Welcome!

You need the following software installed on your computer:

- **Java** Development Kit (OpenJDK 21)
  https://adoptium.net/

- **IntelliJ IDEA** (Community Edition)
  https://www.jetbrains.com/idea/download/

(-possibly **Git** command line tool)
https://git-scm.com/downloads

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Introduction to the Java programming language

Compicampus - IT Courses for Students

Nico Waldispühl
MSc ETH CS

Last update: 2023-10-18

Goals

- Get a '**feeling**' for the language
- Get to know basic **tools** so that you could continue at home
- Learn basic Java **language constructs**
- Be able to **change**/improve existing programs

Download material

- **Slides (PDF):**

  https://java.retorte.ch
  (Please take tiny survey later *during the course!)*

Verify that required software is installed

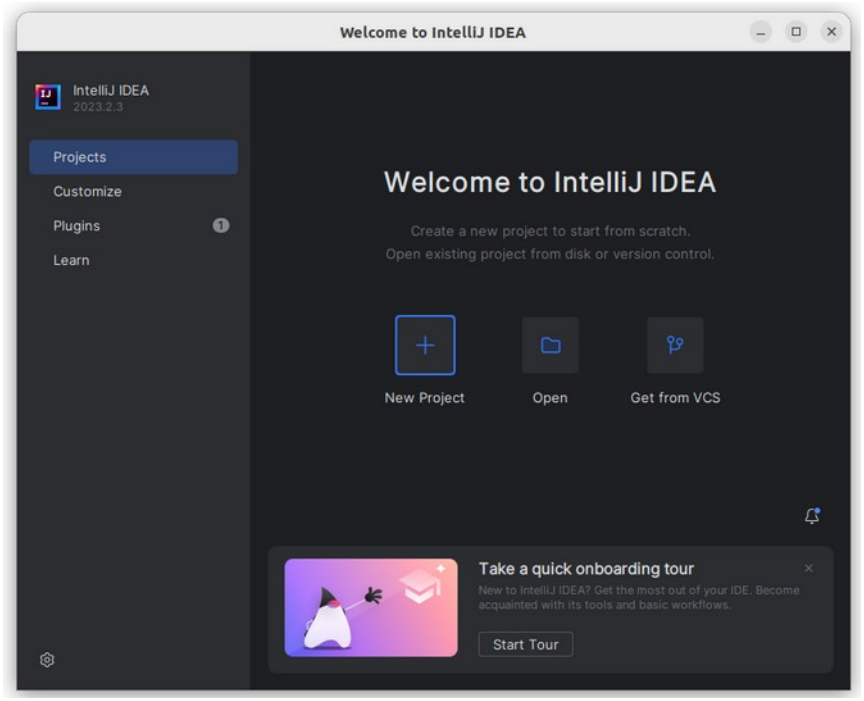
Java:

Open a terminal and enter '`java -version`':

```
$ java -version
openjdk version "21" 2023-09-19 LTS
[...]
```

IntelliJ IDEA Community Edition:

Software should start and look like this:


Welcome to IntelliJ IDEA
IntelliJ IDEA
2023.2.3
Projects
Customize
Plugins
Learn
Welcome to IntelliJ IDEA
Create a new project to start from scratch.
Open existing project from disk or version control.
New Project
Open
Get from VCS
Take a quick onboarding tour
New to IntelliJ IDEA? Get the most out of your IDE. Become acquainted with its tools and basic workflows.
Start Tour

Acquire source code

We need the example source code on our own computer. As example we use a `*projects*` folder where we place the source code:

projects/java-intro

or

projects/java-intro-master
(if you download it manually)

E.g:

Linux: */home/USERNAME/*projects*/java-intro*

Windows: *C:\Users\USERNAME\*projects*\java-intro*

Mac OS: */Users/USERNAME/*projects*/java-intro*

Acquire source code cont'd

With Git installed:

- Open a Terminal: Press *Windows Key* and start typing 'Terminal'. Click on the emerging icon labeled 'Terminal'.
- Execute the following commands (press *Enter* after every line):

```
mkdir -p projects     #(might already exist)
cd projects
git clone https://github.com/nwaldispuehl/java-intro.git
```

Note: If you decided to not install Git, see next slide for manual source code acquisition.

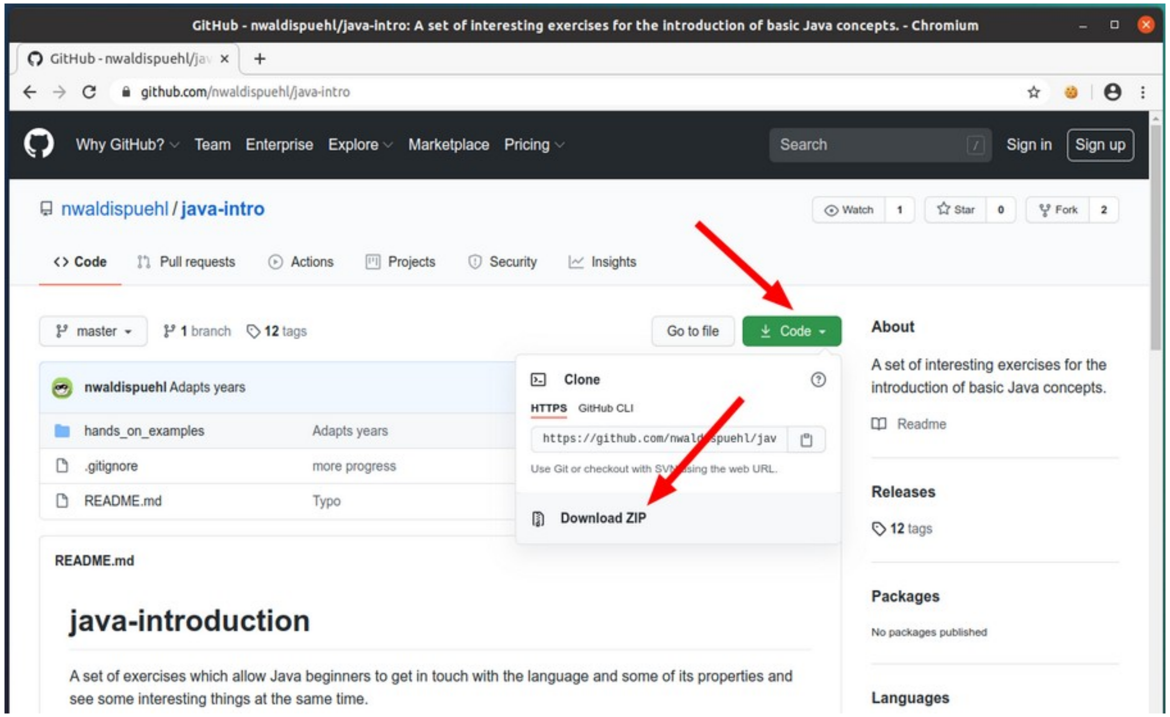
Acquire source code cont'd...

Manually:

- Surf to the repository with a web browser:

https://github.com/nwaldispuehl/java-intro

- Download the Zip archive of the code and extract it into your *projects* directory.


GitHub - nwaldispuehl/java-intro: A set of interesting exercises for the introduction of basic Java concepts. - Chromium
github.com/nwaldispuehl/java-intro
Why GitHub? Team Enterprise Explore Marketplace Pricing
Search
Sign in Sign up
nwaldispuehl / java-intro
Watch 1 Star 0 Fork 2
Code Pull requests Actions Projects Security Insights
master
1 branch 12 tags
Go to file
Code
Clone
HTTPS GitHub CLI
https://github.com/nwaldispuehl/jav
Download ZIP
About
Readme
Releases
12 tags
Packages
No packages published
Languages

nwaldispuehl Adapts years

hands_on_examples	Adapts years
.gitignore	more progress
README.md	Typo

Use Git or checkout with SVN using the web URL.

A set of interesting exercises for the introduction of basic Java concepts.

README.md

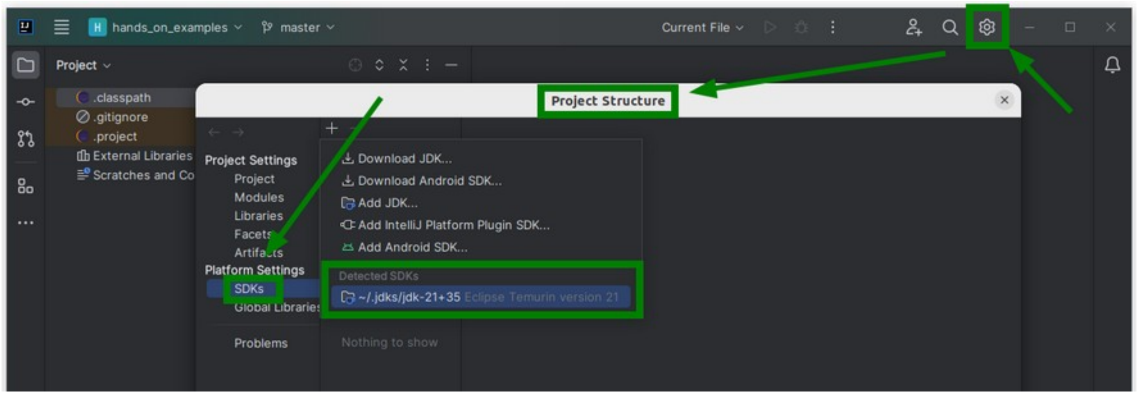
java-introduction

A set of exercises which allow Java beginners to get in touch with the language and some of its properties and see some interesting things at the same time.

