
public class MyFirstJavaClass {
public static void main(String[] args) {
System.out.println("I am Attila.");
}
}
Task List
Connect Mylyn
Connect to your task and ALM tools or create a local task.
Outline
atboth.first.project.itsh
MyFirstJavaClass
main(String[]) : void
Problems Javadoc Declaration Console
<terminated> MyFirstJavaClass [Java Application] C:\Program Files\Java\jre1.8.0_192\bin\javaw.exe (4 Apr 2019, 15:39:25)
I am Attila.
Writable
Smart Insert
8 : 9

Java nyelv szintaxisa (2 óra)

1. Csomagok és Java file struktúra (Packages, Java file structure)

   Hozzuk létre az alábbi osztályt! **Person**



2. Primtív adattipusok (Primitive Data Types)

   byte – short – int – long – float – double char boolean

3. Operátorok (Operators)
   - relációs operátorok: **< > = != <= >=**
   - aritmetikai operátorok: **% * / - + ++ --**
   - logikai operátorok: && || !

4. Vezérlési utasítások (Flow control)

```
if (condition) {
} else {}

for (int i = 0; i < args.length; i++) {}
```

5. Metódusok (Methods)

   setter és getter metódusok,

   toString()

   hashCode()

   equals(Object obj)

6. A main() metódus

```
public static void main(String[] args) {
        // program utasítások
}
```

Objektum Orientált programozás Javaban (8 óra)

- Osztályok és Objektumok (Classes and Objects)
  - Gyakorlati feladat – *Hozzuk létre a Person nevű osztályt*
- Attribútumok és metódusok (Fields and Methods)
  - Gyakorlati feladat – *deklaráljuk a Person attribútumait és alap metódusait*
- Láthatóság (Modifiers) – *mutassuk meg, hogy a másik csomagban lévő osztály látja vagy sem a privát, publikus láthatóságú adattagot*
- Egységbezárás (Encapsulation)
  - Gyakorlati feladat – *write getter and setter methods for the Person class*
- Objektumok létrehozása és inicializálása (Creating and Initializing Objects)
  - Gyakorlati feladat – *példányosítsuk a Person osztályt és adjunk a személynek neve, cimet, st…*
- Elnevezés (naming convention)
  - Gyakorlati feladat – *javítsuk ki az adattagjaink neveit, metódusaink neveit, stb*
- Dokumentáció és megjegyzések (Java Documentation Comments)
  - */**/ or // or /** / this*
  - Gyakorlati feladat – *dokumentáljuk a Person osztályt és a main futtató osztályt*
- **!!! készítsük el az alábbi metódust:**
  - ***isTAJNumberCorrect() // if there are less or more numbers than 9 return false***
  - ***isTAJNumberValid() // if it contains anyting othern than numbers then return false***

Tipikus adatformák és kezelésük (1 óra)

- **Szöveg (Strings and String manipulation)**
  - *String name = "myName"; // creates one String object and one reference variable*
  - *String name = new String("myName"); // creates 2 objects and one reference variable, on object in normal (non-pool) memory (new String) and one "myName" literal will be in the pool*
  - *when the compiler encounters a new String literal it checks the pool to see if an identical String is already there. if a match is found then the reference to the new literal is directed to the existing String, and no new String object is created*
  - *toUpperCase(), toLowerCase(), trim(),*
  - *concat() and "+"*
  - Gyakorlati feladatok
    - irassuk ki a saját nevünket a konzolra. ATTILA both formában
    - fűzzük össze a telefonszámunkat és a TAJ számunkat '+' opeártorral
    - **irassuk ki a nevünket megfordítva!! (for ciklus vagy StringBuffer)**
- **Számok (Numbers)**
  - **Byte – Short – Integer – Long – Float – Double Character Boolean**
  - `AtomicInteger counter = new AtomicInteger();`
  - Gyakorlati feladat
    - play around with Boolean.valueOf()
    - try Integer.valueOf(), Integer.parseInt();…
    - count the number of digits in your address String and write it to the console

- **Dátum és idő (Java 8)**

```
LocalDate date = LocalDate.now();
System.out.println(date);

LocalTime time = LocalTime.now();
System.out.println(time);

LocalDateTime now = LocalDateTime.now();
System.out.println(now);

DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd
HH:mm:ss");
System.out.println(formatter.format(now));
```

  - Gyakorlati feladat
    - change your birthdate from String to LocalDate / LocalDateTime and modify your existing methods to make it work

Adatszerkezetek (2 óra)

- Tömbök
  - int[] tajNumberArray = new int[9]
  - Gyakorlati feladat
    - change your tajNumber attribute from String to int[]
    - write your tajNumber to console using for loop
- Listák
  - `private List<Person> budapestGroup;`
  - Gyakorlati feladat
    - készítsük el a fenti listát és adjunk hozzá résztvevőket
    - irassuk ki a csoport tagokat foreach() - használatával
- *Map és Set*
  - ez egy advanced témakör… de tudjanak róla, hogy van ilyen adatszerkezet is

```
Map<String, TrainingGroup> allJavaTrainings = new HashMap<String,
TrainingGroup>();
   TrainingGroup debrecenTrainingGroup = new TrainingGroup();
         allJavaTrainings.put("Debrecen", debrecenTrainingGroup );
```

Filekezelés és input / output (3 óra)

- Tipikus állományok kezlése (**txt, csv, json, xml**)

- Irjuk ki csv fileba a Person objectumainkat (Listából vagy egyéb adatszerkezetből).
  - `public static void writeContentToFile(List<Person> myFamily, String filePathWithName) {`
- Olvassuk be az előzőleg készített állomány tartalmát és irassuk ki a konzolra.
- Olvassuk be az előzőleg készített állomány tartalmát / készítsünk objektumokat / majd irassuk ki a tartalmat a konzolra
- Beolvasás billentyűzetről – kiiratás konzolra és fileba
  - `Scanner inputFromConsole = new Scanner(System.in);`
  - Gyakorlati feladat
    - Készíts egy programot, ami bekéri billentyűzetről az adatokat a Person objektum létrehozásához.
    - Olvassa be és hozza is létre egy Person objektumot. Irasd ki a konzolra.

Önálló tanfolyam záró feladat elkészítése (4h)

- **Készítsd el az ITSH gépjármű flotta kezelő rendszerét!!!**
- Hozz létre egy PoolCar osztályt! Vedd fel a pool autók alap adatait. (rendszám, tipus, szin, telephely, foglalt ettőleddig időpont, etc)
- Hozz létre egy 10 elemű listát amibe tegyél bele 10 céges auto példányt.
- Ird ki a konzolra az összes magenta szinű autót.
- Készíts egy file-t, amibe kiirod az összes pool autót, a tulajdonságait ';' –val sorold fel a fileba
- Amortizálj le egy pool autót (vedd ki a listából) és irasd ki újra a meglévő autókat.

Bevezetés a Java programozásba 2.

Java objektum orientált tervezés és programozás (haladóbb OO témakörök)

1. Öröklődés (Subclasses and Inheritance)
2. Interfészek (Interfaces)
3. Generikus programozás (Java Generics)
4. Enumok (Enums)
5. Lamba kifejezések (Lambda Expressions)
6. Kivételkezelés (Exception Handling)
7. Garbage collector (Garbage Collection)
8. Collections API
9. Lambda kifejezések és a Collections API
10. Többszálú programozás és szálkezelés