Software development using Java

By Mikaël Morelle

Edited in September 2024

mail: mikael.morelle@junia.com

Course Objectives

- General introduction to Object Oriented Programming
  - Based on the Java syntax & particularities
- Pre requirements :
  - Be attentive
  - Basic knowledge in algorithmics
  - Basic knowledges in C programming
- Practice through Java programming labs

Course Objectives

- 4 learning sessions + 3 weeks of practicing
- C programming review
- Why upgrading your programming skills to Object Oriented Programming (OOP)
- Java vs C++ presentation and environment
- OOP begins
  - Classes – Objects
  - Naming conventions
  - Methods, constructors and overloading
  - Keywords for encapsulation (public – protected – private).
  - Manage the Data visibility, protection and integrity
  - Keywords : class, this, static

Course Objectives

- Inheritance
  - Class hierarchy (class diagram)
  - Keywords : extends, super, final
- Abstraction
  - Keywords : abstract, interface, implements
- Exceptions
  - Keywords : try, catch, finally, throw, throws,

Course grading

- A final exam (50%)
  - Theoretical aspects
  - Evaluates your understanding of OOP concepts
  - 3 parts
    - ✓ Definitions and explanations
    - ✓ MCQ
    - ✓ Programming
    - ❖ No laptop or PC allowed, no documents
- A labwork evaluation (50%)
  - To be defined

Part 1

C programming review

if – else

for

while

do while

switch

Basic C programming - if

```java
public class IfSample1 {
 // C programming review
    public static void main(String[] args) {
        int number;
        int initializedNumber = 7;

        char character1 = 'a';
        char character2 = 65;

        number = 3;

        if(character1 > character2)
            System.out.print("'a' is greater than 65\n");

        System.out.printf("number = \t\t%d\ninitializedNumber = \t%d \n", number, initializedNumber);
        System.out.println("1st Character = " + character1 + ";  2nd Character = " + character2);
    }
}
```

```
compile:
run:
'a' is greater than 65
number =                3
initializedNumber =     7
1st Character = a;  2nd Character = A
BUILD SUCCESSFUL (total time: 2 seconds)
```

Basic C programming – ASCII table

Decimal	Hexadecimal	Binary	Octal	Char	Decimal	Hexadecimal	Binary	Octal	Char	Decimal	Hexadecimal	Binary	Octal	Char
0	0	0	0	[NULL]	48	30	110000	60	0	96	60	1100000	140	`
1	1	1	1	[START OF HEADING]	49	31	110001	61	1	97	61	1100001	141	a
2	2	10	2	[START OF TEXT]	50	32	110010	62	2	98	62	1100010	142	b
3	3	11	3	[END OF TEXT]	51	33	110011	63	3	99	63	1100011	143	c
4	4	100	4	[END OF TRANSMISSION]	52	34	110100	64	4	100	64	1100100	144	d
5	5	101	5	[ENQUIRY]	53	35	110101	65	5	101	65	1100101	145	e
6	6	110	6	[ACKNOWLEDGE]	54	36	110110	66	6	102	66	1100110	146	f
7	7	111	7	[BELL]	55	37	110111	67	7	103	67	1100111	147	g
8	8	1000	10	[BACKSPACE]	56	38	111000	70	8	104	68	1101000	150	h
9	9	1001	11	[HORIZONTAL TAB]	57	39	111001	71	9	105	69	1101001	151	i
10	A	1010	12	[LINE FEED]	58	3A	111010	72	:	106	6A	1101010	152	j
11	B	1011	13	[VERTICAL TAB]	59	3B	111011	73	;	107	6B	1101011	153	k
12	C	1100	14	[FORM FEED]	60	3C	111100	74	<	108	6C	1101100	154	l
13	D	1101	15	[CARRIAGE RETURN]	61	3D	111101	75	=	109	6D	1101101	155	m
14	E	1110	16	[SHIFT OUT]	62	3E	111110	76	>	110	6E	1101110	156	n
15	F	1111	17	[SHIFT IN]	63	3F	111111	77	?	111	6F	1101111	157	o
16	10	10000	20	[DATA LINK ESCAPE]	64	40	1000000	100	@	112	70	1110000	160	p
17	11	10001	21	[DEVICE CONTROL 1]	65	41	1000001	101	A	113	71	1110001	161	q
18	12	10010	22	[DEVICE CONTROL 2]	66	42	1000010	102	B	114	72	1110010	162	r
19	13	10011	23	[DEVICE CONTROL 3]	67	43	1000011	103	C	115	73	1110011	163	s
20	14	10100	24	[DEVICE CONTROL 4]	68	44	1000100	104	D	116	74	1110100	164	t
21	15	10101	25	[NEGATIVE ACKNOWLEDGE]	69	45	1000101	105	E	117	75	1110101	165	u
22	16	10110	26	[SYNCHRONOUS IDLE]	70	46	1000110	106	F	118	76	1110110	166	v
23	17	10111	27	[ENG OF TRANS. BLOCK]	71	47	1000111	107	G	119	77	1110111	167	w
24	18	11000	30	[CANCEL]	72	48	1001000	110	H	120	78	1111000	170	x
25	19	11001	31	[END OF MEDIUM]	73	49	1001001	111	I	121	79	1111001	171	y
26	1A	11010	32	[SUBSTITUTE]	74	4A	1001010	112	J	122	7A	1111010	172	z
27	1B	11011	33	[ESCAPE]	75	4B	1001011	113	K	123	7B	1111011	173	{
28	1C	11100	34	[FILE SEPARATOR]	76	4C	1001100	114	L	124	7C	1111100	174	|
29	1D	11101	35	[GROUP SEPARATOR]	77	4D	1001101	115	M	125	7D	1111101	175	}
30	1E	11110	36	[RECORD SEPARATOR]	78	4E	1001110	116	N	126	7E	1111110	176	~
31	1F	11111	37	[UNIT SEPARATOR]	79	4F	1001111	117	O	127	7F	1111111	177	[DEL]
32	20	100000	40	[SPACE]	80	50	1010000	120	P					
33	21	100001	41	!	81	51	1010001	121	Q					
34	22	100010	42	"	82	52	1010010	122	R					
35	23	100011	43	#	83	53	1010011	123	S					
36	24	100100	44	$	84	54	1010100	124	T					
37	25	100101	45	%	85	55	1010101	125	U					
38	26	100110	46	&	86	56	1010110	126	V					
39	27	100111	47	'	87	57	1010111	127	W					
40	28	101000	50	(	88	58	1011000	130	X					
41	29	101001	51	)	89	59	1011001	131	Y					
42	2A	101010	52	*	90	5A	1011010	132	Z					
43	2B	101011	53	+	91	5B	1011011	133	[					
44	2C	101100	54	,	92	5C	1011100	134	\					
45	2D	101101	55	-	93	5D	1011101	135	]					
46	2E	101110	56	.	94	5E	1011110	136	^					
47	2F	101111	57	/	95	5F	1011111	137						