*) *I*ntegrated *D*evelopment *E*nvironment

Open project in IntelliJ IDEA IDE*

- Start **IDEA**
- On the welcome screen: '**Open**'.
- Select '*projects/***java-intro/hands-on-examples**' directory, → '**Ok**'
- Select **'Trust project'**
- *The project is being opened*
- ⚙ → Project Structure... → SDKs → + → *Select your JDK*
  (or explicitly add it via 'Add JDK...'. It might already be selected.)
- Project → Set SDK
- Open 'Project View'

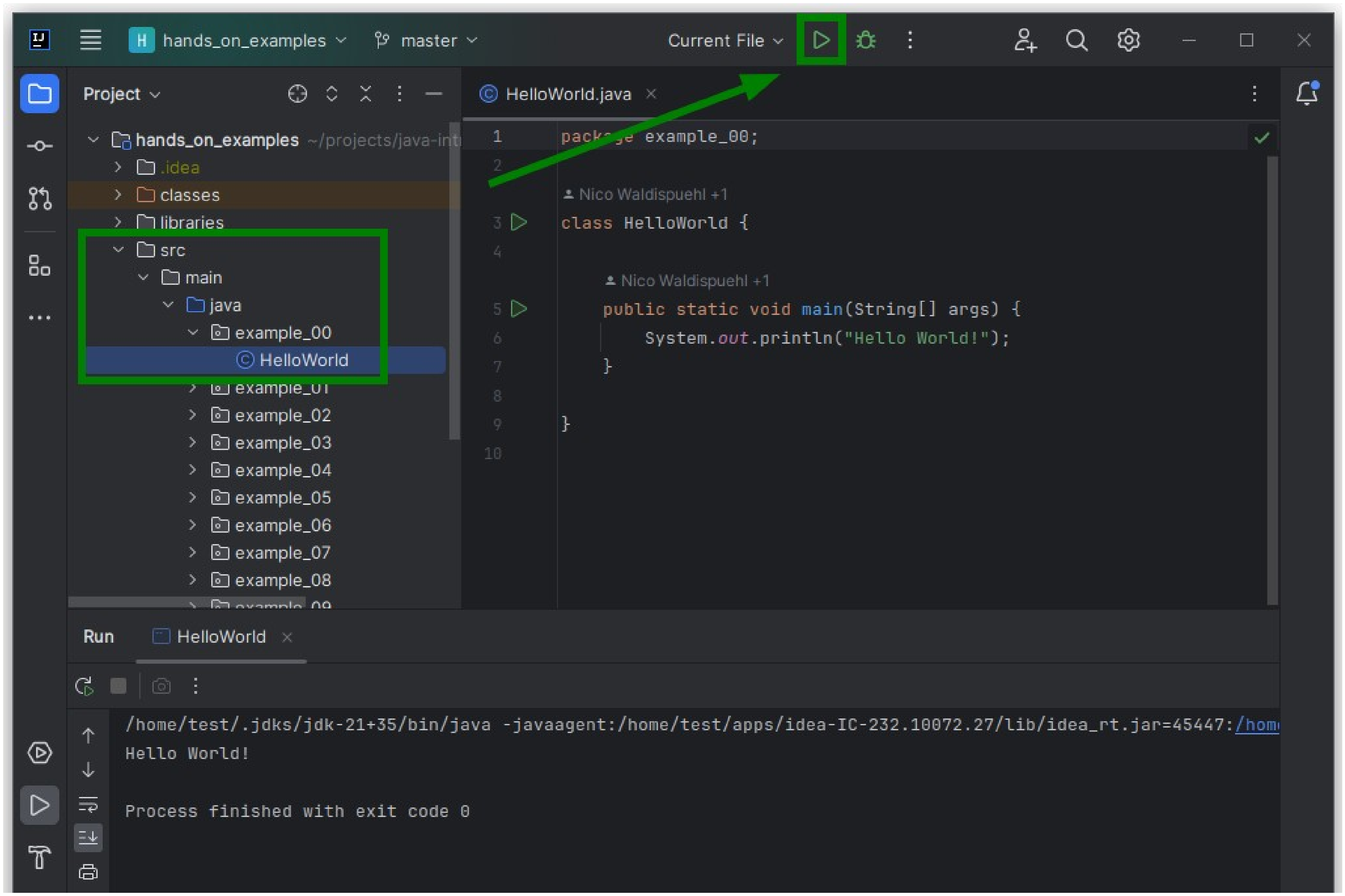


Hands-On

- Expand the package '**example_00**'
- Open the file '**HelloWorld.java**' *(e.g. with double-click)*

Task

Run the program.



```java
package example_00;

class HelloWorld {

    public static void main(String[] args) {
        System.out.println("Hello World!");
    }

}
```

```
/home/test/.jdks/jdk-21+35/bin/java -javaagent:/home/test/apps/idea-IC-232.10072.27/lib/idea_rt.jar=45447:/hom
Hello World!

Process finished with exit code 0
```

Hands-On

Now prepare your environments.

Goal: Every student has run the `Hello World` program.

What does a Java program look like?

```
public class HelloWorld {
    public static void main(String[] args) {

        System.out.println("Hello World");
    }
}
```

For comparison the same program in Python:

```
print "Hello World"
```

→ Java is more *verbose*, and thus more *explicit*.

Read more here: http://docs.oracle.com/javase/tutorial/getStarted/cupojava/

Play around a bit with the sample program

Use this print statement to try out some easy operations:

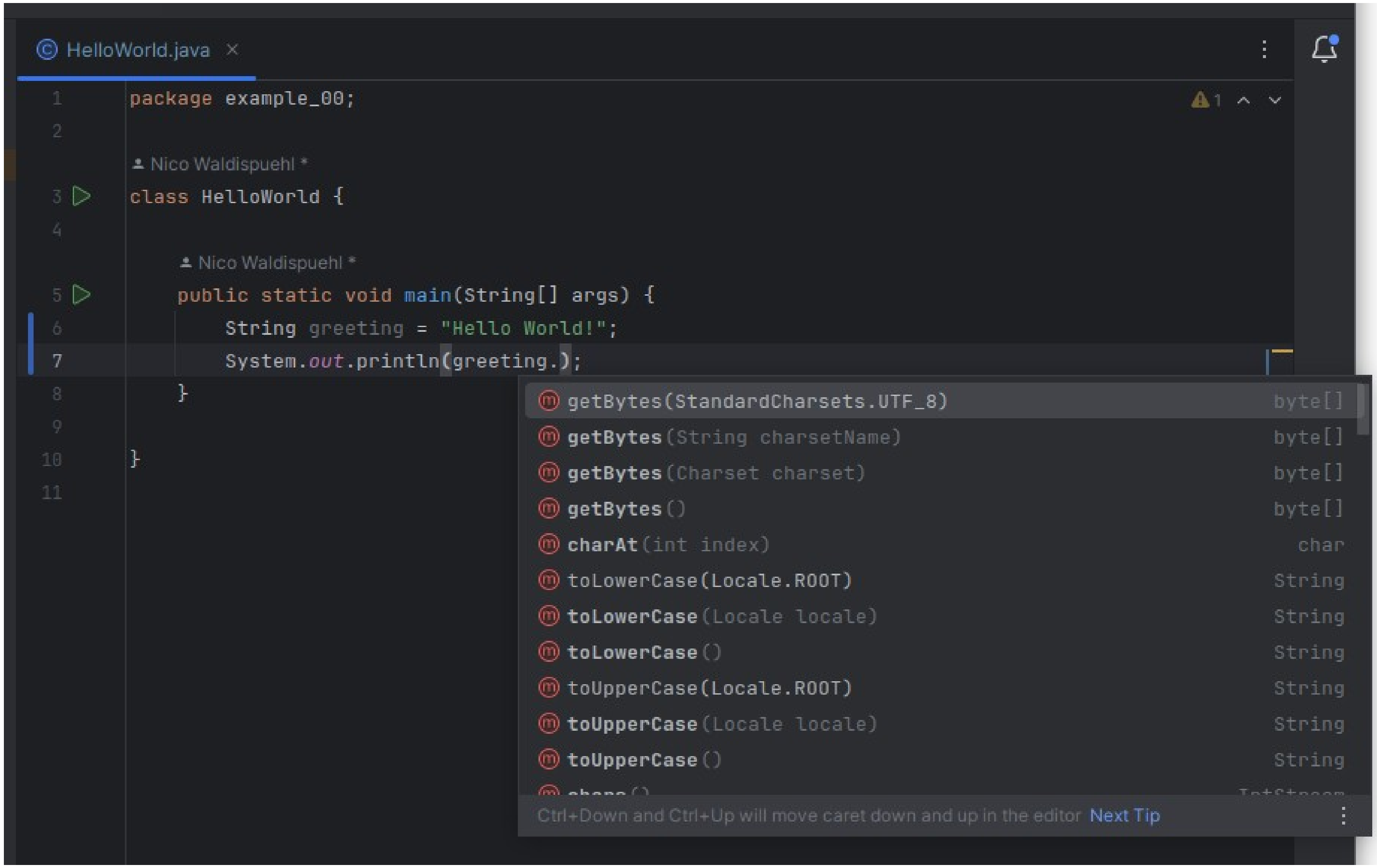
```
System.out.println( x );
```

Replace the '*x*' with these and check the output:

- **Arithmetic operations** ('calculations'):
  - Trivial ones: *1 + 1*, *500 / 0.001*, *3 * 3*
  - Extreme values: *1e300 * 1e200*, *2000000000 * 4*
- **Text manipulations**:
  - Concatenation: "*Hell*" + "*o*"
  - Method calls: "*Hello*"*.toLowerCase()*
  - Chained method calls: "*Hello*"*.toLowerCase().toUpperCase()*

How to take advantage of the IDE

The keystroke **Ctrl+Space** brings up a list of possible and recommended methods/calls for the current cursor position.



```
package example_00;

👤 Nico Waldispuehl *
class HelloWorld {

    👤 Nico Waldispuehl *
    public static void main(String[] args) {
        String greeting = "Hello World!";
        System.out.println(greeting.);
    }

}
```

