Programming and Java

introduction – variables, expressions

P. Bakowski

bako@ieee.org

Programming and Java

Our educational objectives are:

1. Study the basic concepts of programming in high-level languages and apply them in Java

2. Learn the analysis and resolution of problems through programming,

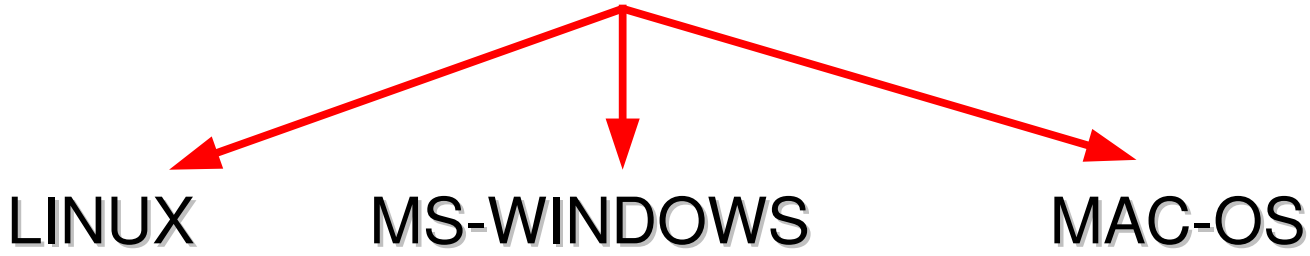
3. Acquire certain methods of resolution of the classic problems in computer science

Why Java ?

