Introduction to OO Programming

WITH THE HELP OF THE JAVA LANGUAGE, OBJECT ORIENTED DESIGN

AND MAYBE SOME CONCURRENT PROGRAMMING

COURSE MATERIAL

- You'll find a huge amount of tutorials online (with videos), some good ones, some less…
- Of course also a lot of course material, in French or in English
- We'll use for our course some base courses from:
  - P. Bellot (ENS Telecom)
  - J. Sopena (Université Pierre & Marie Curie)
  - R. Grin (Université Nice)

COURSE MATERIAL

- A lot of OOP online courses are available around Java:
  - EDX, Coursera, France Université Numérique
  - ITunesU, Open Classroom, etc.

- Some videos are sometimes (but rarely) useful on YT
- Let's GO !
  - Anything found about OOP / JAVA outside this course can be useful
  - Especially some examples or exercises
  - But don't run too fast, stay sync with the course

PROGRAMMING ENVIRONMENT

IDE dedicated to JAVA

- GreenFoot or BlueJ are simple and quite easy to learn
- **Eclipse** or **IntelliJ** are harder to manipulate for novices
- For running code, we'll mainly use … the CLI

Some online tools as well

- repl.it
- Codeboard.io
- But nothing better than… the command line !

FIRST SESSION

THE BASICS OF THE JAVA LANGUAGE

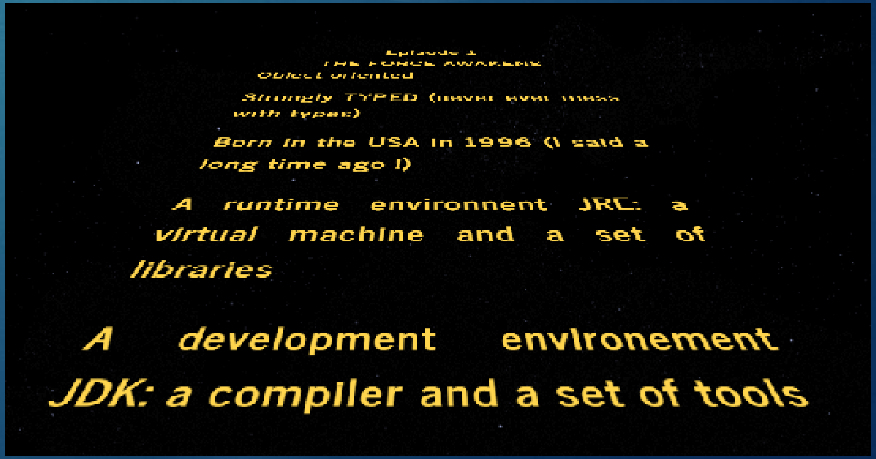
(It’s not OO programming time …..next time)

COURSE OVERVIEW

1. JAVA principles and origin
2. JAVA primitive types
3. Enough, give me the `main`!
4. Control structures
5. Arrays and Strings
6. Parameters (Functions? No, *Methods* !)

1. JAVA PRINCIPLES AND ORIGIN

(A LONG LONG TIME AGO IN A GALAXY FAR, FAR AWAY…)



Episode 1
THE FORCE AWAKENS
Object oriented

Strongly TYPED ([illegible]
with types)

Born in the USA in 1996 (I said a
long time ago !)

A runtime environnent JRE: a
virtual machine and a set of
libraries

A development environement
JDK: a compiler and a set of tools

WHAT'S JAVA

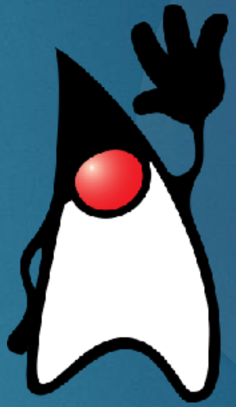
A LANGUAGE !

- Platform agnostic,
- To be available on electronic devices
- Object oriented
- Strongly TYPED (never ever mess with types)
- Born in 1995 (I said a long time ago !)

AN ENVIRONMENT

- A runtime environnent **JRE =** a virtual machine and a set of libraries
- A development environment **JDK**: a compiler, a runtime environment, and a set of tools

A MASCOT



WHAT'S JAVA

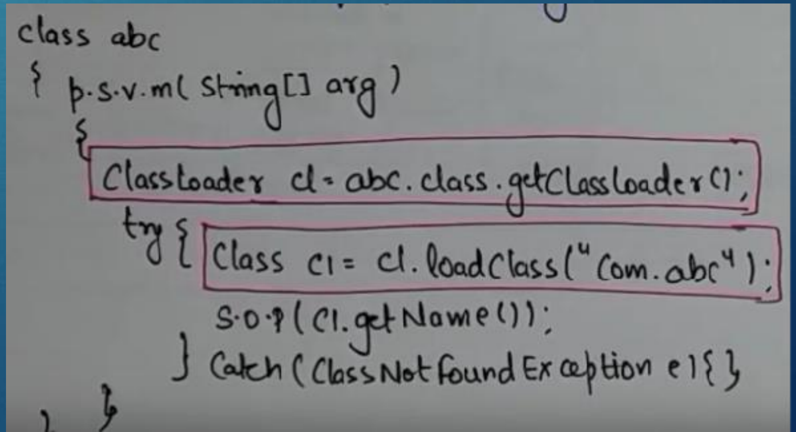
WHO IS BEHIND JAVA?

- Issued from a SUN Microsystems project started in 1990
- The usual suspects as JAVA fathers:
  - James Gosling
  - Patrick Naughton
  - Mike Sheridan
- Currently maintained by *Oracle*

1990 Barbecue chez James Gosling

Why JAVA nowadays?

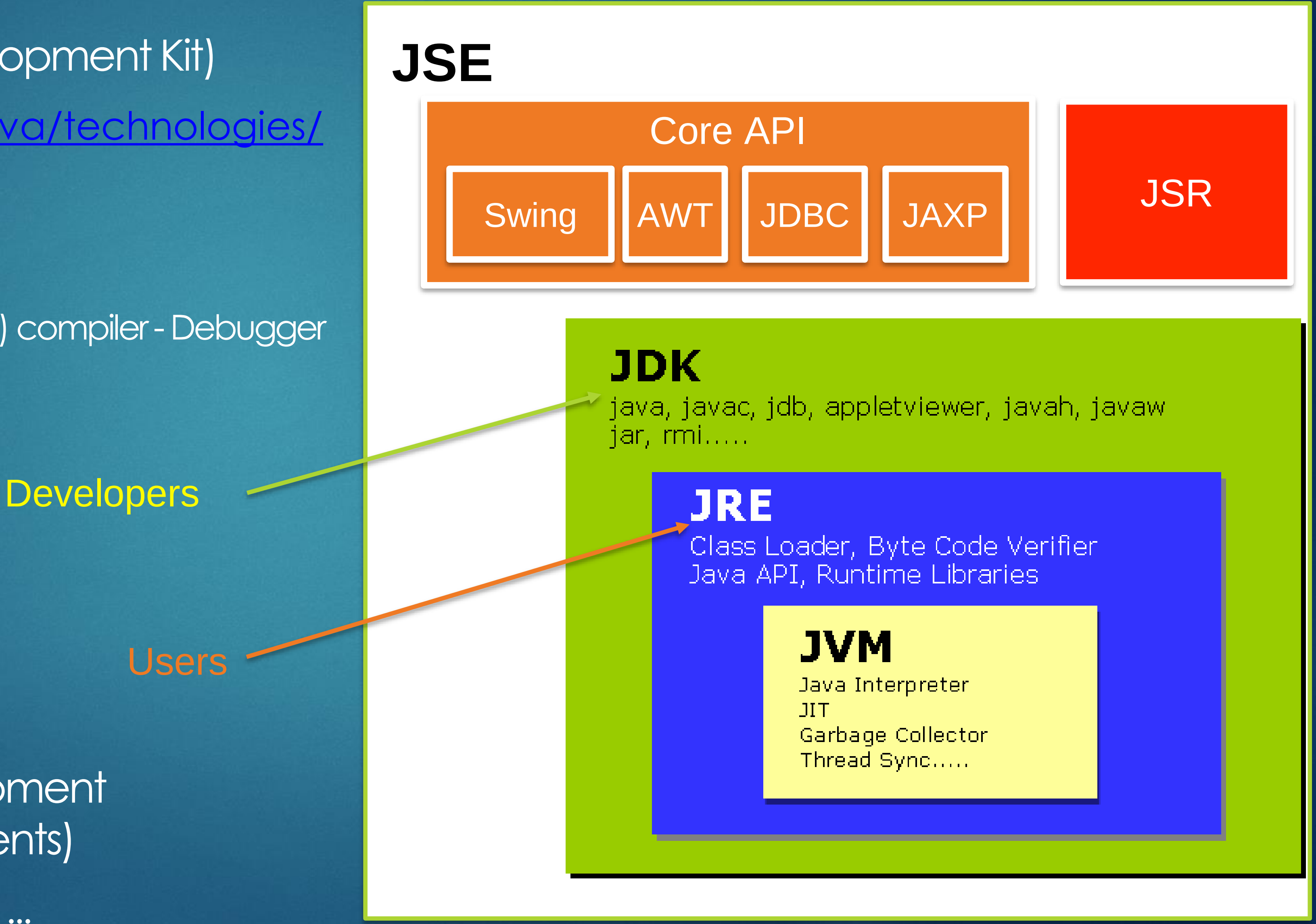
- One of the mostly used programming language (>900 millions computers)
- Available for a very … very large base of computers + various devices and connected objects
- Serves as a basis for standalone apps, mobile apps (Android, Kotlin), web apps for business (Spring boot, …), games (Minecraft)
- Essential in dynamic systems where classes are loaded dynamically (android systems, …)



```
class abc
{ p.s.v.m( String[] arg )
  {
   Classloader cl= abc.class.getClassloader();
     try { Class c1= cl.loadClass("com.abc");
           S.O.P( c1.getName());
     } Catch ( ClassNotFoundException e){}
  }
```