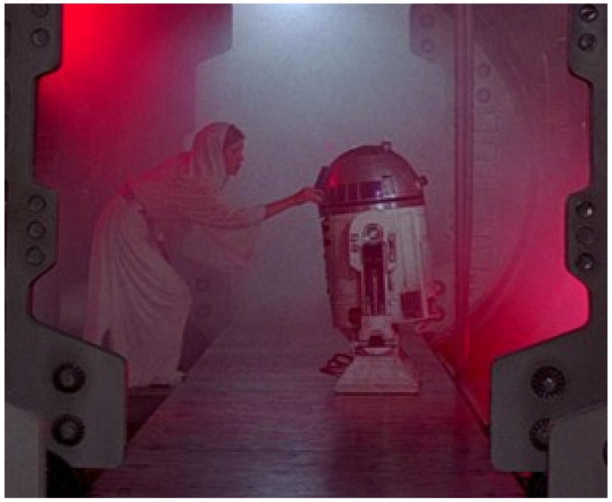
```
getBytes(StandardCharsets.UTF_8)        byte[]
getBytes(String charsetName)            byte[]
getBytes(Charset charset)               byte[]
getBytes()                              byte[]
charAt(int index)                       char
toLowerCase(Locale.ROOT)                String
toLowerCase(Locale locale)              String
toLowerCase()                           String
toUpperCase(Locale.ROOT)                String
toUpperCase(Locale locale)              String
toUpperCase()                           String
```

Ctrl+Down and Ctrl+Up will move caret down and up in the editor Next Tip

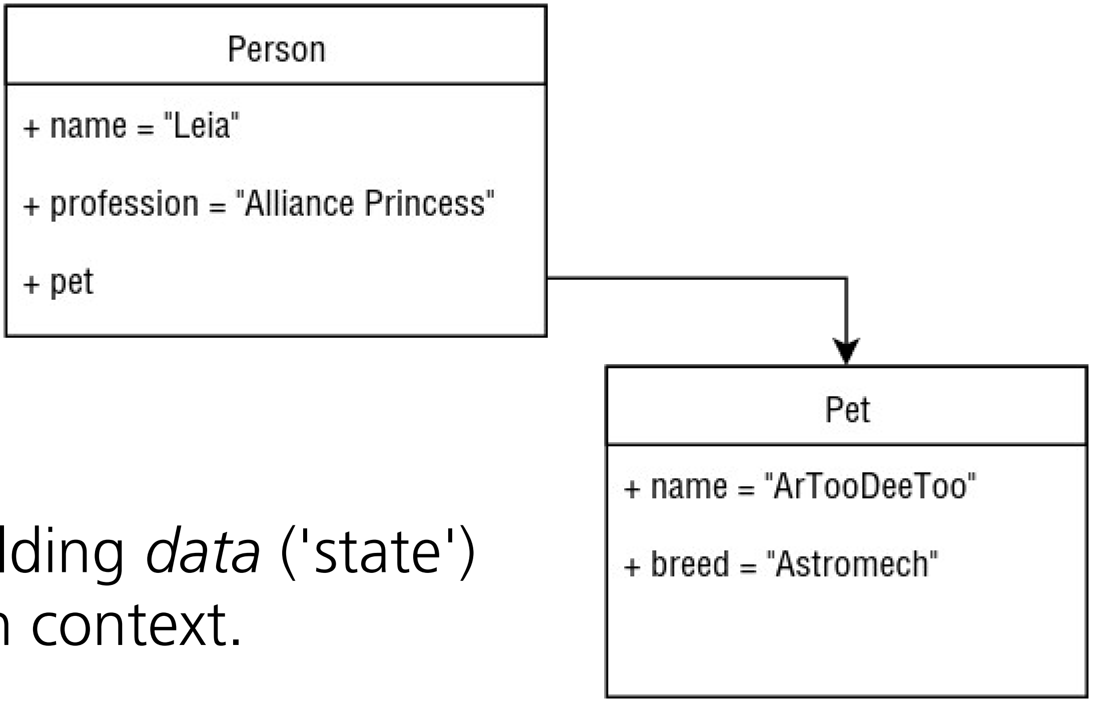
Java property: Object-orientation

In Java *everything is an object*; every real world 'thing' (we need in the software somehow) is modelled as 'object'.

- We tend to have less problems to think/talk about it this way.



vs

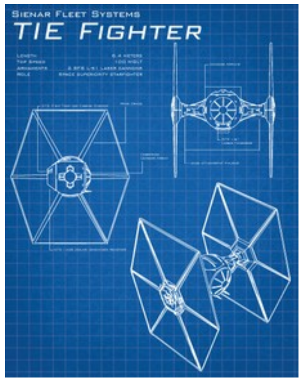


- An object is a 'container' holding *data* ('state') and *functionality* of a certain context.

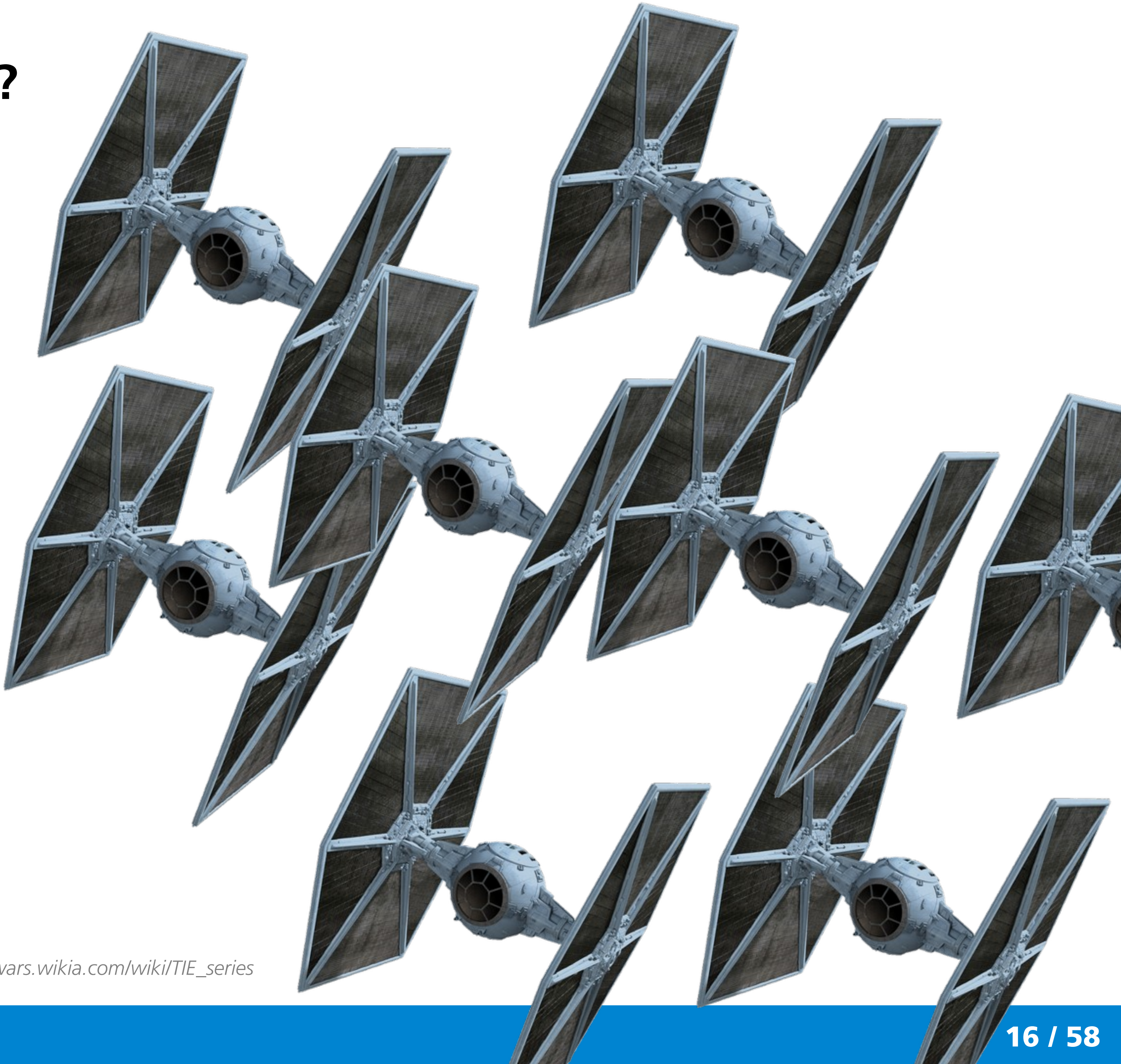
Sources: *http://starwars.wikia.com/wiki/R2-D2?file=Futureoftherebellion.png*

Java property: Object-orientation

- Class vs Object?



vs



Sources: *http://www.patrickkingart.com/, http://starwars.wikia.com/wiki/TIE_series*

Create an object

A Java object is instantiated with the use of the '***new***' keyword.

```java
new String("Hello");
```

Create some objects and assign them to respective variables:
(Note: We need to declare the type before the name.)

```java
// These are equivalent:
String greeting = new String("Hello");
String greeting = "Hello";

// These are equivalent:
Double piApproximation = new Double(3.1415926);
Double piApproximation = 3.1415926;

File textFile = new File("myTextfile.txt");
```

Types, Assignments, Operators

- **Types**
  - **Primitive types:**
    Integers: int
    int a = 5;

    Double prec. float: double
    double b = 3.5;

    Boolean value: boolean
    boolean isRight = true;
  - **Classes (Object types):**
    There are millions!! :)
    String myText = "Hello";
    Person bob = new Person();

- **Assignments: =**

  int aNumber = 5;
  Person alice = new Person();

- **Operators**
  Calculate: +, -, *, /, %
  x + y, z % 2

  Compare: ==, <, <=, >, >=, !=
  x == y, 0 < z

  Condition: &&, ||
  0 < x && x <= 10

  Negation: !