1. Object-oriented, exceptions, polymorphism, memory management, transparency of pointers ..

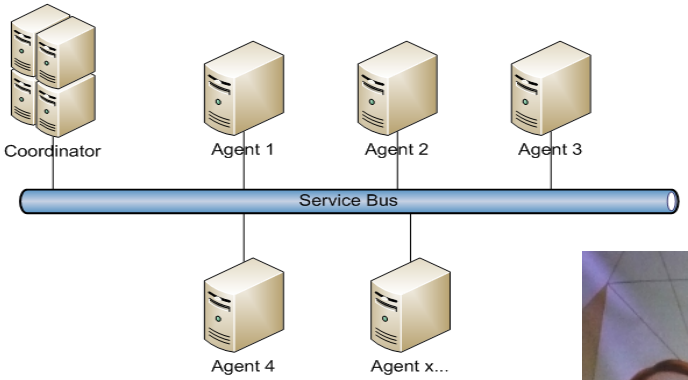
2. Strongly typed

3. Programs are portable - platform independent architecture

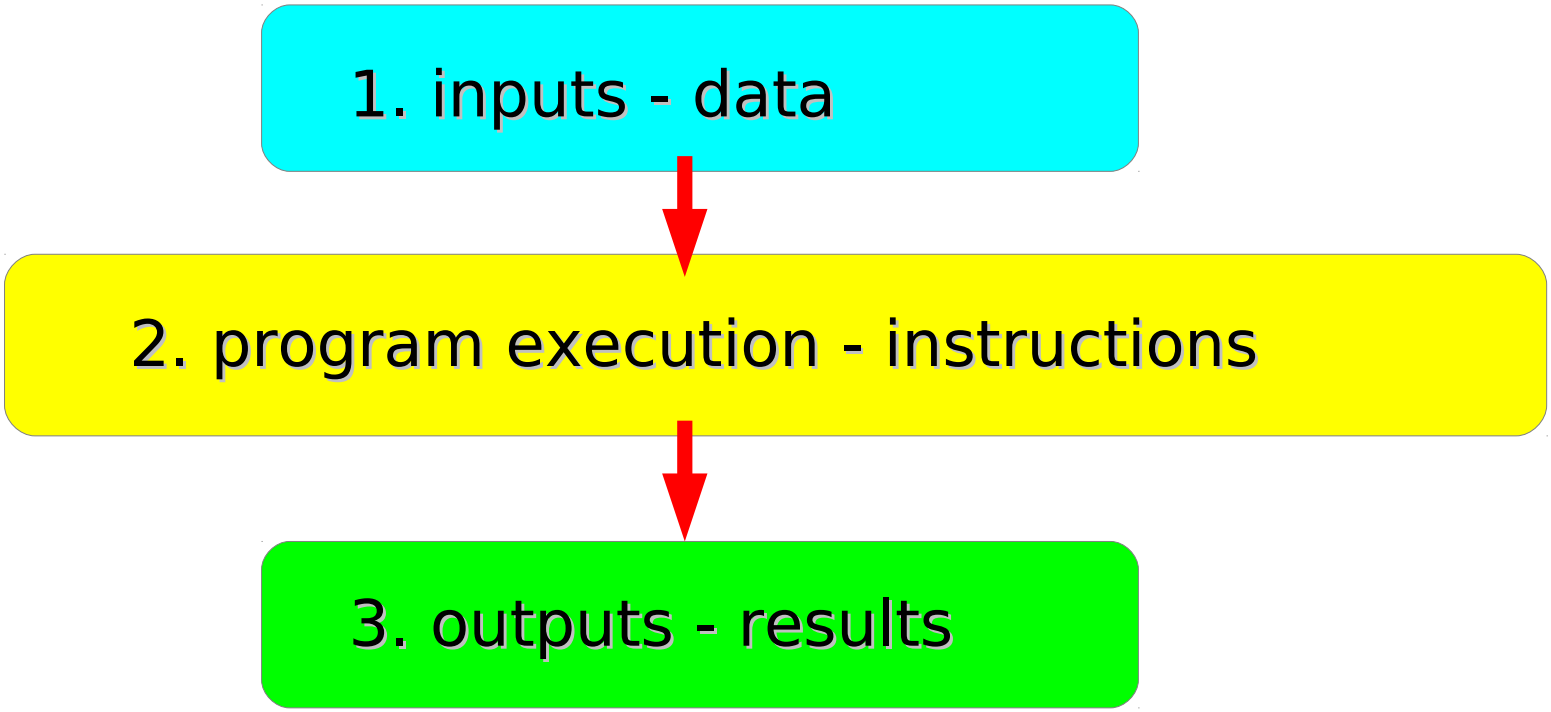


Java and application domains

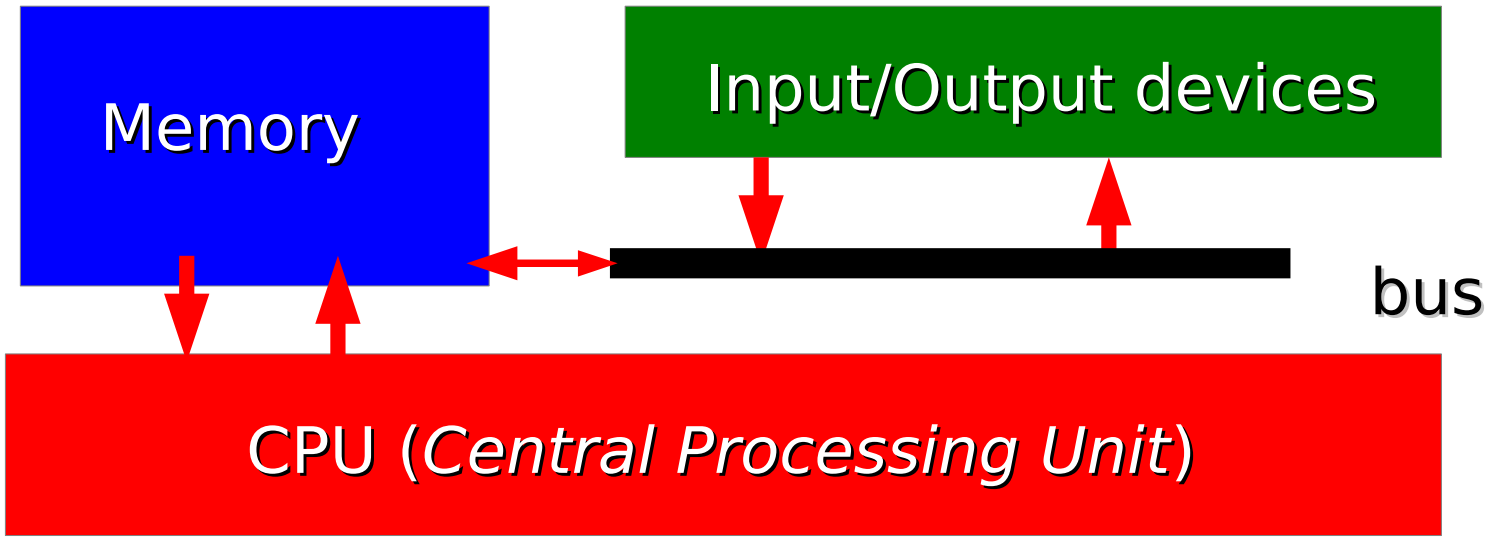
1. Internet and the Web
2. Distributed programming
3. Embedded programming