JAVA

- Basic tool: the JDK (Java Development Kit)
  - https://www.oracle.com/java/technologies/
  - Last version: 17 (LTS) + 18
  - Delivered with many tools :
    - Compiler – and JIT (Just in Time) compiler - Debugger
    - Documentation generator



- Some free Integrated Development Environments (at least for students)
  - NetBeans, Eclipse, IntelliJ IDEA, ...

JAVA

- Install the JDK (version 17) :
  - https://www.itzgeek.com/how-tos/linux/how-to-install-oracle-java-jdk-17-on-linux.html
  - https://fr.linuxcapable.com/how-to-install-java-17-lts-jdk-17-on-ubuntu-20-04/
- Install a Java IDE :
  - IntelliJ
  - Eclipse
  - Mode for VS Code

JAVA *versus* C++

- Historically bound:
  - **1983** (AT&T Bell Labs) : C++
  - **1990** (Sun Microsystems) : Java
- JAVA is closer than you may think to C++
- JAVA is SIMPLER, because critical points of C++ have been simply removed from the language.
  - Pointers (Memory management)
  - Operator overriding
  - Multiple inheritance

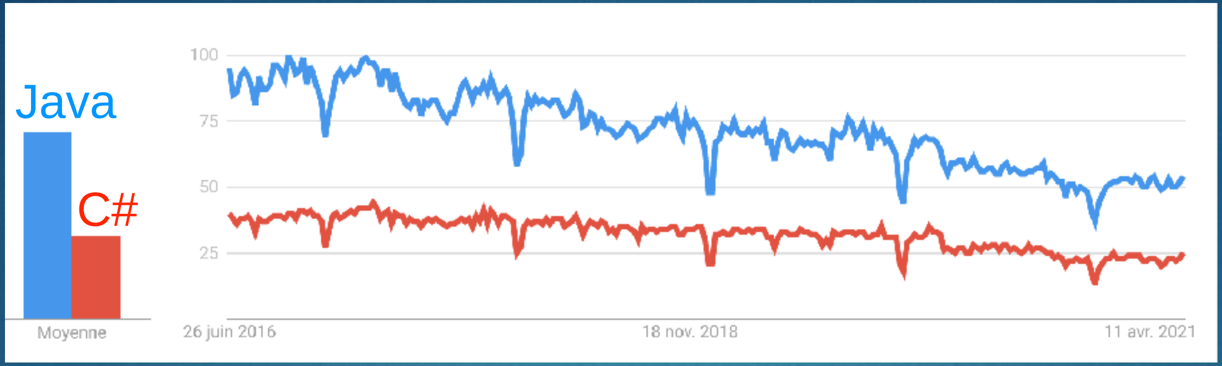
- More
  - Memory management is seamless to users, no need to specify when you don't want to use an object anymore, the GARBAGE collector do the job for you !

- More reliable: no mess with memory allocation / deallocation
- But it costs time to do so …

JAVA *versus* C++

- Binary production:
  - Basically, after development, you can choose to:
    - Give the source
    - Keep it for yourself and just give binaries to run on client machines
- Usually, companies want to keep and protect their sources.
- Binary code has to be portable on whatever machines your client runs (processor, operating system, etc…)

- With C++, your code must be compiled on each machine where it's supposed to run (or fail)…
  - You must then create one binary for each potential client architecture
  - Your code is then optimized, and … runs (or fails) fast



JAVA

- Detailed Documentation is available.
- Allows you to form standard programs and reusable code
- It is a multi-threaded environment which allows you to perform many tasks at the same time in a program.
- Excellent performance
- Huge array of 3rd party libraries
- high memory and processing requirements.

C#

- Language Integrated Query (LINQ)
- Properties with getting/set methods
- C# is less flexible as it mostly depends on the .Net framework
- the server running the application must be windows based
- can handle pointers
- allows operators overloading
- allows goto instruction

Sources: https://www.guru99.com/java-vs-c-sharp-key-difference.html, google trends, …

JAVA *versus* Javascript

(for java-advanced students)

Java is a class-based language

- a new instance is constructed through a class's *constructor* (that reserves a block of memory for the object's members and returns a reference to that block)
- The resulting instance will inherit all the methods and properties that were defined in the class, a kind of template)

1) a "Fruit" class is defined
2) a "Banana" class extending "Fruit" is defined
3) bananas objects are created by creating objects of the *Banana* class considered as a template