Note: Variable name must not be a keyword:
http://docs.oracle.com/javase/tutorial/java/nutsandbolts/_keywords.html

Read more here: http://docs.oracle.com/javase/tutorial/java/nutsandbolts/index.html

Primitive types vs object types

Primitive types

```
int age = 24;
```

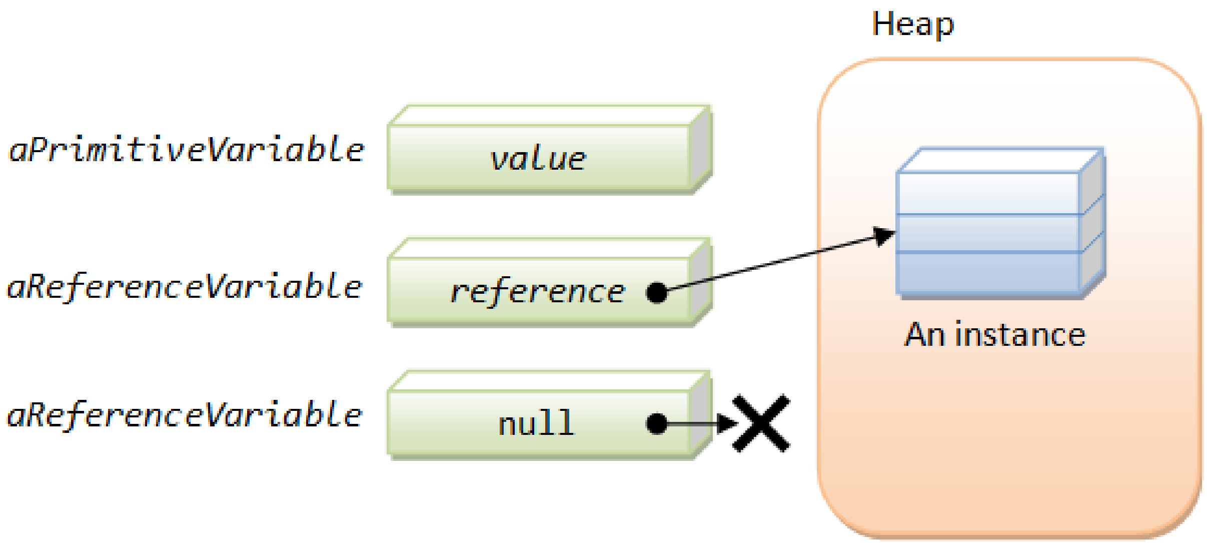
– Fit in a single 'memory cell' in the stack.

– Thus restricted in size (e.g 32bit for *int*)

Object types

```
Integer age = new Integer(24);

Person alice = new Person();
```

– Only a reference to the object is kept in the stack.



Sources: *http://www3.ntu.edu.sg/home/ehchua/programming/java/j3c_oopwrappingup.html#zz-7.1*

Control flow

- **Branching**
  - ***if – then* clause**
    if (boolean condition) { … }
    if (3 < x) { … }
  - ***if – then – else* clause**
    if (cond.) { … } else { … }
    if (x == 0) { … } else { … }
  - **may be combined:**
    if (cond.) { … }
    else if (cond.) { … }
    else { … }

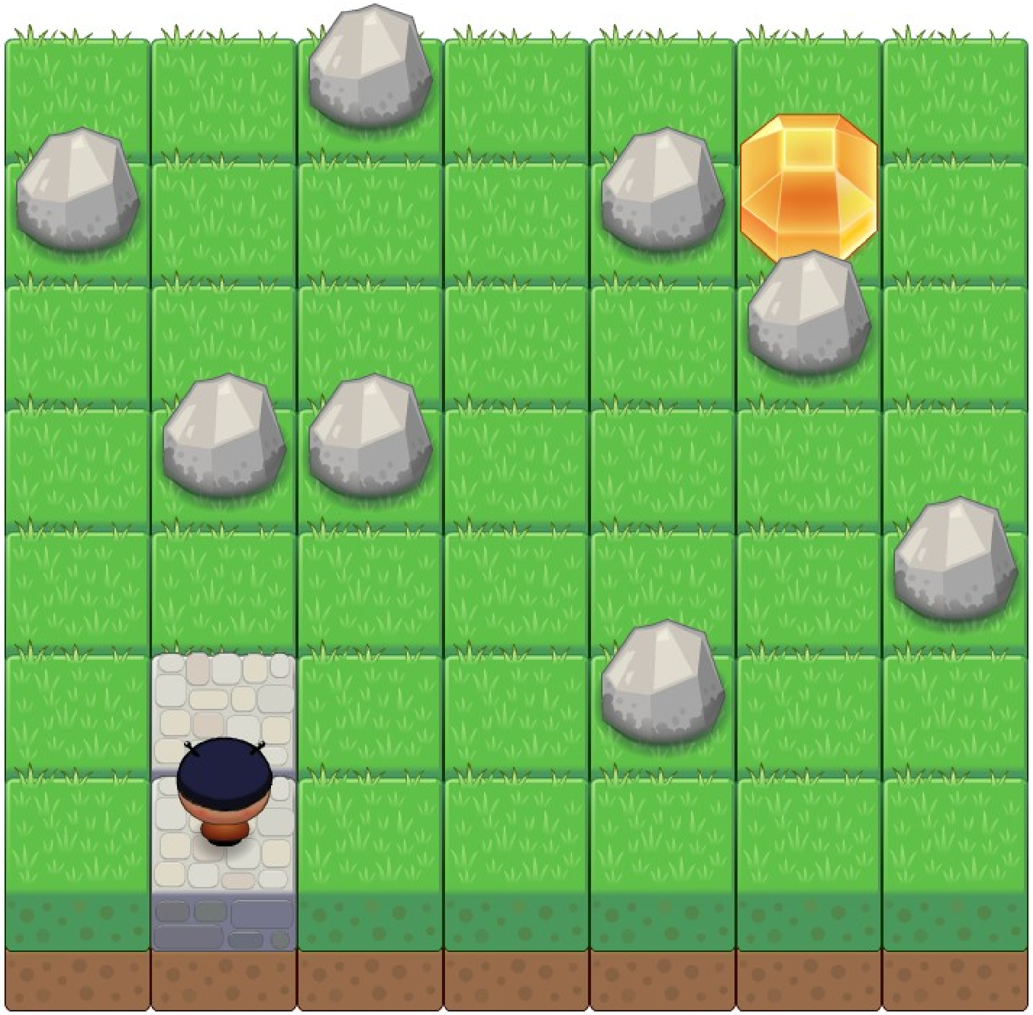
    *) where … denotes an arbitrary expression.

- **Loops**
  - ***for* loop**
    for (init; term ; incr) { … }
    for (int i = 0; i < 10; i++) { … }
  - ***while* loop**
    while (boolean condition) { . }

    ```
    int x = 0;
    while (x < 10) {
        x = x + 1;
    }
    ```
  - **Object iteration** (e.g. String)
    List<String> stringList = …
    for (String s : stringList) { … }

Read more here: http://docs.oracle.com/javase/tutorial/java/nutsandbolts/flow.html

Game: Treasure Hunt



Sources: *http://www.lostgarden.com/2007/05/dancs-miraculously-flexible-game.html (graphic tiles)*

Hands-On

- Expand the package '**example_01**'
- Open the file '**TreasureHunt.java**'

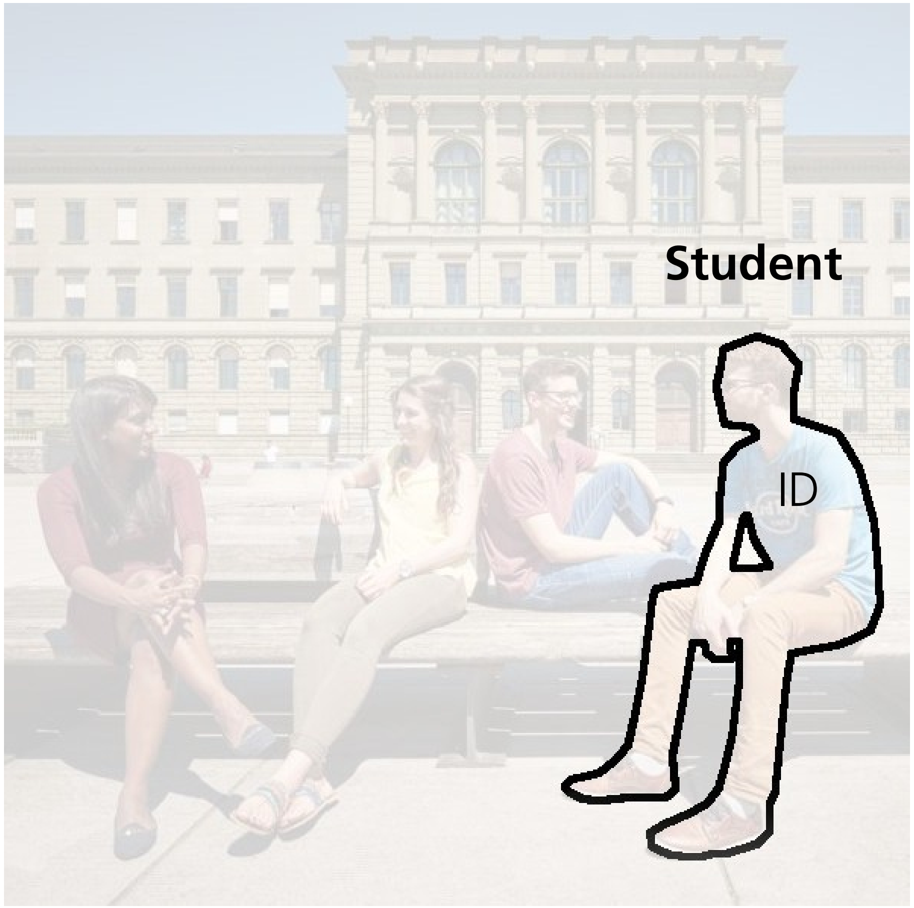
Tasks

- Run the program and observe.
- Open the file '**Avatar.java**'
- Enhance the method '**move()**' in the class '**Avatar**' with directions so that the character in the game catches the treasure.