American **S**tandard **C**ode for **I**nformation **I**nterchange

Basic C programming - if

```java
public class IfSample2 {
 // if statement example
    public static void main(String[] args) {
        int var1 = 10;
        int var2 = 0b1010;
        int var3 = 0x1F;

        System.out.print("var1=" + var1);
        System.out.print(";  var2=" + var2);
        System.out.println(";  var3=" + var3);
        if (var1 > var2)
        {
            if (var2 > var3)
                System.out.println("var1 > var2 > var3");
            else
                System.out.println("var1 > var3 < var2");
        }
        else
            if (var2 > var3)
                System.out.println("var2 > var3 > var1");
            else
                System.out.println("var2 > var1 > var3");
    }
}
```

```
compile:
run:
var1=10;  var2=10;  var3=31
var2 > var1 > var3
BUILD SUCCESSFUL (total time: 2 seconds)
```

Basic C programming - if

- Nested if-else:

```
if (condition) statement1;
else if (condition) statement1;
    else if (condition) statement1;
        ....
```

- No limitation for nested if-else

- A strong advice: **NEVER MORE THAN 3 NESTED IF-ELSE** in a method (will be one criteria for the evaluation of your work)

Basic C programming

```java
public class ConditionalOperator {
 // C programming review
    public static void main(String[] args) {
        int value1 = 10;
        int value2 = 20;
        int value3 = 30;

        String result = "";

        System.out.println(result = (value2 > value1) ? " false" : " true");
        System.out.println(result += (value3 < value2) ? " true" : " false");
    }
}
```

false false true

true true false

true true

true false

false true

false false false

true true true

true true true

false false false

true false

false true

```
compile:
run:
 false
 false false
BUILD SUCCESSFUL (total time: 1 second)
```

Conditional operator ?

- A conditional operator is a notational variant on certain forms of the if else statement.
- Example:

```
if (number1 > number2)
   max = number1;
else
    max = number2;
```

Is equivalent to this conditional operator:

```
max = (number1>number2) ? number1 : number2;
```

Basic C programming - for

```java
public class ForLoop1 {
 // C programming review
    public static void main(String[] args) {
        int loop;
        int result=1;
        int pow = 10;

        for (loop=0; loop <pow ; loop++)
            result *= 2;

        System.out.println("2^" + pow + " = " + result);
    }
}
```

```
compile:
run:
2^10 = 1024
BUILD SUCCESSFUL (total time: 1 second)
```

Basic C programming - for

```java
public class ForLoop2 {
 // C programming review
    public static void main(String[] args) {
        int loop1, loop2;
        int size = 10;

        for (loop1=0; loop1 <size ; loop1++){
            for (loop2=0; loop2 <= loop1; loop2++)
                System.out.print("* ");
            System.out.println("");
        }
    }
}
```

```
run:
*
* *
* * *
* * * *
* * * * *
* * * * * *
* * * * * * *
* * * * * * * *
* * * * * * * * *
* * * * * * * * * *
BUILD SUCCESSFUL (total time: 1 second)
```

Basic C programming - while

```java
public class WhileLoop {
 // C programming review
    public static void main(String[] args) {
        int param = 0;
        String result = "";

        while (result.isEmpty()){
            if (param == 5)
                result = "Ok";
            else
                System.out.print("Not yet");

            for (int i = 0; i < param ; i++)
                System.out.print(".");
            System.out.println("");

            param++;
        }
    }
}
```

```
run:
Not yet
Not yet.
Not yet..
Not yet...
Not yet....
.....
BUILD SUCCESSFUL (total time: 0 seconds)
```

```java
public class WhileLoop {
 // C programming review
    public static void main(String[] args) {
        int param = 0;
        String result = "";

        while (result.isEmpty()){
            if (param == 5)
                result = "Ok";
            else
                System.out.print("Not yet");

            for (int i = 0; (result.isEmpty() && (i < param)) ; i++)
                System.out.print(".");
            System.out.println("");

            param++;
        }
    }
}
```

```
run:
Not yet
Not yet.
Not yet..
Not yet...
Not yet....

BUILD SUCCESSFUL (total time: 1 second)
```