Distinction between *classes* and *objects*

Javascript is a prototype-based language

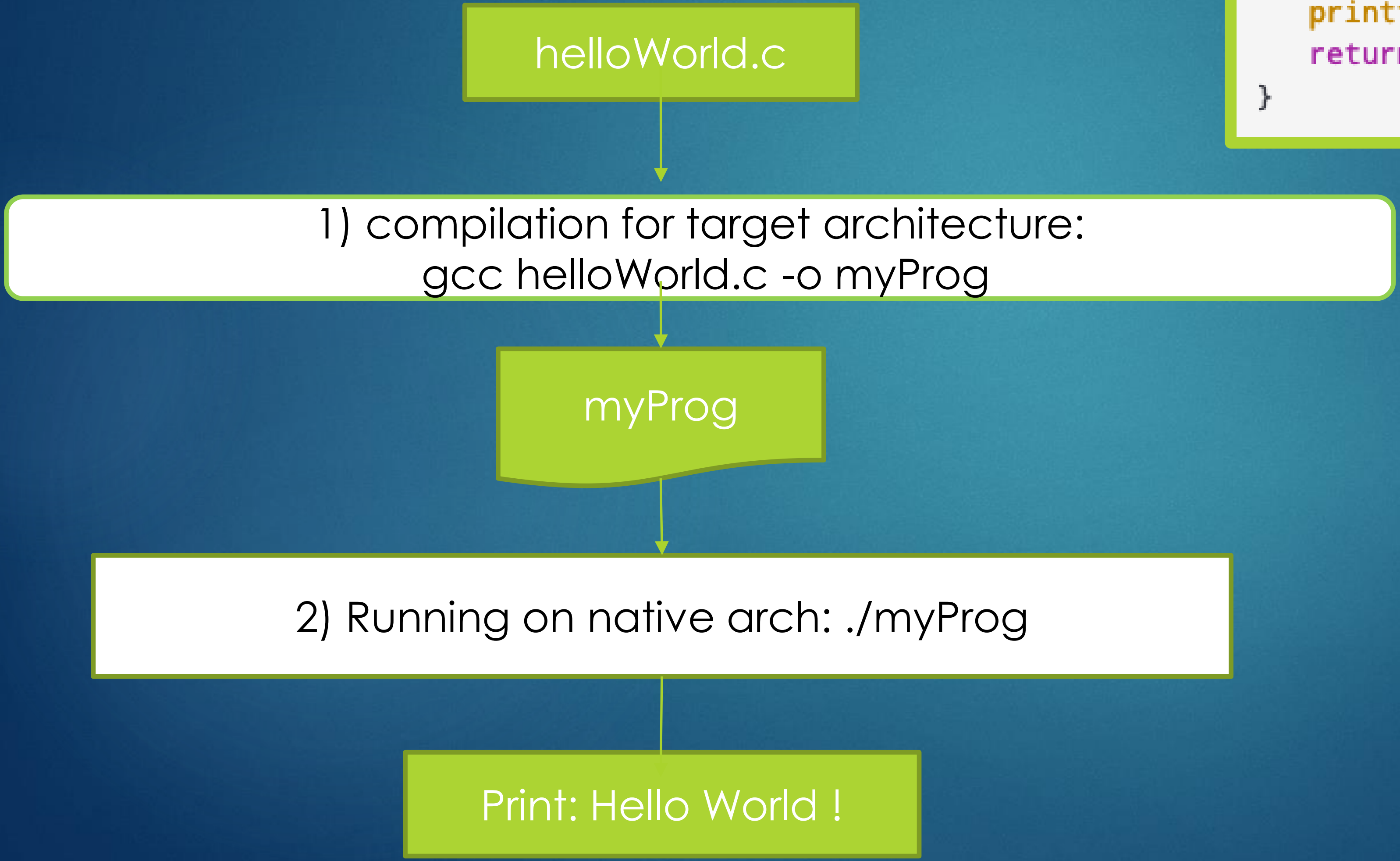
- behavior reuse (known as *inheritance*) is performed via a process of reusing existing objects that serve as prototypes
- uses generalized objects, which can then be cloned and extended

1) a "*fruit*" object represents properties and functionality of fruit in general.
2) a "*banana*" object is then cloned from the "*fruit*" object
3) general properties specific to bananas are appended to *banana*
4) each individual banana object is obtained as a clone of the generic "*banana*" object.

Everything is an *object*

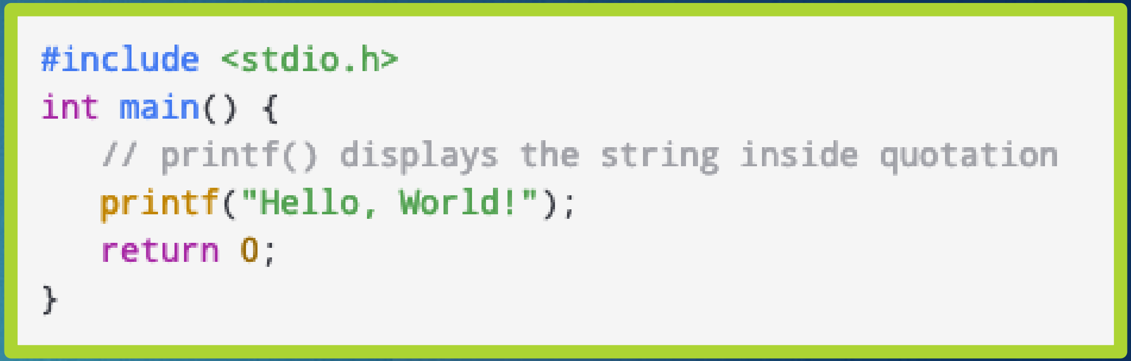
source: Wikipedia, https://developer.mozilla.org/fr/docs/Web/JavaScript/Guide/Details_of_the_Object_Model

Production scheme in C/C++


helloWorld.c
1) compilation for target architecture:
myProg
2) Running on native arch:
Print: Hello World !

```
gcc helloWorld.c -o myProg
```

```
./myProg
```



```c
#include <stdio.h>
int main() {
   // printf() displays the string inside quotation
   printf("Hello, World!");
   return 0;
}
```

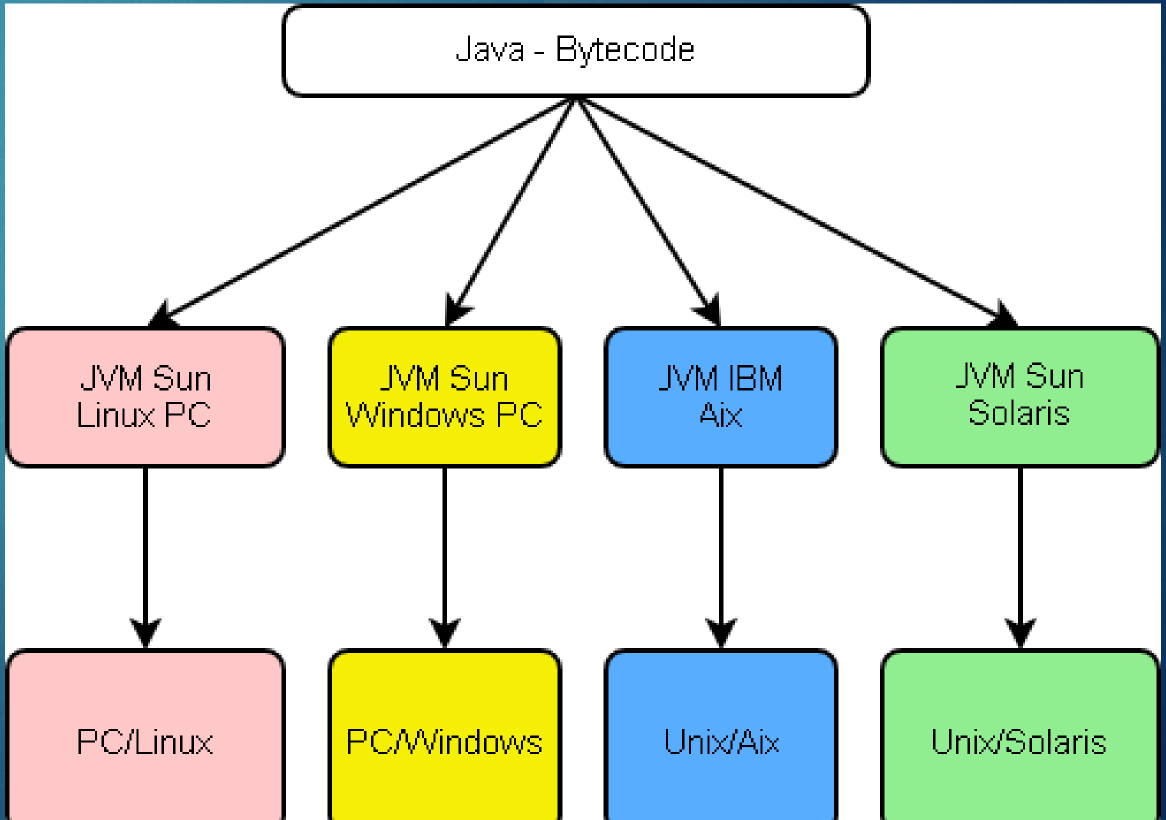
JAVA *versus* C, C++, C#

- In JAVA, code is NOT directly executed in the native computer architecture.
- It is first translated in some intermediate thing called « **bytecode** », a specific language **interpreted** by the **JVM** (Java Virtual Machine)