Own program: Student Exam Manager

- Writing software = apply abstraction to the real world
  → Keep only the relevant parts.
- First, create data **model**, then **functionality**.

→



Anatomy of a Java program

Person.java

```java
public class Person {

  private String name;
  private int yearOfBirth;

  public Person(String name, int yearOfBirth) {
    this.name = name;
    this.yearOfBirth = yearOfBirth;                    Constructor
  }

  public int getAgeIn(int year) {                                   Access modifier
    return year - yearOfBirth;                 Method               Keyword
  }                                                                 Type
}                                                                   Class variable
                                                                    Local variable
```

Parts of Program.java

```java
Person ronald = new Person("Ronald", 2001);
int ronaldsAge = ronald.getAgeIn(2023);
System.out.println("Age: " + ronaldsAge);
```

Hands-On

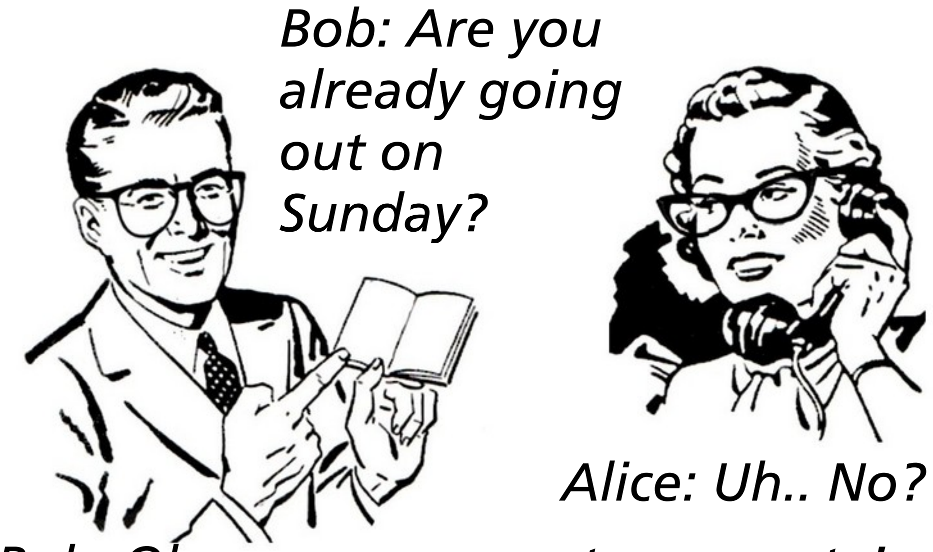
- Expand the package '**example_02**'
- Open the files '**Person.java**' and '**Program.java**'

Tasks

- Run the program.
- Enhance the class '**Person**' with a new *method* '**getName**'.

Java property: Object-orientation

– **Everything is an object in Java. OO means: Information hiding, asking someone to do something**



Bob: Ok, you can come to my party! **vs**

```
if (list.length == 0) {
    print "empty";
}
```

```
if (list.isEmpty()) {
    System.out.println("empty");
}
```

Read more here: http://en.wikipedia.org/wiki/Object-oriented_programming

Sources: *http://aliceandbobcurate.files.wordpress.com/2012/02/ask.jpg*

Hands-On

- Expand the package '**example_03**'
- Open the files '**Person.java**' and '**Program.java**'

Tasks

- Run the program.
- Enhance the method '**compareAgeWith**' of the class '**Person**' in a way that it returns the proper answer.
- Does the answer remain correct if you change names and birth years of the person instances?

Some more Java facts

- Current standalone Java version: **Version 8 Update 381**
  (New versioning scheme: two versions per year; currently: 21)
- **JRE** (Java Runtime Environment) aka **'Java'** *(<50 MB)*
  This needs to be installed to *run* Java programs.
  http://www.java.com/
- **JDK** (Java Development Kit) *(>120MB)*
  This needs to be installed to *write* Java programs.

- Mostly used: **java** (application launcher), and **javac** (Java compiler).

Read more here: http://www.oracle.com/technetwork/java/index.html

How to be able to program Java at home?

- **To *run* the programs: Install the JDK**
  https://adoptium.net/
  or google for "jdk download"
- **To *edit* the programs: Install an IDE**

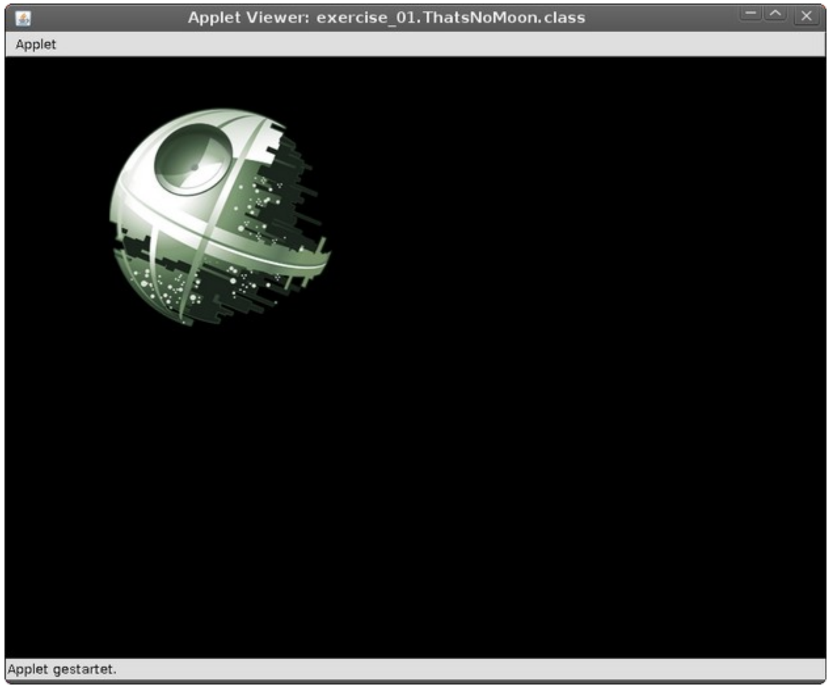
  E.g. IntelliJ IDEA: https://www.jetbrains.com/idea/

  There are others:

  - *Eclipse (https://www.eclipse.org/)*
  - *Netbeans (https://netbeans.org/)*

Hands-On

- Expand the package '**example_04**'
- Open the file '**ThatsNoMoon.java**'



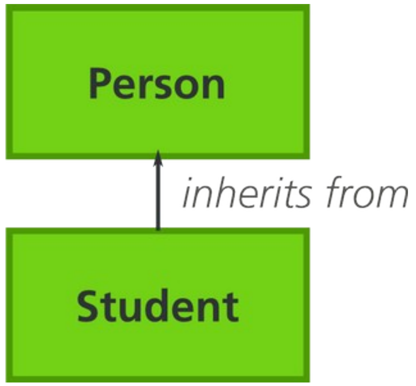
Tasks

- Run the program.
- Enhance the method '**updateValues**' of the class '**ThatsNoMoon**' in a way that the moon moves.
- Make the moon **change direction** when it hits a border.
- Introduce **gravity**: Let the speed change over time.

  Are you getting the moon **bouncing**?
  (Maybe you need dampening so it does not bounce too much?)

Inheritance

A powerful feature of object-orientation is inheritance. By **extending** another object, we inherit its properties.



Consequence: *A student is a person.*

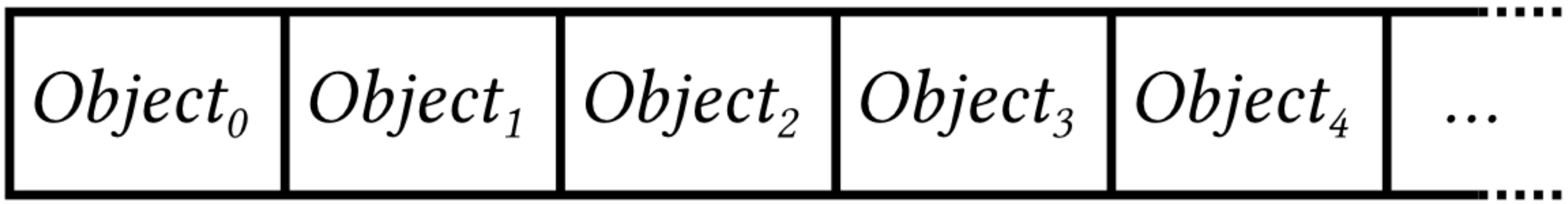
```java
public class Student extends Person {
    private String studentNumber;

    public Student(String sNr, String name, int yearOfBirth) {
        super(name, yearOfBirth);
        this.studentNumber = sNr;
    }

    public String getStudentNumber() {
        return studentNumber;
    }
}
```

Keyword

Note: Every class implicitly extends the class **Object**.

Read more here: http://en.wikipedia.org/wiki/Inheritance_(object-oriented_programming)

Basic data structures: List



Basic data structures: List cont'd

- **List**
  ArrayList is a popular implementation:

```
ArrayList<String> myList = new ArrayList<>();

// Usual operations:
myList.add("some string");
String fifthElement = myList.get(4);

// Iterate over list:
for (String s : myList) {
    // Do something with s.
}
```

Read more here: https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/List.html

Hands-On

- Expand the package '**example_05**'
- Open the files '**Program.java**', and '**FinalExam.java**'.