Structure of program



The Computers



The programming languages

There are two main groups:

- High level languages.
Examples: C, C + +, Java, Python,..

- Low-level languages.
assemblers:
machine code: ARM, Intel / PC, etc.

High Level Languages

- the Types of data:

```
int, char, double, ..
```

- the Syntax:

```
1 <= x && x <= 7
```

- the Semantics:

```
"hello" * 2
```

(problem?)

The production of programs

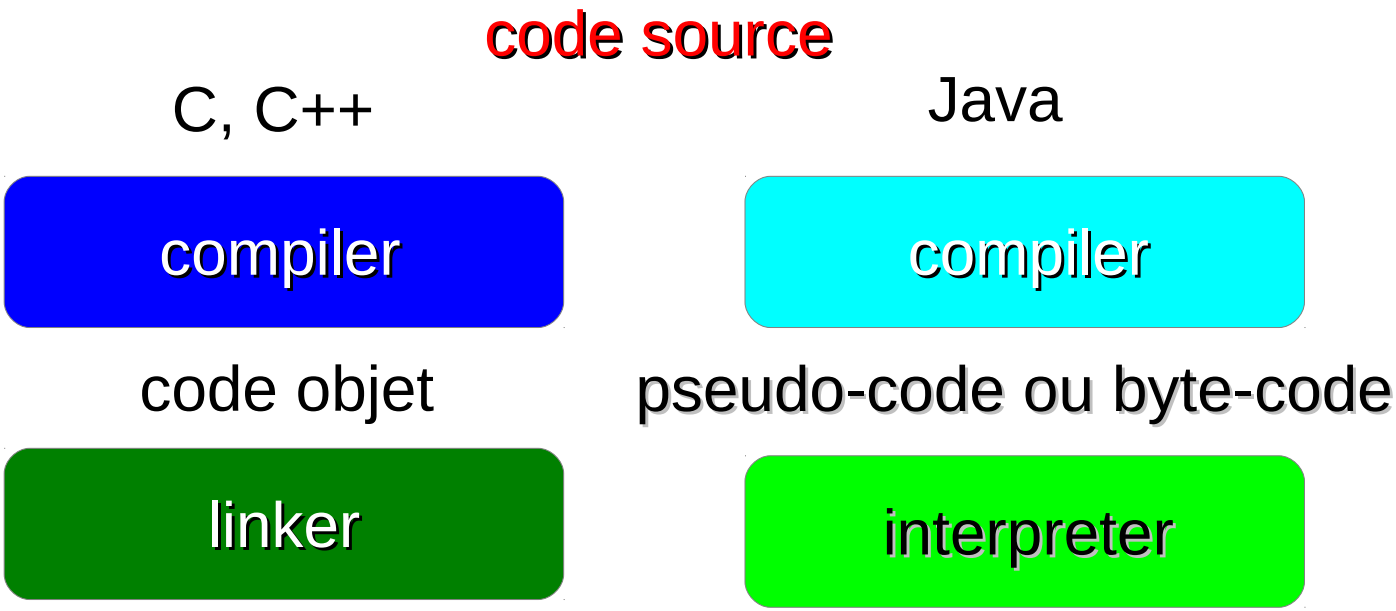
1. Analysis and Design

2. Coding

3. Compilation
4. Tests
5. Error correction

Translation of programs

– source code
– target code
– object code
– pseudo-code or byte-code



The program «Ni hau»

```
/ * first program
- to store in Nihau.java
- to compile by:  javac Nihau.java
- to execute by:  java Nihau
*/
public class Nihau {
public static void main (String[] args) {
System.out.println("Ni hau!");
}
}
```