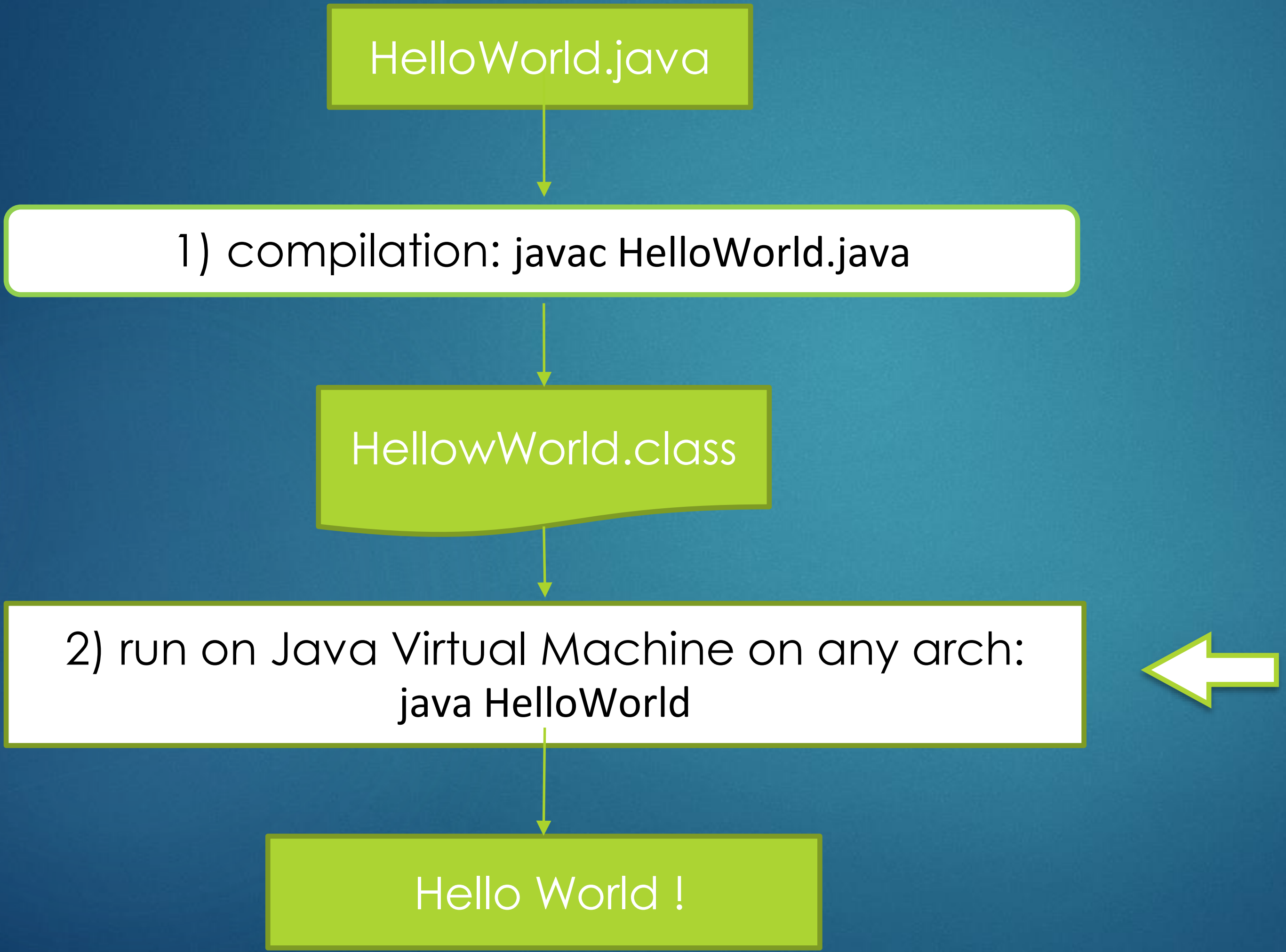
PORTABILITY:

Bytecode will run (almost) everywhere, through a virtual machine for the target architecture

But still, you need to have a compatible version of the virtual machine installed on the destination computer…



JAVA production scheme


HelloWorld.java
1) compilation: javac HelloWorld.java
HellowWorld.class
2) run on Java Virtual Machine on any arch:
java HelloWorld
Hello World !

no file extension must be indicated

Let’s try a first example

- Create a text file with your favorite editor: ???.java

```
public class HelloWorld {
    public static void main(String... args) {
        System.out.println("Hello Java World");
    }
}
```

1) Compile it in a terminal: javac HelloWorld.java
2) Check what the compiler created
3) Is it a text file?
4) Run the JVM to execute your program: java HelloWorld

Advanced students: solve the Mystery.java enigma (available on Moodle at https://moodle.umontpellier.fr/mod/resource/view.php?id=557563)

JAVA executing bytecode

- The bytecode must be executed on a Java Virtual Machine.

- But this JVM doesn’t exist. It’s simulated by a program which, **at runtime**:
  1. Reads bytecode instructions from .class files
  2. verifies code in a first pass (operand types, stack size, data flow, variable initialization…) to be sure that nothing may be dangerous
  3. optimizes the code (re-passes and again)
  4. Translates all this in the destination processor native language
  5. And finally… executes the code

Summary: JAVA characteristics

- Java has a « C-Like » syntax (approximately)
- Object Oriented: Almost everything is an Object, except simple types : int, float, double, booleans, …
- Robust : Strongly typed, no pointers (~ hidden to developers)
- The compiler only produces intermediate code (« bytecode »)

In very few words:
- JAVA lost the game in term of rapidity / efficiency (this is less true with latest versions of JAVA)
- JAVA wins (by far) in code portability !