Tasks

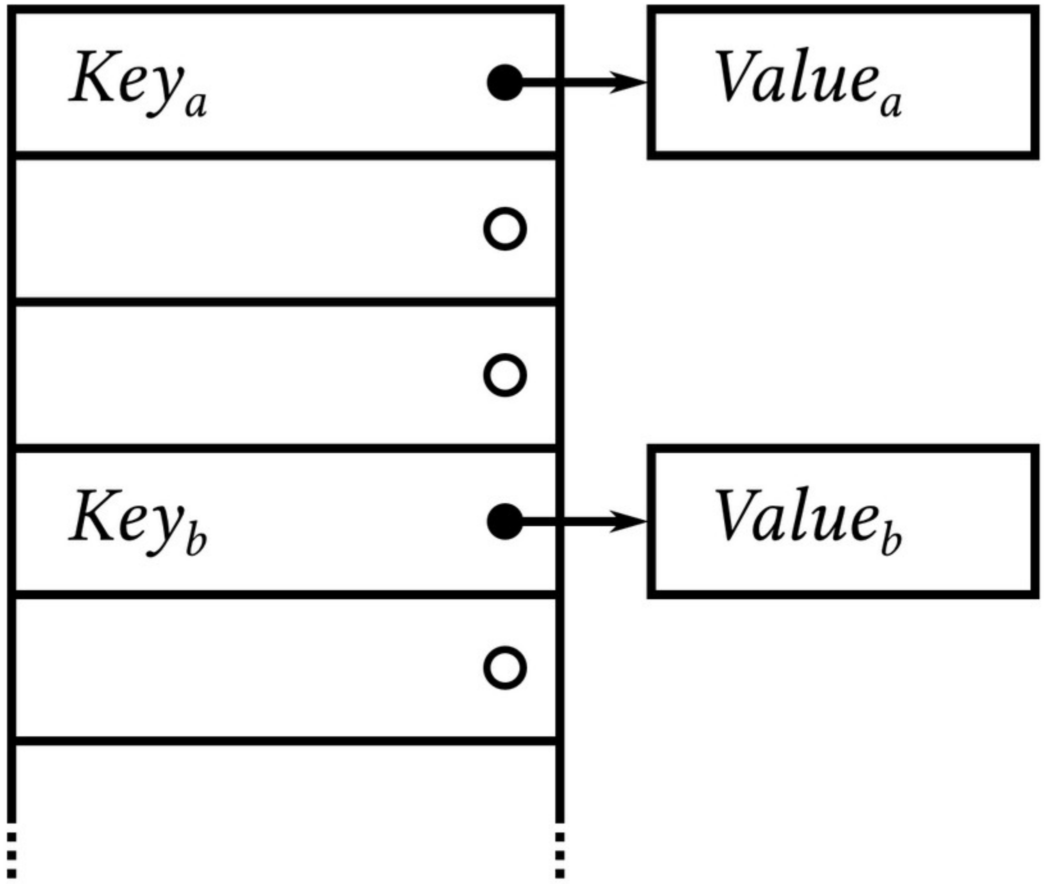
- Run the program.
- Complete the method '**printAcceptedApplicants**' in the class '**FinalExam**' in a way that a list of all eligible students is printed to the console.

End of the first part

See you tomorrow!

Basic data structures: Map

Other languages call it 'hash', or 'dictionary'.
(Works like a phone book.)



Basic data structures: Map cont'd

- **Map**
  HashMap is a popular implementation.

```
HashMap<String, Integer> myMap = new HashMap<>();

// Usual operations:
myMap.put("key", 123);
Integer value = myMap.get("key");

if (myMap.containsKey("key")) {
    // ...
}

// Iterate over map values (also possible for keys):
for (Integer value : myMap.values()) {
    // Do something with value.
}
```

Read more here: https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/Map.html

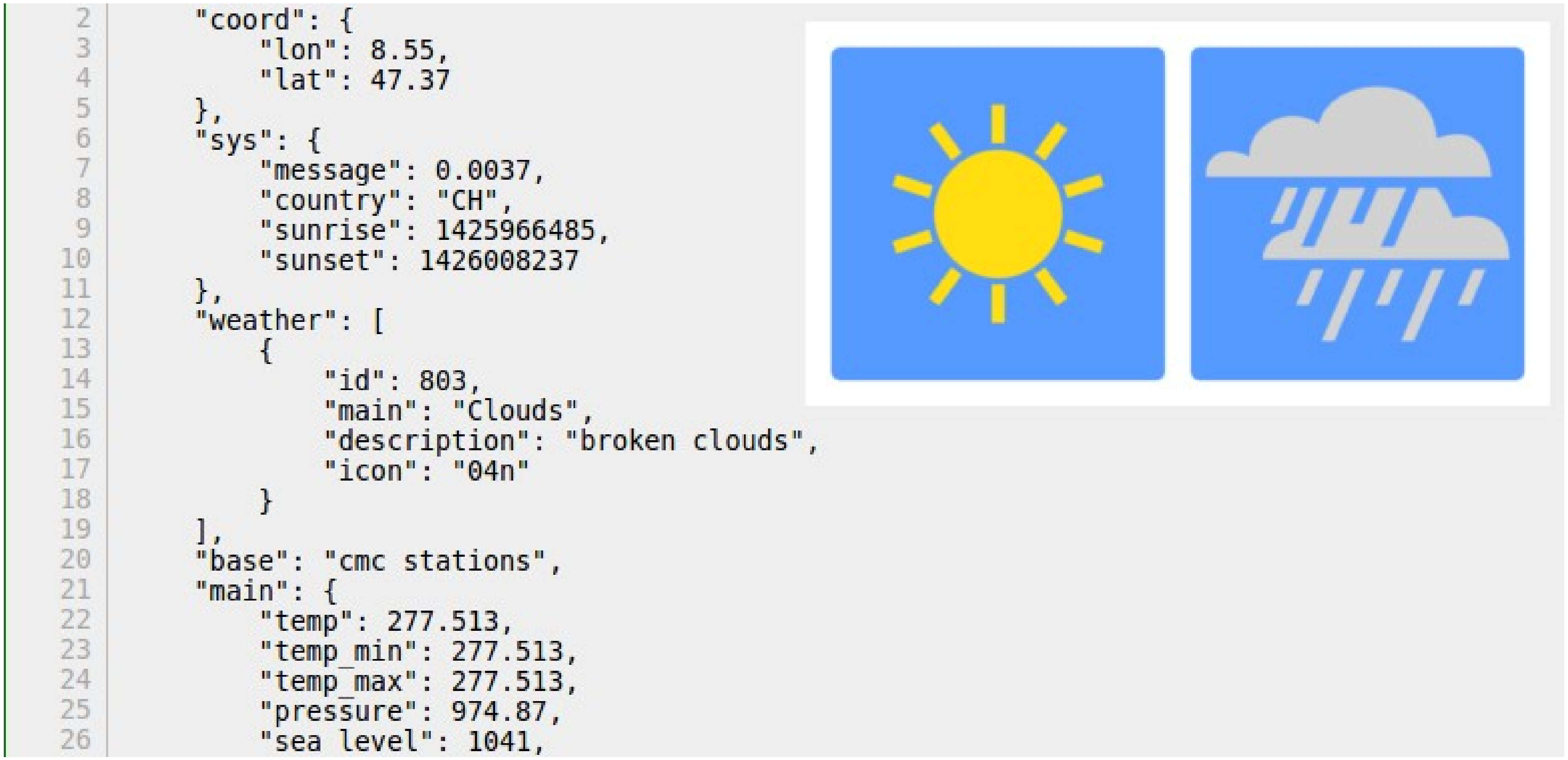
Hands-On

- Expand the package '**example_06**'
- Open the files '**Program.java**' and '**WordLengthFrequencyCounter.java**'.

Tasks

- Run the program.
- Implement the method '**calculateFrequencyTableFrom**' in the class '**WordLengthFrequencyCounter**' in a way that it stores a frequency table of the words length in the map '**frequencyTable**'.

Fetching information from the internet



```
    "coord": {
        "lon": 8.55,
        "lat": 47.37
    },
    "sys": {
        "message": 0.0037,
        "country": "CH",
        "sunrise": 1425966485,
        "sunset": 1426008237
    },
    "weather": [
        {
            "id": 803,
            "main": "Clouds",
            "description": "broken clouds",
            "icon": "04n"
        }
    ],
    "base": "cmc stations",
    "main": {
        "temp": 277.513,
        "temp_min": 277.513,
        "temp_max": 277.513,
        "pressure": 974.87,
        "sea_level": 1041,
```

Hands-On

- Expand the package '**example_07**'
- Open the file '**Program.java**'.

Tasks

- Run the program.
- Export it as so called .jar (Java archive) file.
- Run it from command line.

How to export an executable Java program

- Create a '**Run Configuration**' for your program (or just run it and look in the console how it was called, e.g. '*Program(5)*').
- **Right mouse-click** on the main program file, select '**Export...**'.
- Select 'Java' -> '**Runnable JAR file**'.
- Then, select then run configuration from above. And a meaningful filename, e.g. '*temperature.jar*'. Library: 'Package...'.
- Then you can start the program from the command line:

```
$ java -jar temperature.jar
Temperature in 'Zurich, CH': 13.18 °C
```

My opinion on Java use

Well-suited for:

- General data processing
- Simulations
- Games
- Android Apps
- Servers of all kind
- Desktop applications
- …

Less well suited for:

- Statistics (*use R*)
- Linear algebra (*use Matlab or Octave*)
- Mathematics (*use Maple, Mathematica, Maxima or other CAS*)
- Machine Learning (*use Python*)
- Quick'n'dirty string processing (*use Python, Bash, ~~Ruby~~, ~~Perl~~, JS*)
- Data Visualization
- …

Class vs Interface?

Sources: *http://en.wikipedia.org/wiki/File:HITACHI_1_ZOLL_C.jpg,http://ios.wonderhowto.com/how-to/3-music-player-apps-put-your-iphones-built-music-app-shame-0140654/*

Class vs Interface?

```
interface MusicPlayer {
    void play();
}
```

The interface is implemented by:

```
class TapeRecorder implements MusicPlayer {
    void play() {
        // Start music tape playing.
    }
    // ...
}
```

```
class SmartPhone implements MusicPlayer {
    void play() {
        // Load mp3 from storage and play it.
    }
    // ...
}
```

Keyword

Class vs Interface?