Compilation and execution of «Nihau»

Compilation

```
Java/Essais> javac Nihau.java
Java/Essais>
```

Interpretation of byte-code in : `Nihau.class`

```
Java/Essais> java Nihau
Ni hau !
```

Program (class) « Conversion »

```
public class Conversion {
public static void main(String[] args) {
// variables of program
double euros, renminbis ;
// message to capture
System.out.println("Give the amount in euros:");
// reading to variable euros
euros = Keyboard.readDouble( );
// the conversion
renminbis = euros * 8.21;
// display the resultat
System.out.println("The converted amount in
renminbis:" + renminbis);
}
}
```

Program (class) « Conversion »

The compilation and the execution (interpretation) of Conversion program.

```
Java/Essais> javac Conversion.java
Java/Essais> java Conversion
Give the amount in euros :
-3
The converted amount in renminbis: ????
Java/Essais> java Conversion
Give the amount in euros :
15.5
The converted amount in renminbis: ????
```

Program in Java

This program has a skeleton similar to any other Java program. Here it is:

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

Program includes reserved words like:

public, class, static, void, main

It also includes names that are given by the programmer with considerable freedom.

This is the case of the name of the program: Conversion and data names euros and renminbi.

Program in Java

This program has a skeleton similar to any other Java program. Here it is:

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

A program is built from at leasť one class and one method qui called **`main`**.
It only remains to give a name to the program between the reserved word **`class`** and a sequence of instructions between the curly braces of the **`main`** method.

Elements of program

- Variable declarations.

Statements that are used to give a name to a memory cell, where you can store the values a the time of program execution.

Once a variable is declared and it has a value, we can use its value.

Elements of program

- The input-output instructions.

Our program calculates the renminbi from the amount in euros that is given by the user at the time of execution.

We have to bring the program data through the keyboard.

In Java, as in any other programming language, predefined commands that can do this.

(fetch a value from the keyboard or a file ; send the value out of the program to the screen, etc ...).

Elements of program

- The assignment statement (=)

are to manipulate the declared variables

The assignment statement puts the value of what is on the right side of the = sign in the named variable on the left side of the = sign.

The name of variables

`euros` or `renminbi` are the names of variables, they are freely chosen by the author of the program. There are some constraints in the choice of symbols constituting the variable names. They are:

- Variable names are identifiers that is to say necessarily begin with a letter, uppercase or lowercase, which may or may not be followed by as many characters as you want from the set : `. . . z, A. . . Z, 0 .. 9, _, $`

- A variable name can not be a reserved word: (**`abstract, boolean, if, public, class, private, static,`** etc).

Examples :

- `a, id_a` and `X1` are valid variable names,
- while `1X` and `x-x` are not.

The type of the variable

A type is a particular set of values known to the machine. We will work with the following Java types:

– The type **`int`** denotes the set of all integers representable in 32-bit machine, with the values in the range $\{-2^{31}, \ldots, 2^{31}-1\}$.

– The type **`double`** denotes the numbers with floating point (double-precision 64-bit). The elements of this set are (examples) -12.98,0.0, 0.1, . . .18.58 . . .

– The type **`boolean`** models the two truth values in propositional logic. Its elements are **`true`** and **`false`** .

– The type **`char`** models all Unicode characters (16 bits). Its elements are characters surrounded by single quotes, eg: ‘a‘, ‘2‘, ‘@‘ , 'à', 'ç'

– The type **`String`** models the strings. Its elements are sequences of characters surrounded by double quotes: "hello", "enter an amount in euros: ?", "a" . . .