Basic C programming - while

```java
public class WhileLoop {
 // C programming review
    public static void main(String[] args) {
        int param = 0;
        String result = "";

        while (result.isEmpty()){
            if (param == 5)
                result = "Ok";
            else
                System.out.print("Not yet");

            for (int i = 0; i < param ; i++)
                System.out.print(".");
            System.out.println("");

            param++;
        }
    }
}
```

```
run:
Not yet
Not yet.
Not yet..
Not yet...
Not yet....
.....
BUILD SUCCESSFUL (total time: 0 seconds)
```

```java
public class WhileLoop {
 // C programming review
    public static void main(String[] args) {
        int param = 0;
        String result = "";

        while (result.isEmpty()){
            if (param == 5)
                result = "Ok";
            else
                System.out.print("Not yet");

            if(result.isEmpty()){
                for (int i = 0; i < param ; i++)
                    System.out.print(".");
                System.out.println("");
            }

            param++;
        }
    }
}
```

```
run:
Not yet
Not yet.
Not yet..
Not yet...
Not yet....
BUILD SUCCESSFUL (total time: 0 seconds)
```

Basic C programming – do while

```java
public class DoWhileLoop {
 // C programming review
    public static void main(String[] args) {
        int param = 0;
        String sentence = "Programming is great!";
        char research = '!';
        char c;
        do{
            c = sentence.charAt(param);
            param++;
        }
        while (c != research);

        System.out.println("The character '" + research + "' is located at position : " + param);
    }
}
```

```
run:
The character '!' is located at position : 21
BUILD SUCCESSFUL (total time: 1 second)
```

Basic C programming – do while

```java
public class DoWhileLoop {
 // C programming review
    public static void main(String[] args) {
        int param = 0;
        String sentence = "Programming is great!";
        char research = '?';
        char c;
        do{
            c = sentence.charAt(param);
            param++;
        }
        while (c != research);

        System.out.println("The character '" + research + "' is located at position : " + param);
    }
}
```

Beware of infinite loop

```
run:
Exception in thread "main" java.lang.StringIndexOutOfBoundsException: String index out of range: 21
        at java.base/java.lang.StringLatin1.charAt(StringLatin1.java:48)
        at java.base/java.lang.String.charAt(String.java:709)
        at correction.DoWhileLoop.main(DoWhileLoop.java:21)
C:\Users\mikael.morelle\Documents\NetBeansProjects\Correction\nbproject\build-impl.xml:1330: The following error occurred while executing this line:
C:\Users\mikael.morelle\Documents\NetBeansProjects\Correction\nbproject\build-impl.xml:936: Java returned: 1
BUILD FAILED (total time: 0 seconds)
```

Basic C programming – do while

```java
public class DoWhileLoop {
 // C programming review
    public static void main(String[] args) {
        int param = 0;
        String sentence = "Programming is great!";
        char research = '?';
        char c;

        do{
            c = sentence.charAt(param);
            param++;
        }
        while ((c != research) && (param != sentence.length()));

        System.out.print("The character '" + research);
        if(param < sentence.length())
            System.out.println("' is located at position : " + param);
        else
            System.out.println("' is not present");
    }
}
```

```
run:
The character '?' is not present
BUILD SUCCESSFUL (total time: 0 seconds)
```

Basic C programming – switch

```java
public class SwitchStatement {
 // C programming review
    public static void main(String[] args) {
        int value = 3;
        String text = "";
        switch(value){
            case 0 : text = "zero";
            case 1 : text = "one";
            case 2 : text = "two";
            case 3 : text = "three";
            case 4 : text = "four";
            case 5 : text = "five";

            default : text = "not a correct number !";
        }
        System.out.println("the value is " + text);
    }
}
```

```
run:
the value is not a correct number !
BUILD SUCCESSFUL (total time: 0 seconds)
```

```java
public class SwitchStatement {
 // C programming review
    public static void main(String[] args) {
        int value = 3;
        String text = "";
        switch(value){
            case 0 : text = "zero";break;
            case 1 : text = "one";break;
            case 2 : text = "two";break;
            case 3 : text = "three";break;
            case 4 : text = "four";break;
            case 5 : text = "five";break;

            default : text = "not a correct number !";
        }
        System.out.println("the value is " + text);
    }
}
```

```
run:
the value is three
BUILD SUCCESSFUL (total time: 0 seconds)
```

Basic C programming – switch

```java
public class SwitchStatement {
 // C programming review
    public static void main(String[] args) {
        String text = "four";
        int value = 3;
        switch(text){
            case "zero" : value = 0; break;
            case "one" : value = 1; break;
            case "two" : value = 2; break;
            case "three" : value = 3; break;
            case "four" : value = 4; break;
            case "five" : value = 5; break;

            default : value = -1;
        }
        System.out.println("the text is " + value);
    }
}
```

```
run:
the text is 4
BUILD SUCCESSFUL (total time: 0 seconds)
```

```java
public class SwitchStatement {
 // C programming review
    public static void main(String[] args) {
        String text = "";
        int value = 2;
        switch(value){
            case 0 :
            case 2 :
            case 4 : text = "even"; break;
            case 1 :
            case 3 :
            case 5 : text = "odd"; break;

            default : text = "not a correct value"; break;
        }
        System.out.println("the value is " + text);
    }
}
```

```
run:
the value is even
BUILD SUCCESSFUL (total time: 0 seconds)
```