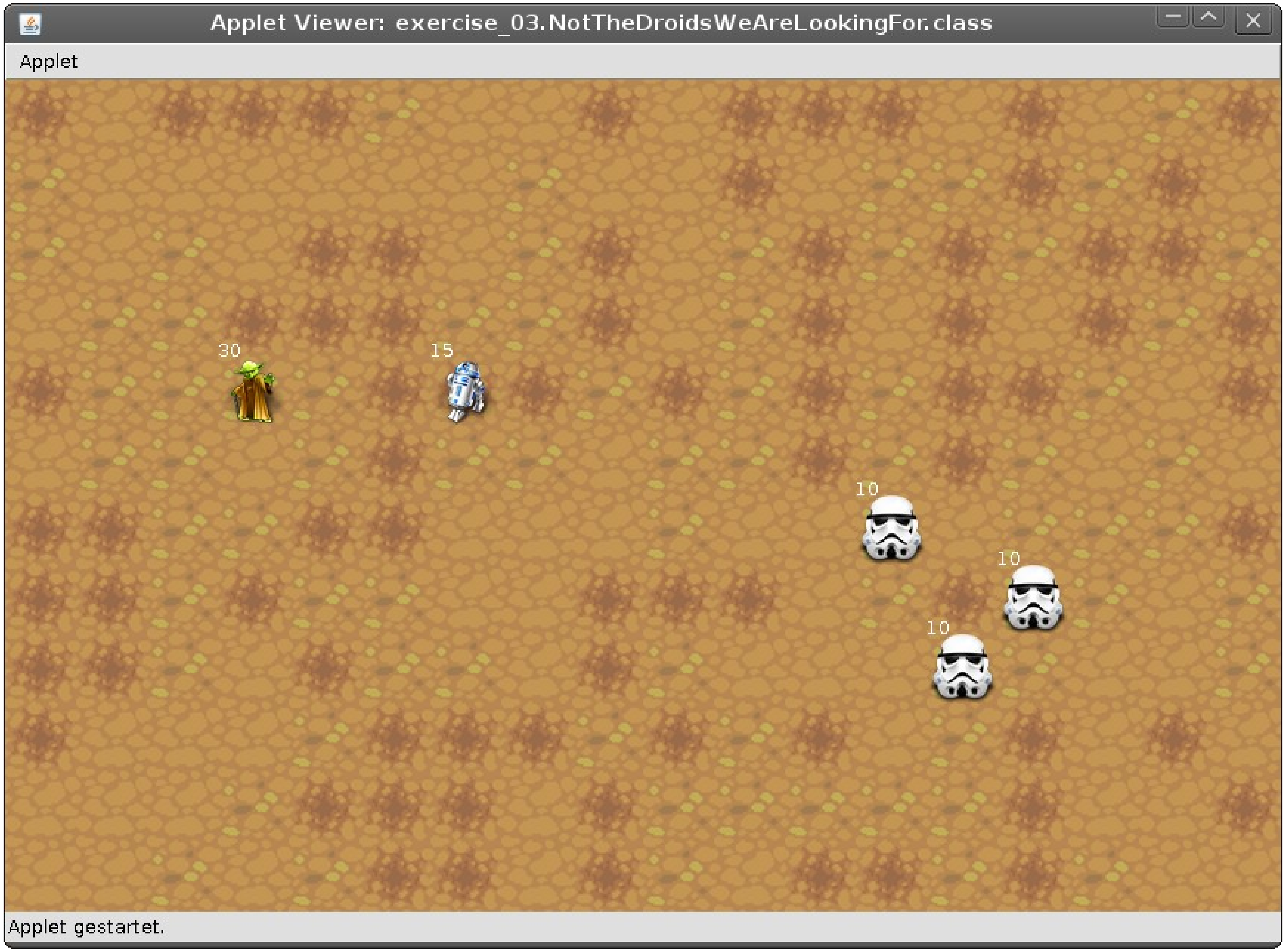
Why is that useful?

- Users may provide own implementations.
- Keep software as generic as possible.

```java
class Club {
    private MusicPlayer musicPlayer;

    void setMusicSource(MusicPlayer musicPlayer) {
        this.musicPlayer = musicPlayer;
    }

    void startParty() {
        musicPlayer.play();
    }
}
```

Read more here: http://docs.oracle.com/javase/tutorial/java/concepts/interface.html

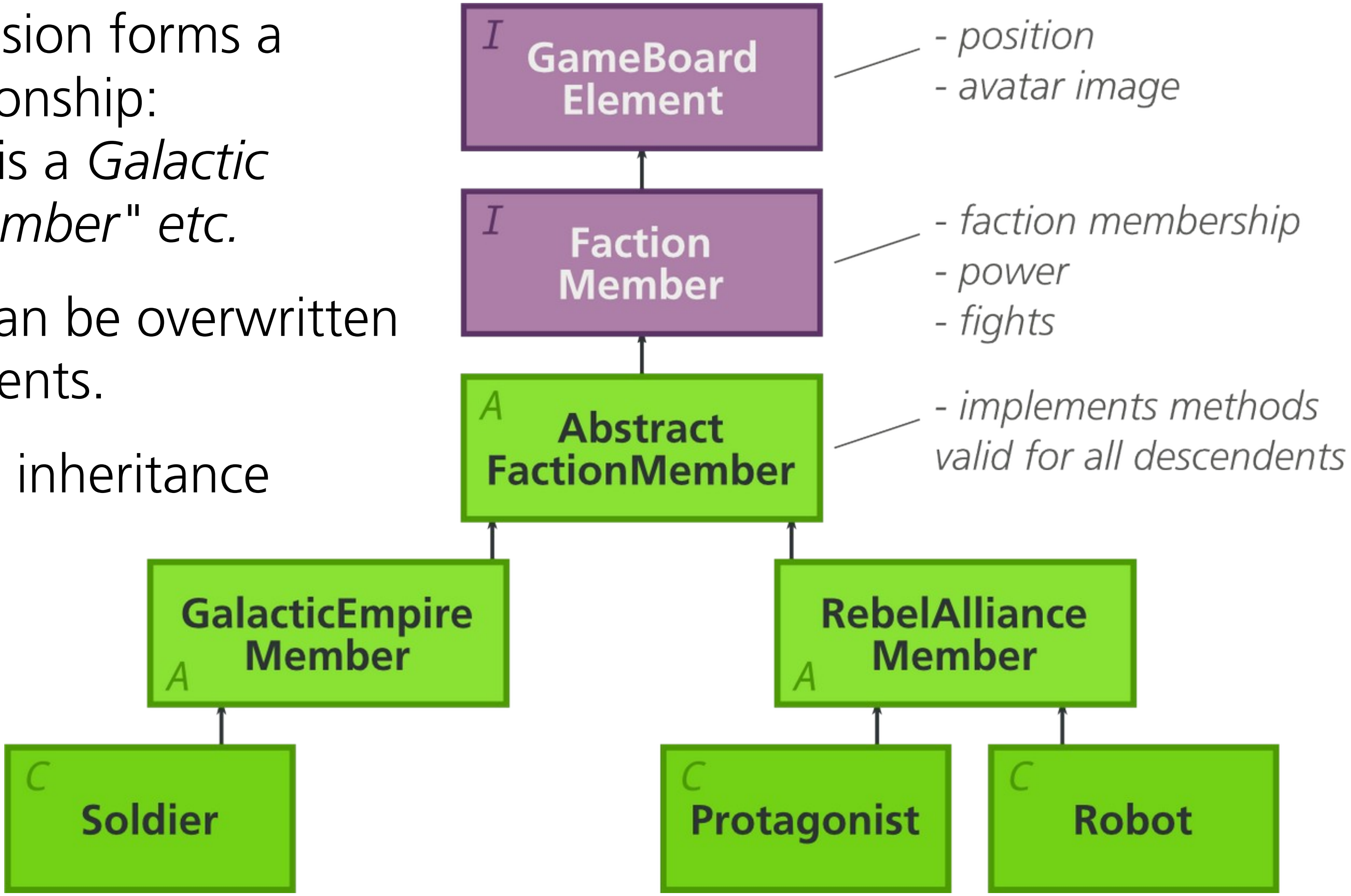
Game: Not the droids you're looking for


Applet Viewer: exercise_03.NotTheDroidsWeAreLookingFor.class
Applet
30
15
10
10
10
Applet gestartet.

Sources: *http://www.iconarchive.com/show/star-wars-icons-by-artua.html, https://www.iconfinder.com/icons/15483/clone_droid_helmet_star_wars_storm_trooper_icon*

Game: Object structure

- By **extend**ing a class you inherit all properties from it.
- Such extension forms a '**is a**' relationship: "*A Soldier* is a *Galactic Empire Member" etc.*
- Methods can be overwritten in descendents.
- Only single inheritance



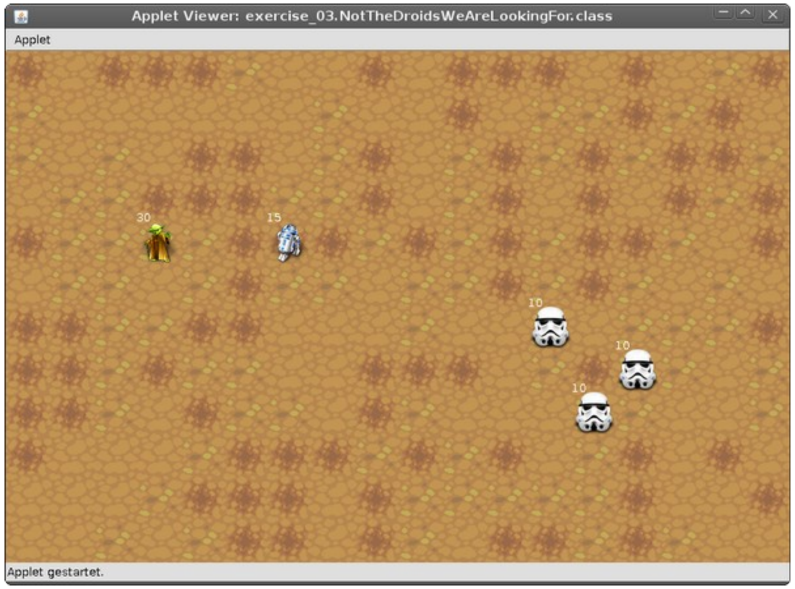
Check if object is of a certain type

The '**instanceof**' operator returns `true` if the argument is in the object hierarchy of the inspected object, `false` otherwise:

```
Soldier soldier = new Soldier(somePosition);
Robot robot = new Robot(anotherPosition);

if (soldier instanceof RebelAllianceMember) {
    // is not executed
}

if (robot instanceof RebelAllianceMember) {
    // is executed
}

if (robot instanceof GameBoardElement) {
    // is executed                                  Keyword
}
```

Hands-On

- Expand the package '**example_08**'.
- Open the file '**NotTheDroids...**'


Applet Viewer: exercise_03.NotTheDroidsWeAreLookingFor.class
Applet
30
15
10
10
10
Applet gestartet.