Syntax of variable declarations

To declare a variable, you must give a type name from : `int`, `double`, `char`, `boolean`, `String` followed by a variable name that you invent.

To declare multiple variables of the same type at the same time, we must give a type name followed by the names of your variables (separated by commas).

Examples :

`int x ; String hello;` are correct

while

`entier x ;` or `x int ;` are rejected

Assigning a value to a variable

Once a variable has been declared, you can assign a value. To do this, you must use the assignment statement whose syntax is:

name variable = expression ;

for example : `x=2 ;`

It is good practice to give an initial value to a variable when it is declared. Java allows us to assign a value to a variable at the time of its declaration:

for example : `int x=2 ;`

If you declare two variables on the same line, the initialization looks like this:
`int x=0, y=0 ;`

To the right of the = sign, you can put any value of the correct type.

Expressions (and assignments)

Expressions are calculations to get a value of a certain type. They are used to give a value to a variable in an assignment. This value is calculated first and then put in the memory space reserved for the variable. An example of arithmetic expression is:

(18+20)/2

This expression is a calculation to compute an integer. This is an expression to assign a value to a variable declared of type `int`

We can write the assignment:

```
X=(18+20)/2;
```

We are accustomed to use arithmetic expressions in everyday life. In the programs, the terms may include other types of data such as characters and booleans.

Constituents of expressions

- A base value or literal, such as 18, 20 and 2. Each type has its own java base values.

- An operator who computes a result using one or two operands. For example, + for adding two numbers that are the operands. Each type has its own java operators. Operands are not necessarily base values: they are expressions that calculate the value used in the calculation. In our example, the division operator / has two operands : (18 +20) and 2. The first operand is an expression involving the + operator.

- A variable. When a variable appears in an expression, the calculation is carried out to fetch the contents of the variable, i.e., the value stored in the memory location associated with the variable.

- A method call. We present such expressions a little later in this lecture.

Operators

- For numbers: the arithmetic operators are addition (+), subtraction (-), multiplication (*), division (/). There is also the modulo operator (%), which provides the remainder of the integer division.

Example : 7% 3 provides the remainder of the division of 7 by 3, that is to say 1.

- For booleans: there are operators like :

&& - logic and , || logic or , ! logic not

- For strings: there is a very useful operator called concatenation, which allows butt paste two strings. This operator is denoted by +.

For example :
"abcd" + "xyz" is an expression, a calculation whose result is "abcdxyz."

Operators

- Comparison operators: they are used to compare two values of the same type. The result of the comparison is a **`boolean`**. These operators are:

– equal ==. Gives **`true`** if the two compared values are equal, **`false`** otherwise. Example : 1==4 gives **`false`**, 'a'=='a' gives **`true`**
– smaller (<). Gives **`true`** if the first value is strictly smaller than the second, **`false`** otherwise. Exemple : 1<4 gives **`true`**, 'a' < 'b' gives **`true`** (order between characters).
– bigger (>). Gives **`true`** if the first value is strictly bigger than the second, **`false`** otherwise. Example : 1>4 gives **`false`**.
– bigger or equal (>=). Gives **`true`** if the first value is bigger than or equal to the second, **`false`** otherwise.
– smaller or equal (<=). Gives **`true`** if the first value is smaller than or equal to the second, **`false`** otherwise.
– different (!=). Gives **`true`** if the values are different and **`false`** if they are equal.

Boolean expressions

Boolean expressions sometimes cause problems for novice programmers. These are expressions like the others; they calculate a value and can be used in an assignment to a variable of type boolean. Examples of **boolean** expressions used in perfectly correct assignments.

```
boolean varbool ;
varbool = true ;
varbool = true||false ;
varbool = 14<=5 ;
varbool = (14>5)&&('a'!='b');
```

Two boolean expressions are equivalent if they denote the calculations have the same result. For example, if **vb** is a boolean variable, the two expressions **vb==true** and **vb** are equivalent. If **vb** contains the **true**, then **vb==true** is also **true**. If **vb** contains **false**, tahen **vb==true** is also **false**. For this reason, we never write an expression such as **vb==true** , because it contains an unnecessary computation.