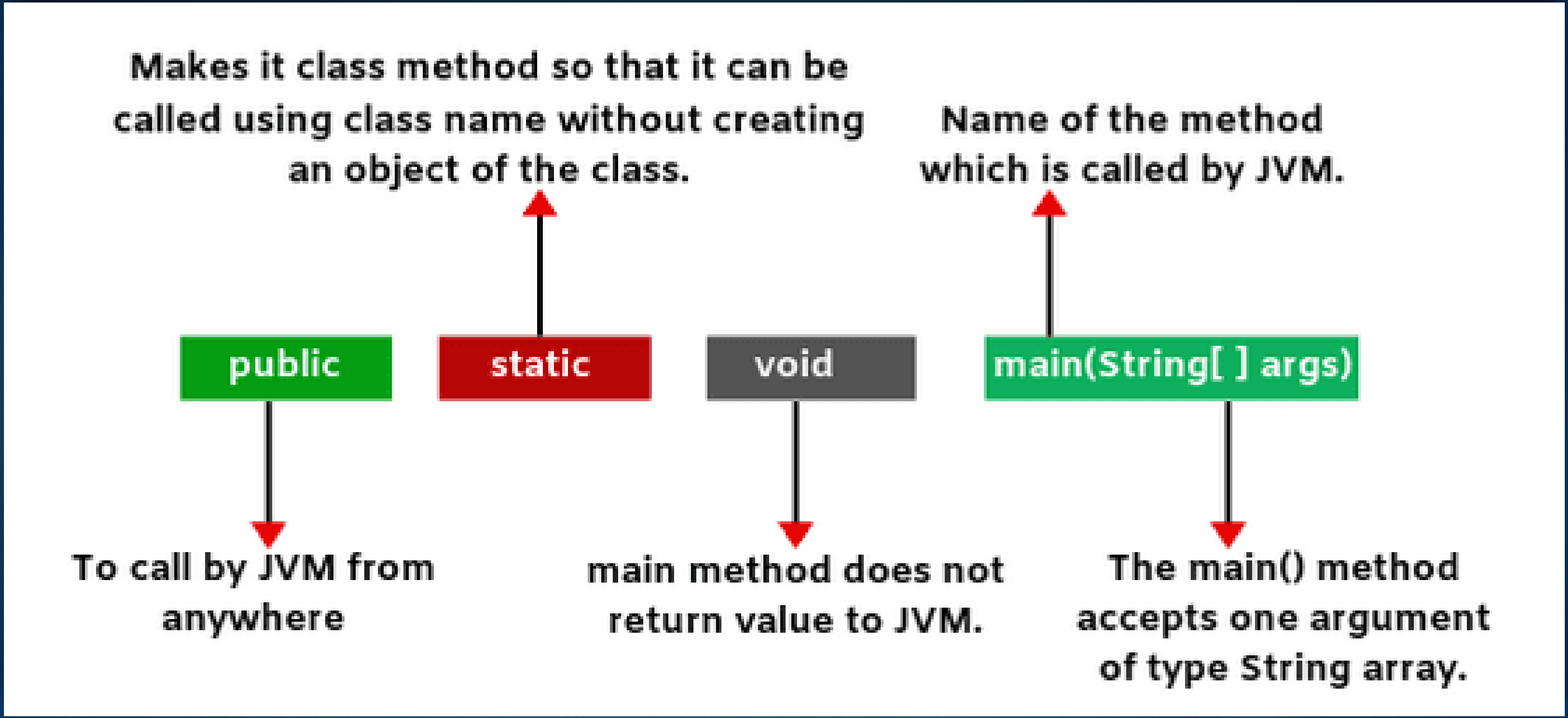
new JAVAs

- Java 16 (born in 2021):
  - records
  - sealed classes and interfaces
  - packaging utility
  - …
- Java 17 LTS is out since last September!
  - Applets deprecated, LTS, …
- Java 18 is the current release …

(for java-advanced students)

Provide others with a brief summary of each of these features on Mattermost

2. Give me the main


Makes it class method so that it can be called using class name without creating an object of the class.
Name of the method which is called by JVM.
public
static
void
main(String[ ] args)
To call by JVM from anywhere
main method does not return value to JVM.
The main() method accepts one argument of type String array.

source: https://www.scientecheasy.com

My First source code « another Hello World »

```
public class HelloHAL {
 public static void main(String[] args){
  System.out.println("Can you tell me why we are in this
                            film, Bob?");
 }
}
```

- HelloHAL is declared '**public**', then the file containing this class **MUST be named « HelloHAL.java »**, and no ignoreCase around!
- The main method is the entry point of your program
  - It's given params corresponding to args indicated in the terminal when launching the program
  - it will run only when this class is the entry-point in the program (a code can be reused as a module in a larger program)

Lab on plain old java

- Sarkozy, Macron and Hollande have quite a fortune to rule, say each own 50 M€.

1. Seeing Sarkozy's problems with justice, Macron decides to give him half of his fortune.
2. Seeing this, Hollandes decides he should participate but he can't give money straight to Sarkozy, so he gives half of his fortune to Macron
3. Sarkozy's being quite sure to be soon out of trial then decides to give half of his fortune to Hollande

Turn that into a Java program with proper variable declaration and assignment instructions, then print the fortune of each of these fictional characters in the end.

3. JAVA primitive data types

Some non-object but…

Primitive types / defaults

- **boolean** (1 byte) / false
- Whole Numbers:
  - **byte** (1… byte) / 0
  - **short** (2 bytes) / 0
  - **int** (4 bytes) / 0
  - **long** (8 bytes) / 0L
- Other numbers:
  - **float** (4 bytes) / 0.0F
  - **double** (8 bytes) / 0.0D
- Single Character
  - **char** (2 bytes) in Unicode

But they have « Wrappers »

- Boolean
- Numbers:
  - Byte
  - Short
  - Integer
  - Long
- Numbers again:
  - Float
  - Double
- Character:
  - Character

CASTING

Java is « STRONGLY » typed…

Sometimes, you NEED to force the program to consider an expression in some type which is not its… type. CAST is there to help, but it's restricted:

- ★ Between primitive types
- ★ Between parent / ancestors and child classes (related to Inheritance concept)

```
int x = 10, y = 3;
double z = x /y;
// both x and y are int, so z = 3
// what if we want 3.3333333 and not 3?
double z = (double) x   /   y  ;
// Casting only x is enough
```

Try it yourself with JShell[1] !!!

[1] https://docs.oracle.com/en/java/javase/18/jshell/introduction-jshell.html

CASTING PROBLEMS

- If an assignment can result in value loss, it MUST contain an explicit cast (you'll be responsible for side effects)
- Omitting this cast might not impede compilation
- BUT ! It can result in totally rogue results, without ANY WARNING from compiler !!!
- Declare a float variable and assign it 3.14
- What happens and why?

MIND THE GAP

```
int i = 1000;
short s = (short) i;
//print

// SIDE EFFECTS
int j = 130;
byte b = (byte) j ;
//print

int c = (int)1e+9;
//print

c = (int)1e+10;
//print

c = Integer.MAX_VALUE;

var d = 100;
// What is var???
```

Test it by yourself (e.g. on JShell or repl.it)

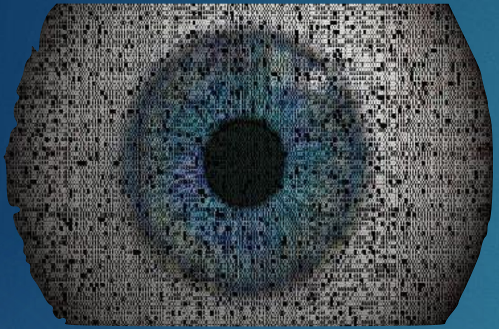
Casting between `ints` and `chars`

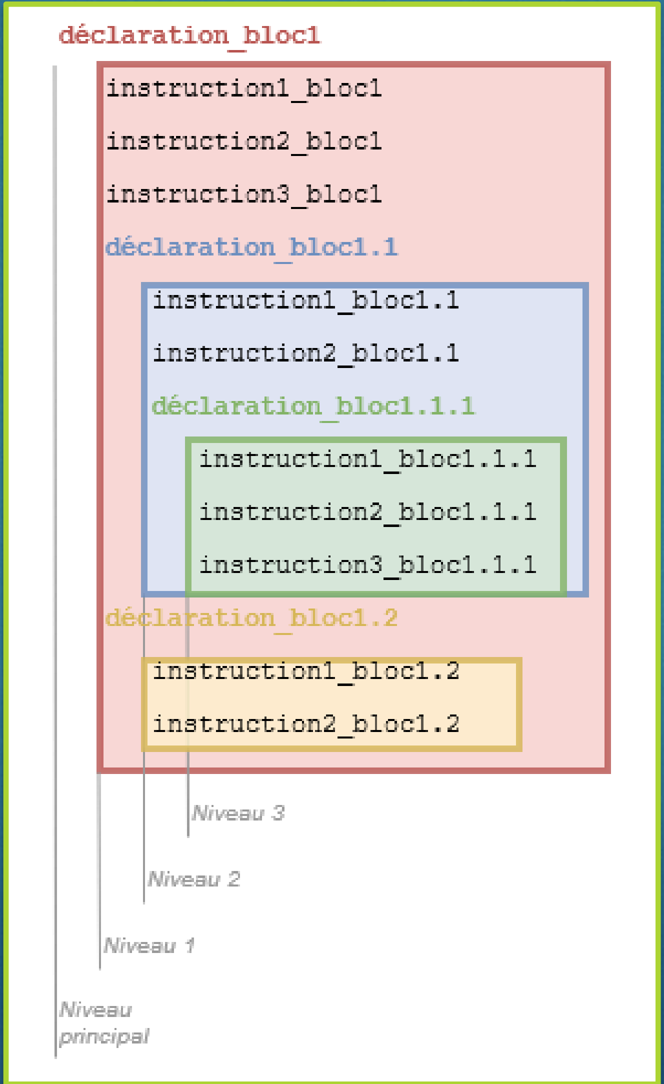
- You can map an int to a char:
  - such a casting results in the Unicode value for this integer.
- Correspondence **char → int, long** is **implicit**