Basic C programming – switch

```java
public class SwitchStatement {
 // C programming review
    public static void main(String[] args) {
        String text = "";
        int value = 2;
        switch(value){
            case 0 :
            case 2 :
            case 4 : text = "even"; break;
            case 1 :
            case 3 :
            case 5 : text = "odd"; break;

            default : text = "not a correct value"; break;
        }
        System.out.println("the value is " + text);
    }
}
```

```
run:
the value is even
BUILD SUCCESSFUL (total time: 0 seconds)
```

```java
public class SwitchStatement {
 // C programming review
    public static void main(String[] args) {
        String text = "";
        int value = 3;
        text = switch(value){
            case 0,2,4 -> "even";
            case 1,3,5 -> "odd";

            default -> "not a correct value";
        };

        System.out.println("the value is " + text);
    }
}
```

```
run:
the value is odd
BUILD SUCCESSFUL (total time: 0 seconds)
```

break and continue

- **break** statement used in a while, for, do..while or switch statement causes immediate exit from that statement
  Execution continues with the first next statement after the control statement

- **continue** statement executed in a while, for or do..while statement skips the remaining statements and proceed to the next iteration

Type casting

- When, in a division, at least one operand are a floating-point type, division results is a floating-point type

15.0 / 2 evaluates to 7.5

- When both operands are integer types, division results is an integer type, the fractional part is discarded and the result is not rounded

15 / 2 evaluates to 7

Type casting

- When type casting from a floating-point to an integer type, the number is truncated, not rounded

```
(int) 2.9 evaluates to 2, not 3
```

- When the value of an integer type is assigned to a variable of a floating-point type, Java performs an automatic type cast called a type coercion

```
double d = 5;
```

- In contrast, it is illegal to place a double value into an int variable without an explicit type cast

```
int i = 5.5; // Illegal
int i = (int) 5.5 // Correct
```

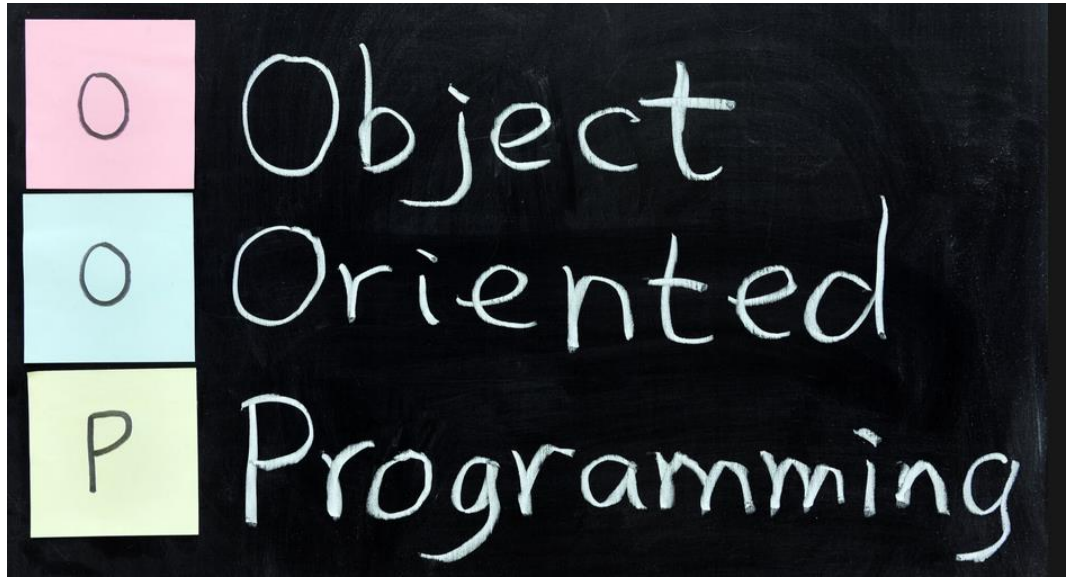
What is a good developer

A few rules:

- Think before programming
- Do it once
- Do it right
- Think reuse
- Do not program only for yourself
- Use coding standards (coding rules, design patterns, ...)

Part 2

From sequential to



Basic C vs OOP

sequential programing : instruction are executed in a determined order

1 main method

1 main method

Functions

Procedures

method : module which contains several instructions to perform one action

Naming restrictions :

only 1 function with the same name

Polymorphism :

a method can take several implementations

Difficult to represent complex elements (Structures, Union)

main OOP objective

objects = data + actions

C code is close to machine language (assembly) : ideal for embedded programming

OOP is close to reality, complex elements can be represented easily

Basic C vs OOP

Compiling a C program under Windows → executable file .exe

Compiling a C program under Linux → executable file .o

Executing a program developed under Windows on a Linux OS → need to recompile the program

Executing a program developed under Linux on a Windows OS → need to recompile the program

recompiling the program from Windows to Linux or reverse way → need to change sources files
different compile options …

Interoperability : execution of a program whatever the OS and whatever the hardware configuration required for every client – server based applications