Tasks

- Run the program.
- Let the game finish properly: Implement '**isGameFinished()**'.
- Add a new game board element '**Rock**' which is just an obstacle. *(Extend '**AbstractGameBoardElement**')*
- Add a new game board element '**Antagonist**' which should be a galactic empire member.
- Replace the random strategy of the empire members with a more elaborate strategy. (*E.g. move towards next enemy.)*

Android: 'Hello World' App

activity_main.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/.../android"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent" >

   <TextView
       android:layout_width="wrap_content"
       android:layout_height="wrap_content"

       android:text="Hello World!" />

</RelativeLayout>
```

MainActivity.java

```
public class MainActivity extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

Some interesting Links

- Get started with **Android** apps
  https://developer.android.com/training/basics/firstapp/
  http://developer.android.com/studio/
- 'Learn Java online' interactive **tutorial** (browser based)
  http://www.learnjavaonline.org/
- Questions and (mostly) **answers**
  http://stackoverflow.com/questions/tagged/java
- An ETHZ education project (in german) to learn Java: **Kara**
  http://swisseduc.ch/informatik/karatojava/kara/
- Popular computer **game** written in Java
  https://minecraft.net/

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Thanks for your attention!

Introduction to the Java programming language
Compicampus - IT Courses for Students

Nico Waldispühl

Appendix

How one would compile the program by hand

Assuming we have the HelloWorld.java file from slide 4 at hand.

Compile with the Java compiler ('javac'):

```
javac HelloWorld.java
```

A class file 'HelloWorld.class' is created. We then call the java **interpreter** with the class name as argument:

```
java HelloWorld
```

*Note that we don't provide the file name, but the class name. Java searches automatically all class files in the so-called **class path** (the set of all paths java searches for classes) for this class.*

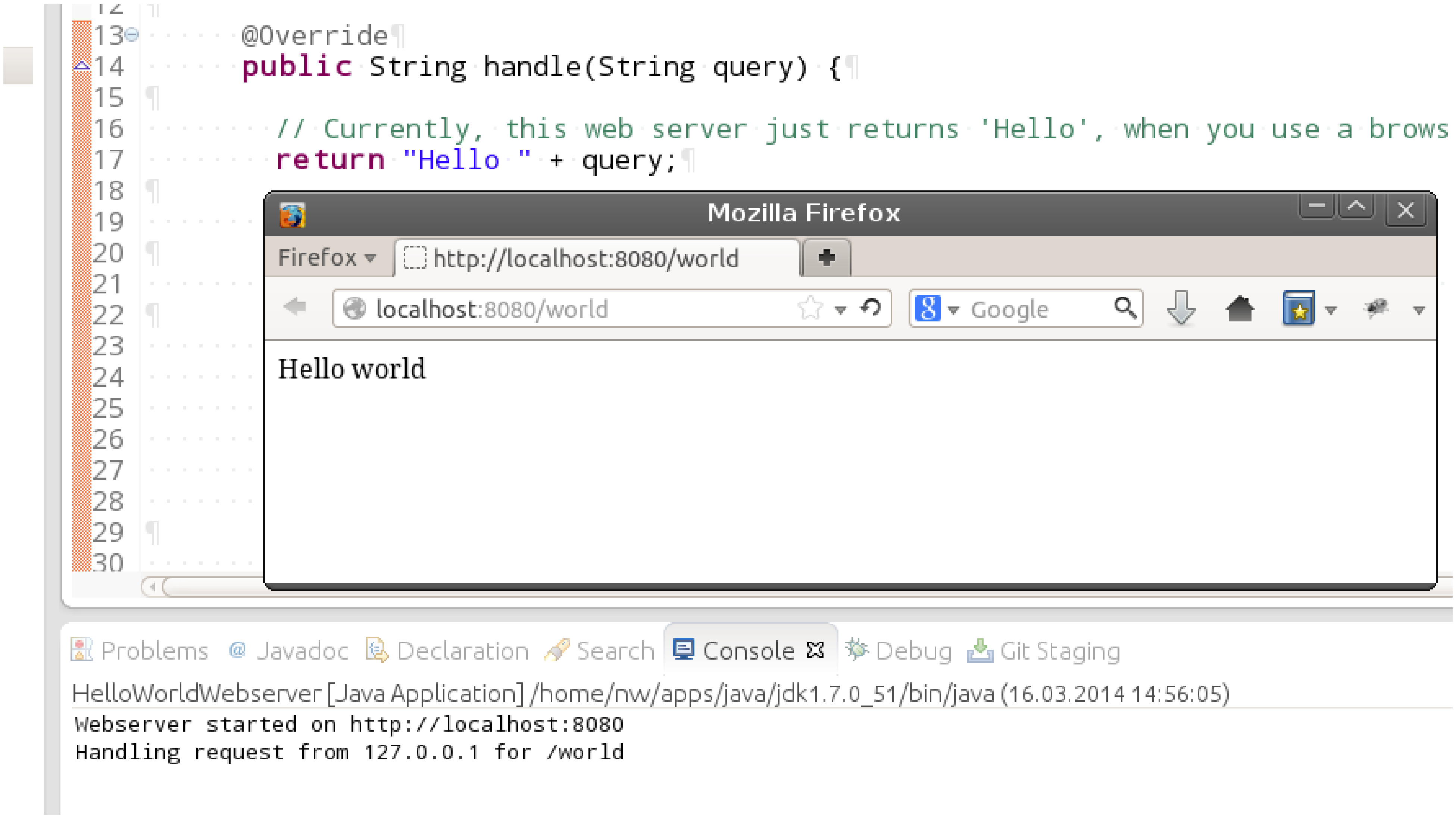
By default, the class path is the current path (and some known places).

How to download and use a library

A library provides functionality not yet contained in the Java environment. It usually comes as .jar file.

- Create new directory (e.g. *libraries*) in your Eclipse project
- Download/obtain library (unzip if needed)
- Place .jar file(s) in new directory, refresh view in Eclipse
- Right-click on library file: '***Build path***' -> '***Add to build path...***'
- Library should then appear under 'Referenced libraries' and can be used in your classes.

My first web server



```
    @Override
    public String handle(String query) {

        // Currently, this web server just returns 'Hello', when you use a brows
        return "Hello " + query;
```

Hello world

```
HelloWorldWebserver [Java Application] /home/nw/apps/java/jdk1.7.0_51/bin/java (16.03.2014 14:56:05)
Webserver started on http://localhost:8080
Handling request from 127.0.0.1 for /world
```

Some interesting snippets

- Tipp: All classes provide a (more or less informative) string representation:

```
SomeClass someObject = new SomeClass();
someObject.toString();
```

- **Current date**

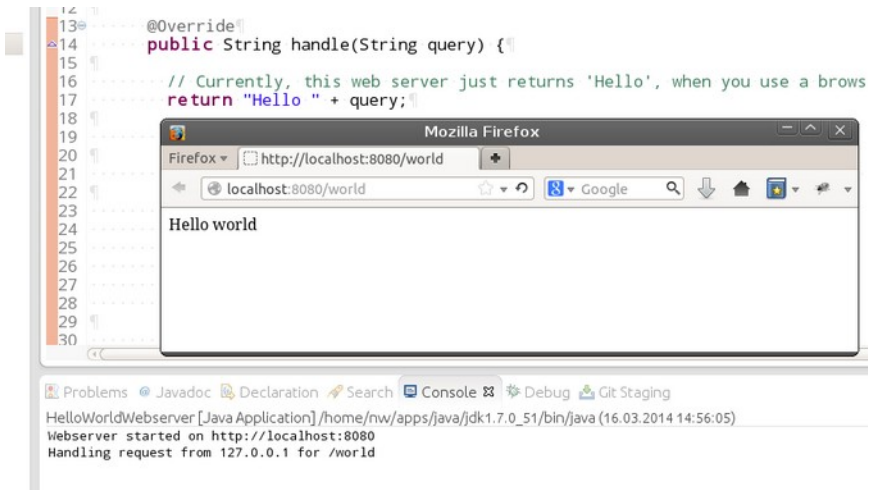
```
Date date = new Date();
```

- **Random number**

```
Random random = new Random();
random.nextInt();  // Provides an integer in [0, 2^31)
random.nextInt(n);    // Provides an integer in [0, n)
```

Hands-On

- Expand the package '**example_09**'
- Open the file '**HelloWorld...**'.



```
    @Override
    public String handle(String query) {

        // Currently, this web server just returns 'Hello', when you use a brows
        return "Hello " + query;
```

Hello world

HelloWorldWebserver [Java Application] /home/nw/apps/java/jdk1.7.0_51/bin/java (16.03.2014 14:56:05)

```
Webserver started on http://localhost:8080
Handling request from 127.0.0.1 for /world
```

Tasks

- Run the program.
- Spice up the web servers output a bit:
  - Return the current time
  - Return a random number
  - ... ?
  - *Can you reach your neighbours webserver by the way? Ask him for his IP address.*
    On the terminal, it can be acquired as follows: `$ ip addr`