A char (2 bytes) can encode 0 to 65 535 integers so
char → short, byte needs an **explicit** cast – a short can only reach 32 767

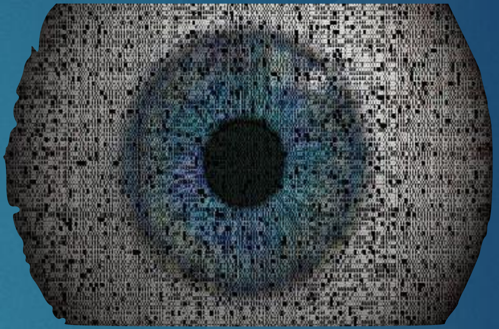
Moreover, integers are signed, but chars are not, so
long, int, short or byte → char need an explicit cast

4. Control Structures





```
déclaration_bloc1
    instruction1_bloc1
    instruction2_bloc1
    instruction3_bloc1
    déclaration_bloc1.1
        instruction1_bloc1.1
        instruction2_bloc1.1
        déclaration_bloc1.1.1
            instruction1_bloc1.1.1
            instruction2_bloc1.1.1
            instruction3_bloc1.1.1
    déclaration_bloc1.2
        instruction1_bloc1.2
        instruction2_bloc1.2
```



Alternatives:
« if » or « if … else »

```
if (booleanTestExpression)
        block or single statement
[ else
        block or single statement ]
```

Optional block

Alternatives « if » or « if … else »

```
int  x = y + 5;
if (x % 2 == 0) {
        bla = 0;
        x++;
} else {
        bla = 1;
}
```

Switching when too many ifs!

```
switch (expression) {
 case val1:
          ...statements;
          break;
 case val2:
          ... statements;
          break;
 default:
          ... other statements
}
```

Expression is either **char**, **byte**, **short**, **int**, **Enumeration**, or **String**

- No need for { } even for multiple statements in « case ».
- Warning : if no « **break** », the following statements are executed.

Optional block

PAUSE ?

Lab on alternatives

- SNCF wants to accommodate ticket fares to adapt the COVID period:
  - child under 10 pay 50% of normal price
  - elder persons above 65 have a 33% reduction
  - "students" (18 <= age <= 25) with no subscription card can win 15€ for each ticket above 20€
  - students can also buy a 10€ subscription card and have 75% price reduction on any ticket
- Implement a `reduction(…)` method in an SNCF class that given a user's age, the original price of a ticket and whether he already has a subscription card (meaningful only for students) gives the price this user will pay. Make it beneficial for the client, meaning for students it outputs the best price (be it with or without subscription).

details

(for java-advanced students)

At what price threshold is it worth it for a student who does not have a subscription card to buy one, even for a single trip?

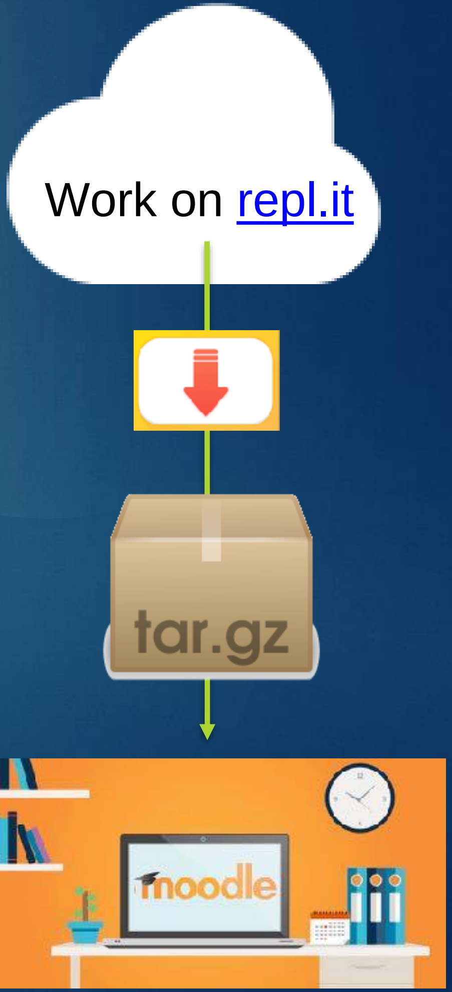
Lab on alternatives

(for students new to Java)

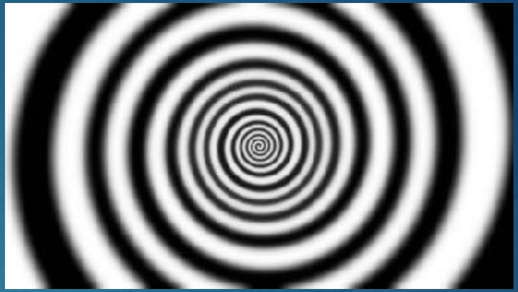
- You can make all values fixed in the main, not caring about the input.

(for java-advanced students)

- Ask the values to the user from standard input:
- Ask the values to the user from args given when launching the program



Looping
« While loops »



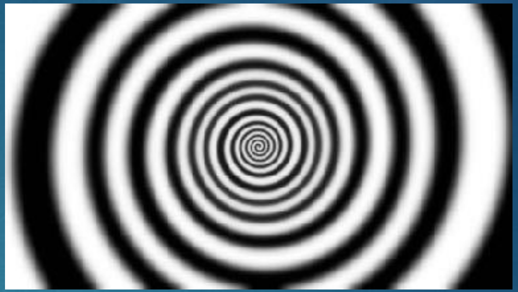
while (*booleanContinuationExpression*)
block or single statement

---

do
block or single statement
while (*booleanContinuationExpression*) *;*

Statements are executed at least once !!!

Looping « For loops »



for (*init; stopTest; increment*)

statements;

Equivalent to:

init;

while (*stopTest*) {

statements;

increment;

}

"Foreach" Statement (works with arrays):

for (*Type variable: tab*)

statements;

Example:

```
double[] scores = … ;
for (double val: scores) {
        System.out.println(val);
}
```

Variable statement

Local, and stay local

You can declare a variable anywhere in a { … } block

- With control structure « `for` » you may declare, and initialize your control variable in one single statement:

```
for(int i=0; i<=10; i++) {
    … // whatever
}
```

Verbauten !

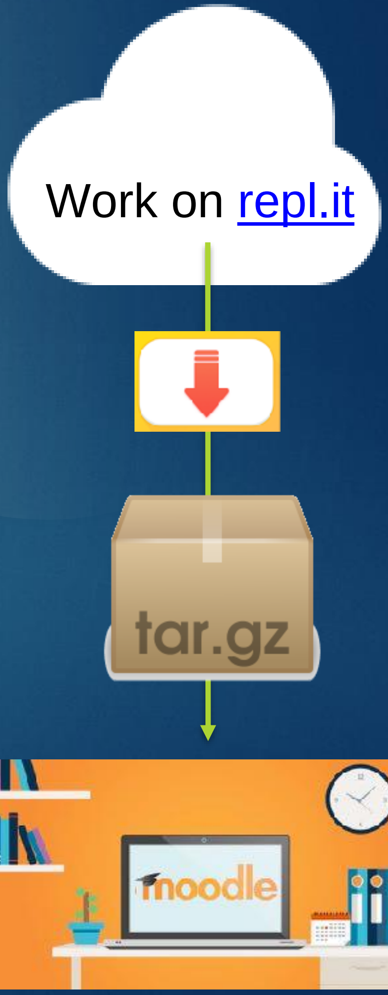
MIND THE GAP

- You can’t use the same name from any surrounding block

```
int somme(int init) {
        int i = init;
        int j = 0;
        for (int i=0; i<10; i++) {
                j += i;
        }
        int init = 3;
}
```

Lab on loops

- Implement the famous Chi-Fo-Mi game where a user chooses at each turn either leaf / stone / scissor, then the computer chooses randomly one of these alternatives. Then the program decides who wins the current turn.
- The game continues until a player has 2 points above the other player, and the winner is then declared
- Is a while or a for loop preferable?