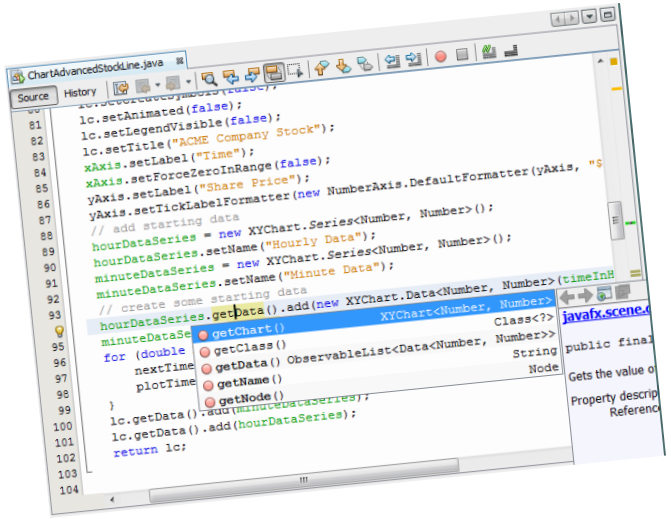
C++ vs Java

Both are Object Oriented Language

C++	Java
C++ has been created in early 80's	Java is created by Sun company in 1991 (bought by Oracle)
C++ is an extension of C language	Java is born Object but evolves regularly (v17 released in 09/21)
Platform dependant write once, compile anywhere	Platform independant write once, run anywhere
Memory is developer's business	Memory managed by System control (JVM + garbage collector)
Close to harware	Runs in a virtual environment
Programs execute faster	Known to be slower
Not restricted but ideal for embedded	Not restricted but ideal for Client – Server based applications
C++ is compiled and not interpreted	Java is both compiled and interpreted

Java – How to

- One basic tool: the JDK (Java Development Kit)
- Some Java IDE (Integrated Development Environment) to allow a greater productivity:



- IntelliJ
- Eclipse
- **Netbeans (the one used during Java 1 course)**
- Android Studio
- AIDE (Android IDE)
- Codenvy
- BlueJ
- Greenfoot…

https://hackr.io/blog/best-java-ides

Java – How to

MarsRoverViewer - NetBeans IDE 6.8

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

Projects
- MarsRoverViewer
  - Source Packages
    - META-INF.services
    - marsroverviewer
      - LoadImageTask.java
      - MarsRoverViewerAboutBox.java
      - MarsRoverViewerApp.java
      - MarsRoverViewerView.java
    - marsroverviewer.resources
    - marsroverviewer.resources.busyicons
  - Libraries

LoadImageTask - Navigator

Members View
- LoadImageTask :: Task<BufferedImage, Void>
  - LoadImageTask(Application app, URL url)
  - cancelled()
  - completionMessage(String resourceKey)
  - doInBackground() : BufferedImage
  - failed(Throwable e)
  - findImageReader(URL url) : ImageReader
  - finished()
  - getImageURL() : URL
  - interrupted(InterruptedException e)
  - loadImage(ImageReader reader, IIOReadProgr
  - succeeded(BufferedImage image)

LoadImageTask.java

```
60  public class LoadImageTask extends Task<BufferedImage, Void> {
61      private static Logger logger = Lo                    ass.getName());
62      private final URL url;
63      private ImageReader imageReader =
64
65      public LoadImageTask(Application
66          super(app, "LoadImageTask");
67          this.url = url;
68      }
69
70      protected final URL getImageURL()
71          return url;
72      }
73
        @Override protected BufferedImage
75          IIOReadProgressListener rpl =
            @Override
77          public void imageProgress
78              setProgress(p, 0.0f,
79          }
80      };
81      message("startedLoadingImage",
82      imageReader = findImageReader(url);
83      return loadImage(imageReader, rpl);
84  }
85
86  private void completionMessage(String resourceKey) {
87      message(resourceKey, url, getExecutionDuration(TimeUnit.MILLISECONDS));
88  }
89
        @Override protected void cancelled() {
91      if (imageReader != null) {
92          imageReader.abort();
93      }
```

Boolean
BufferedImage
Byte
Imported Items; Press 'Ctrl+SPACE' Again for All Items

java.awt.image

public class BufferedImage extends Image implements WritableRenderedImage, Transparency

The BufferedImage subclass describes an Image with an accessible buffer of image data. A BufferedImage is comprised of a ColorModel and a Raster of image data. The number and types of bands in the SampleModel of the Raster must match the number and types required by the ColorModel to represent its color and alpha components. All BufferedImage objects have an upper left corner coordinate of (0, 0). Any Raster used to construct a BufferedImage must therefore have minX=0 and minY=0.

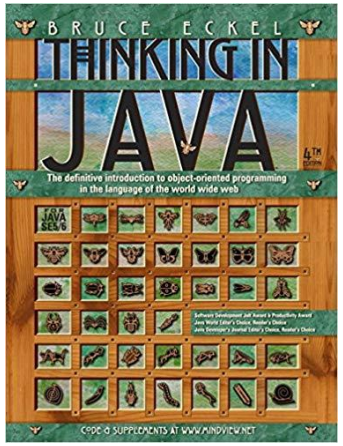
This class relies on the data fetching and setting methods of Raster, and on the color characterization methods of ColorModel.