The method Keyboard.writeStringln()

The initial program gives the order to display the message on the screen `Amount in euros?`. It does this by using a function written for you. This type of function is called a method in Java. This method - `System.out.println()` is part of the `System.out` class. You can use it as many times as you wish.

To work, this method requires that the user transmits information: the string that he wish to display on the screen. The information transmitted by the user of the method is what is between the parentheses following the method name. This is called the argument or parameter of the method.

Here we use `System.out.println` to display the message *`Amount in euros ?`* by calling :

```
System.out.println("Amount in euros ? :);
```

To display « Hen hau », you need to call :

```
System.out.println("Hen hau").
```

The method Keyboard.readDouble()

```
euros=Keyboard.readDouble();
```

This line is an assignment. At the right of the = sign you find the method call. The called method is named `readDouble()` ; it can be found in `Keyboard` class.

The fact that there is nothing between the parentheses indicates that this method does not require that the program would provide information. It is a method without parameters. However, the fact that this call is at the right of an assignment indicates that the result of its execution produces a result (which will be in the variable euros). This result is the value from the Keyboard. This is called the return value or result of the method.

As for the parameters of the method, the fact that the methods return a value or not is determined by the author of the method once and for all. He also sets the type of the return value. Method, when it returns a value, always returns a value of the same type. For `readDouble()`, this value is of type `double`.

Methods and expressions

When returning a result, the method calls can appear in expressions.

Example : The method `Math.min()` : takes 2 parameters of type `int` and returns one value of type `int` : the smaller of the two parameters. In this way the instruction `x = 3 + (Math.min(4,10) + 2) ;` gives `x` the value 9 because 3+(4+2) is 9.

The instruction : `x = 3 + (Keyboard.readInt() + 2) ;` also has a meaning.

It will run as follows: to calculate the right value, execution will wait until a keyboard key is pressed (a blinking cursor on the screen will indicate this). As soon as the user presses a key, the calculation of `Keyboard.readInt()` ends with the result for that value..

Suppose the user has pressed 6. 3+6+2 gives 11. The value of `x` will be 11.

Method calls in general

Classes containing many methods exist in Java or in common language library, or in particular directories that must indicated. Any existing method has a name, belongs to a class, has parameters whose number, type and order are fixed. A method can return a value, the type of the return value is fixed. To call a method, you must know all this information. The method call will conform to the following syntax:

```
NomClasse.NomMethode(par1,...,parn);
```

We can see a method as a black box capable of performing treatments on entries (parameters)of the method. The values of the parameters may vary for each method call. At output we obtain: either a value, the result of the calculations, or a change in status or a change in the status of the machine also called effect, sometimes both.

Display methods have a visible effect, but they do not return result.

The Keyboard library

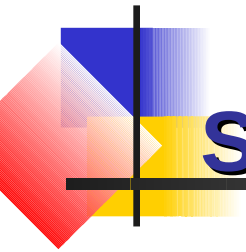
The `Keyboard` class is a program written to facilitate the usageof keyboard. We will use it in our examples. It includes methods of input and output terminal for the five predefined types:

`int, double, boolean, char` et **`String`**.

To use it, simply call the method you want preceded by the class name. For example, `Keyboard.readInt()` returns the first integer grabbed from the keyboard.

Input methods in **`Keyboard`**:

```
Keyboard.readInt()
Keyboard.readDouble()
Keyboard.readChar()
Keyboard.readBoolean()
Keyboard.readString()
```

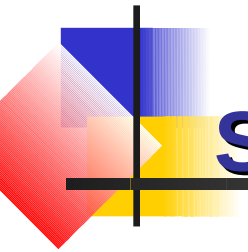


Summary

A computer:
- Processor (CPU), memory, peripherals

A program:
- Instructions, data, I/O

Compilation, interpretation:
- Lexis, syntax, semantics



Summary

Elements of the program:

- Statements - Variables

- I/O instructions

- expressions

Methods and method calls

I/O methods by class `Keyboard`