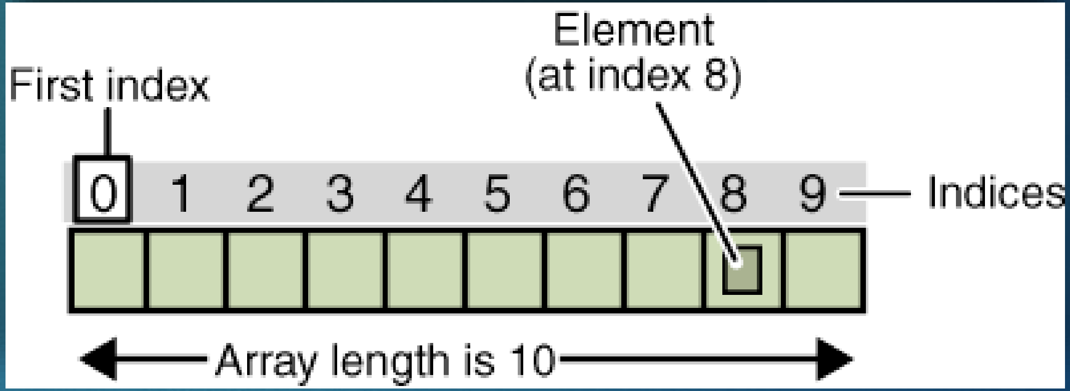


(for java-advanced students)

- Use emoticons to indicate the choice of each player at each turn (see http://dplatz.de/blog/2019/emojis-for-java-commandline.html)

5. Strings and arrays

OBJECTS, FINALLY!



String(s) are objects

- Objects from one of the following classes: String, StringBuilder and StringBuffer
- But THEY ARE **NOT**:
  - A primitive type
  - Arrays
- **Strings** are **constant**: their value cannot be modified after creation (unlike StringBuffer objects)
- A String represents a string in the UTF-16 format

String creation

A string may be created like :

myVar = "Hello World";

a string *literal* is a sqeuence of chars enclosed in " signs

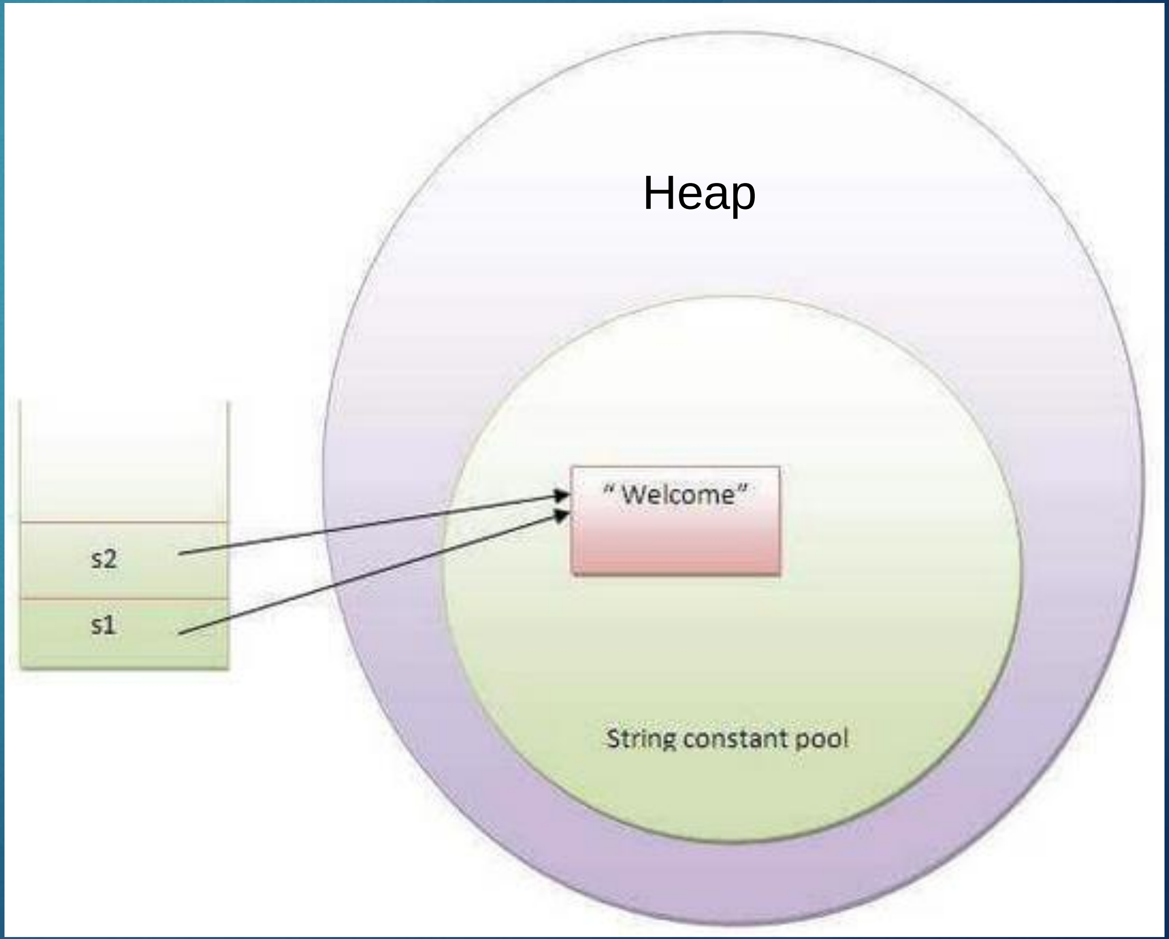
Note:

```
myFirst = "Welcome";
mySecond = "Welcome";
```

Creates only ONE String object in memory (with two references to it)

What about:

```
myFirst = "Hello World";
mySecond = myFirst;
```



String creation

A string may also be created like :

String s= new String("Welcome");

creates two objects and one reference variable

In such case, the JVM will create a new string object in normal (non-pool) heap memory, and the literal "Welcome" will be placed in the string constant pool.
The variable s will refer to the object in a heap (non-pool).

String comparison

- You may use the **`equals(…)`** method :

```
s1 = "Joker is stronger";
s2 = " than Batman";
if ((s1 + s2).equals("Joker is stronger than Batman")){
      System.out.println("Gotham is lost"); }
```

- « == » doesn't actually compare strings, but object references, you should not use it to compare String objects.

  - *Use == to test whether two "Hello world" really reference only one same String in memory.*
  - *what about myFirst.intern()== mySecond.intern() ?*

String creation

(for java-advanced students)

What about:

`s.intern()`

```
String str = new String("Welcome to JavaTpoint.");
String str1 = new String("Welcome to JavaTpoint");
System.out.println(str1 == str);
```

```
String str = new String("Welcome to JavaTpoint").intern();
String str1 = new String("Welcome to JavaTpoint").intern();
System.out.println(str1 == str);
```

```
public class InternExample{
  public static void main(String args[]){
    String s1=new String("hello");
    String s2="hello";
    String s3=s1.intern();//returns string from pool, now it will be same as s2
    System.out.println(s1==s2); // ?
    System.out.println(s2==s3); // ?
  }}
```

String concat:

You may use the '+' operator between String objects :

```
myVar = "Hello World";
myVar = myVar + " !!! ";
```

Note:
If one operand is a String, all the others are « automatically » transtyped to string.

```
int x = 5;
s = "Value of x:" + x; //
```

Cool for printing primitive types

String extraction

- `charAt(int i)`: extracts the i-th char (primitive type) from a String object instance.