See Also:

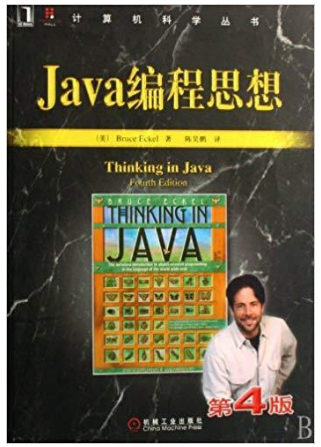
60:42 INS

Java – How to

- Official web site: https://docs.oracle.com/en/java/
- Official tutorials: https://docs.oracle.com/javase/tutorial/index.html
- Official web community: https://www.javaprogrammingforums.com/
- Non-Official but one of the most known source of information: https://openclassrooms.com/

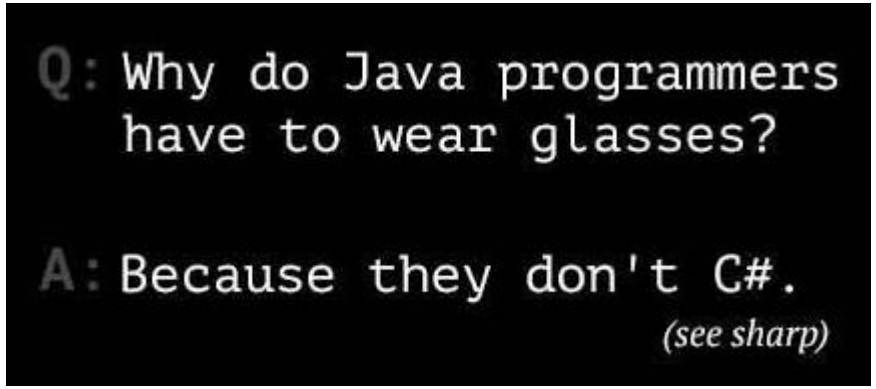


- **One great free book: "Thinking in Java" from Bruce Eckel**



Part 3

Introduction to Java



What is Java (1/5)

- A programming language created by Sun in 1991 originally for programming home appliances
- Java is generally linked to the web but it is not only for the web (integrated into web navigators in 1995)
- Current version is 17 since september 2021
- 3 languages in 1:
  - Java ME (Mobile Edition)
  - Java SE (Standard Edition) [Version used in this course]
  - Java EE (Enterprise Edition)

What is Java (2/5)

The idea :



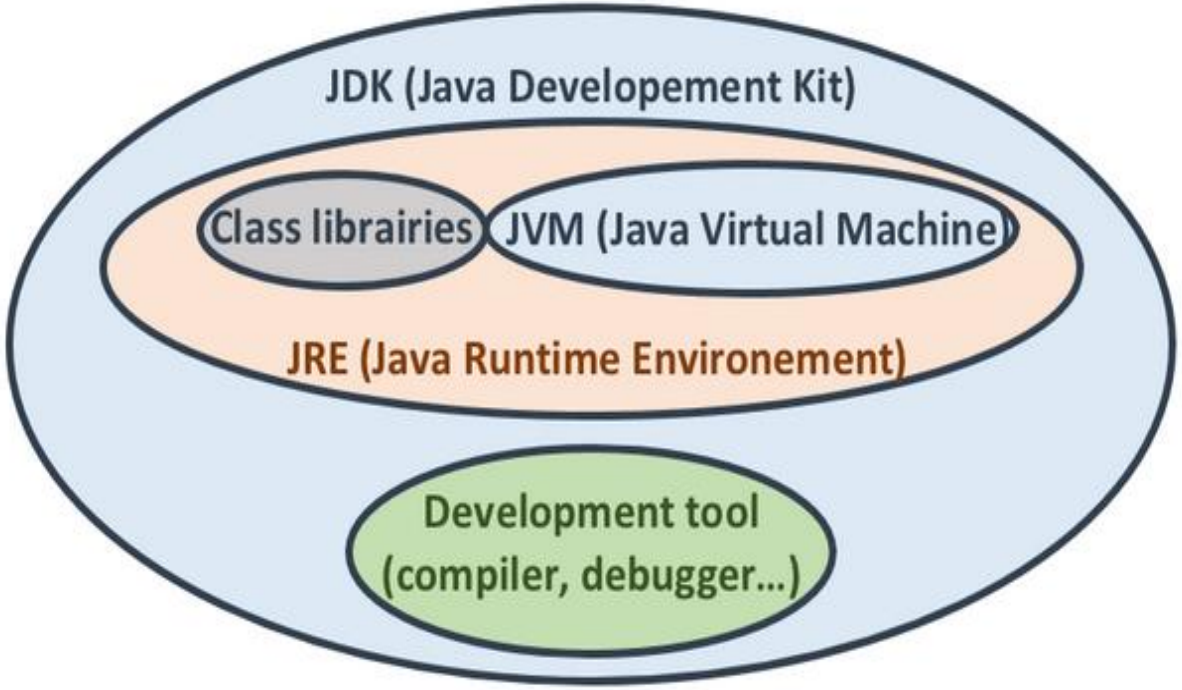
- Programming home appliances is a difficult task as they are controlled by a wide variety of computer processors