    Index starts at **0**

- Go to the documentation of String and list the 3 most useful methods for manipulating strings: https://docs.oracle.com/en/java/javase/16/docs/api/java.base/java/lang/String.html

- Extracting several chars in a raw:

```
String c = "abc".substring(2, 3);
```

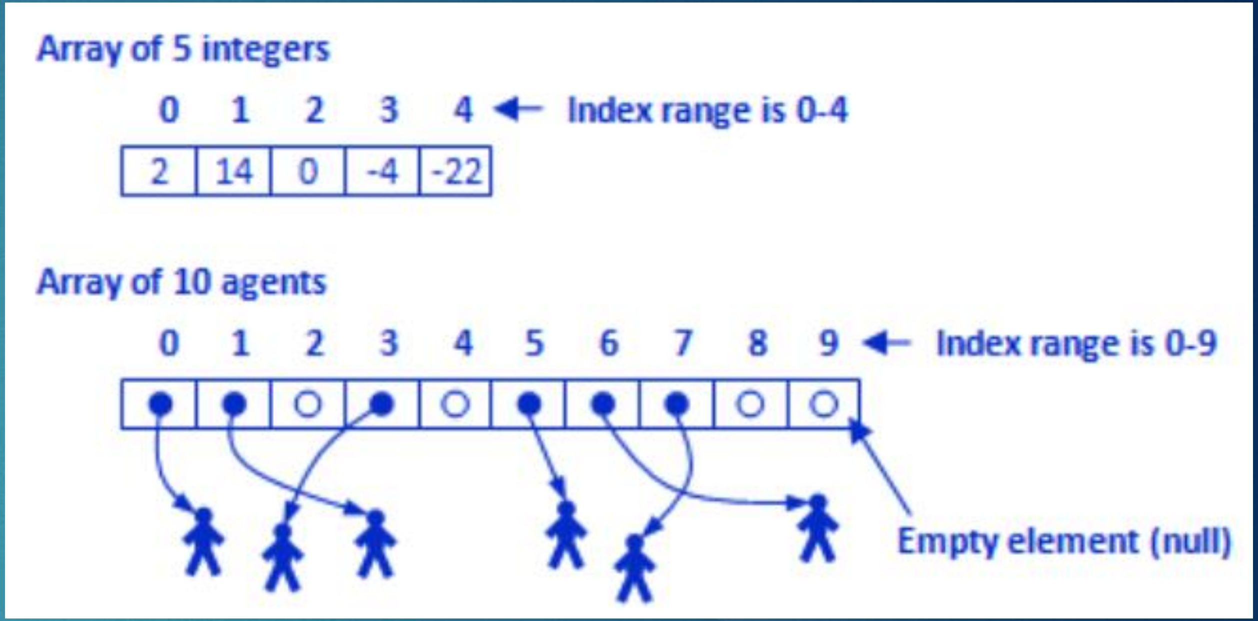
String lab

- Follow the first tutorial called "*Onboarding*" consisting in killing space invaders in Java! :
  https://www.codingame.com/training/easy/onboarding

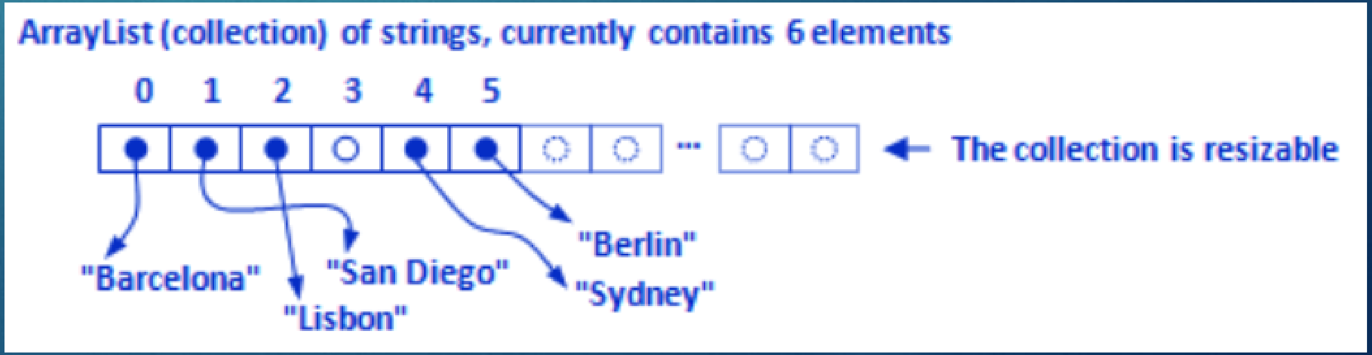
- Go to the « Chuck Norris » puzzle, read carefully the problem to solve. It's not the funniest game around, but it'll help you to better understand some basic issues from this first course:
  - Take a paper & pen, and forget about your keyboard for a time
  - Tip 1: To solve the problem, try to divide the main problem in smaller ones
  - Tip 2: Use and abuse of `System.err.println` to debug your code
  - Tip 3: Considering the binary code for each char may be a good idea:
    - `Integer.toBinaryString((byte)MESSAGE.charAt(i));` // the String corresponding to the ASCII code of the ith character in MESSAGE (read that twice 😅).
  - Tip 4: Test your code with different input before you submit your solution

Array(s) are objects

- Arrays are objects:
  - Arrays can be created by the ‘`new`’ keyword
  - They have a field (an object’s variable) called `length`
- Arrays have a fixed size!!!



- Collections to be seen later:



ArrayList (collection) of strings, currently contains 6 elements
0 1 2 3 4 5
The collection is resizable
"Barcelona"
"Lisbon"
"San Diego"
"Berlin"
"Sydney"

Array(s) are special

- Created with the *new* keyword:

```
int[] myIntTab = new int[10];
```

- Arrays can be declared without the ‘new’ operator, Java has a special syntax (*literal* array):

```
int[] myIntTab = { 125, 2*45, 3, 5, 42 };
```

Using Array(s) object

- Assigning elements: first index is 0 and ends at '.length -1'

```
    int s = scores.length;
double lastScore = scores[s];
// will raise an Exception (~ managed error)
```

- You can declare an Array as a method parameter

```
    double[] average(int[] values) {
        ... // calculate the average of values
}
```

Useful Array(s)

```
class Test {
        public static void main(String[] args) {
         for (int i=0, i < args.length; i++)
           System.out.println(args[i]);
        }
}
```

- java Test banshee Wolverine

back to while

Comparing Arrays

- We can compare each array value one by one, to check if they contain the same values.

- **Warning:**

  `==` will only compare array references (internal pointers)

`equals(Object[] a, Object[] a2)` method returns `true` if the two specified arrays of objects are equal to one another:

- The two arrays are considered equal if both arrays contain the same number of elements,
- and all corresponding pairs of elements in the two arrays are `equal`.

Multidimensional Arrays

- Declaration:

```
double[][] scores;
```

- Each Array element contains a reference to another Array
- Creation and assignment :

```
scores = new double[10][3];
ages = new int[10][];
```

Each cell within the first-level array references another array of at most 3 cells

First dimension is mandatory !

Unknown number of cells !

```
ages[0][0] = 19;
```

Multidimensional Arrays

- Declaring and initializing at the same time:

```
int[][] scores = {
            {10, 11, 9}, // 3 scores
            {15, 8} // 2 scores . .
};
```

Assignment at creation time

- Direct assignement:

```
scores[10][2] = 6;
scores[5] = { 0, 16 };
```

Assignment of a whole subarray

Printing out content:

```
for (int i = 0; i < scores.length; i++) {
 for (int j = 0; j < scores[i].length; j++) {
   System.out.print(scores[i][j] + "; ");
 }
 System.out.println();
}
```

Array lab

- Create your account on the codeboard.io platform: firstName+lastName, setup that you are from « Polytech Montpellier »
- Go to the Labyrinth1 activity: https://codeboard.io/projects/13637
- Creating a more complex labyrinth: https://codeboard.io/projects/262929
- Exiting from a lab: https://codeboard.io/projects/262928