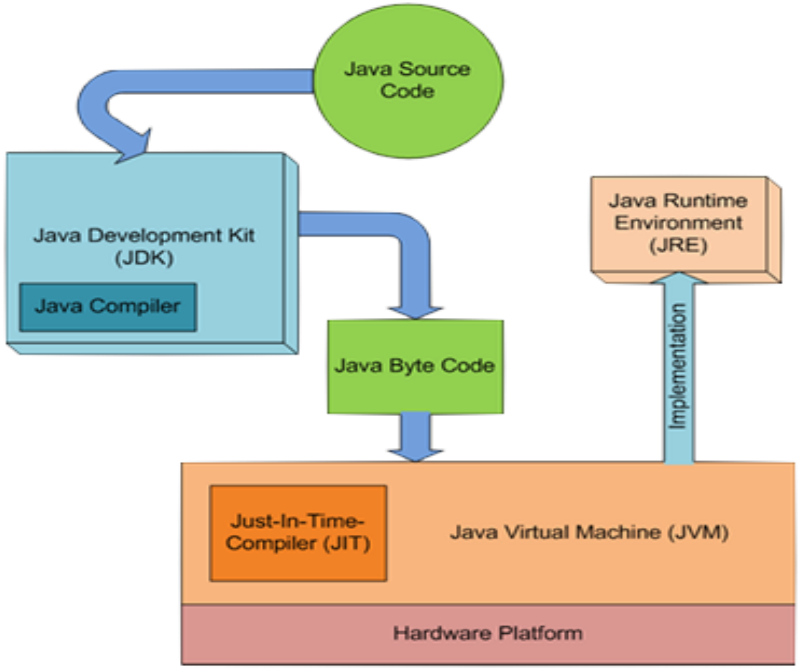
- Java team developed a two-step translation process :

  By the intermediate of a Java byte-code that is the same for all types of processors

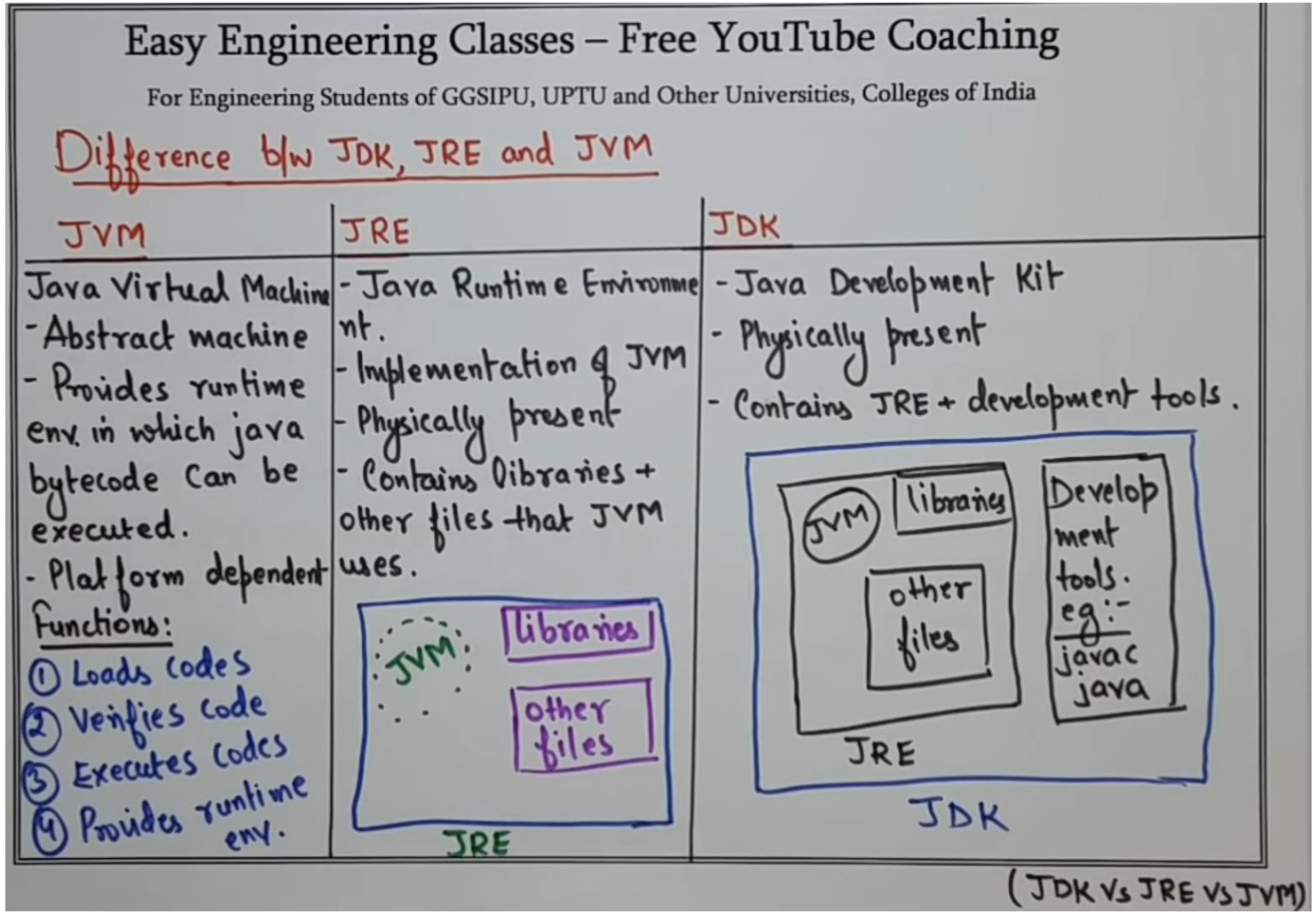
- Therefore, only a small, easy to write program was needed to translate byte code into the machine code for each processor

What is Java (3/5)





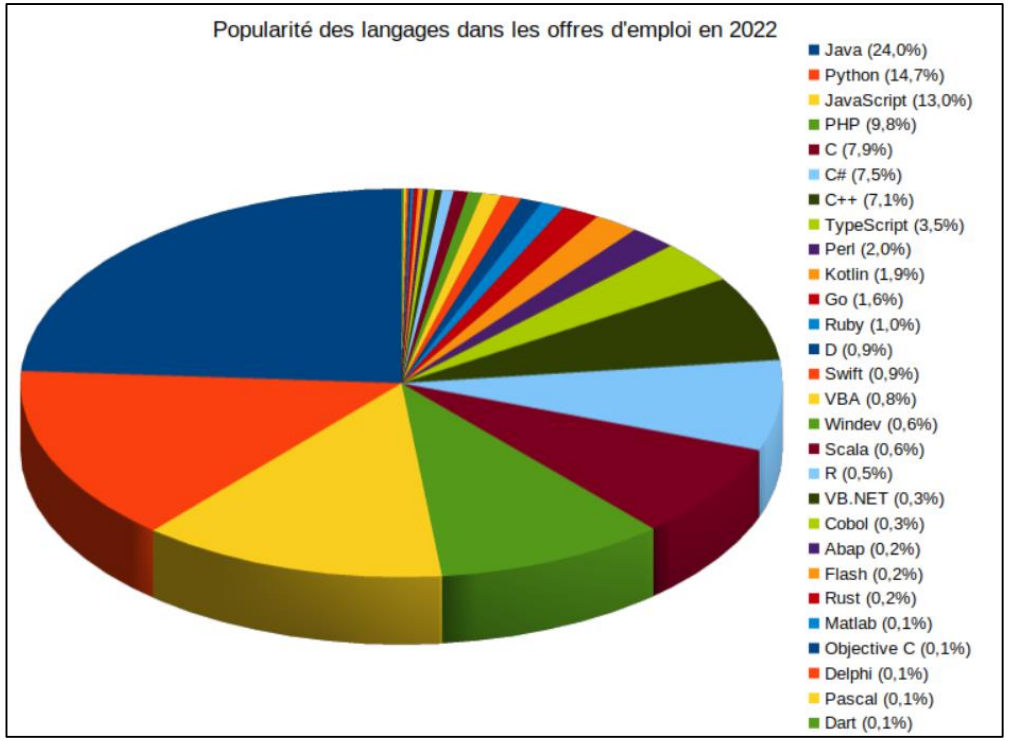
What is Java (4/5)

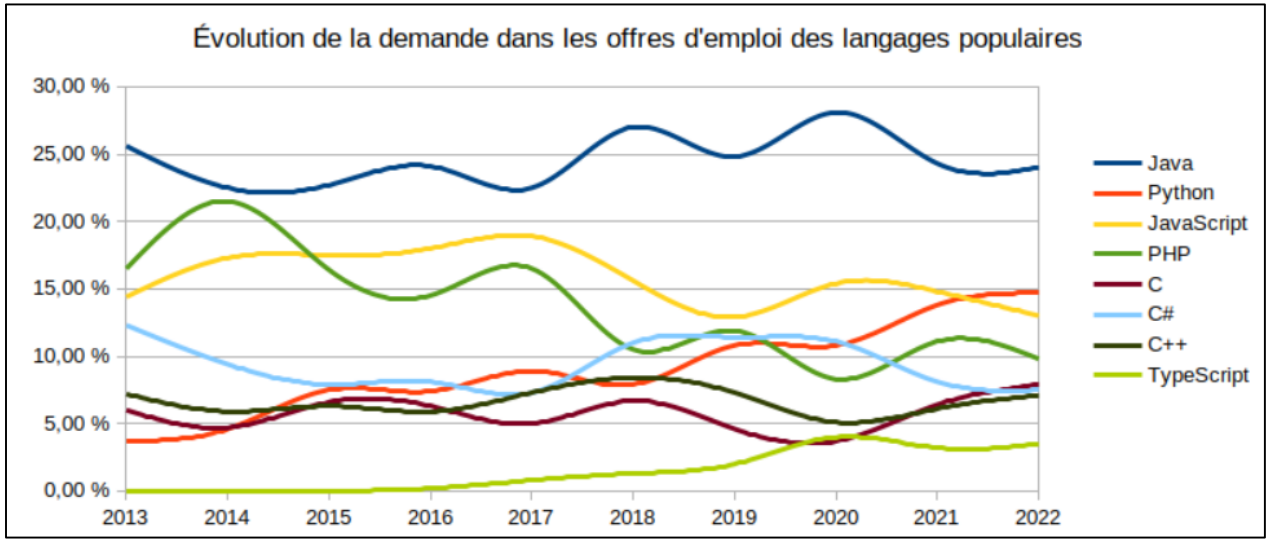


Source : x

What is Java (5/5)

Why learning Java ?





Required programming language over 30.000 job offers in 2022

Source : https://emploi.developpez.com/actu/343151/Emploi-informatique-2022-les-langages-les-plus-demandes-et-les-mieux-payes/
Edited September 2023, last visited September 2023

JUNIA Grande école d'ingénieurs
HEI·ISEN·ISA

Speakers Info Partner Program About Us JourneyZone AweZone

Welcome to JavaZone 2024

Every Year, the Javazone community presents a short video,

The 2013 about Javapocalypse : https://www.youtube.com/watch?v=E3418SeWZfQ

The 2014 about Game of Codes : https://www.youtube.com/watch?v=3vI_7os2V_o

Part 4

Let’s analyze a simple Hello World program

```
<hello-world />
```

Hello World in Java

```java
package helloworld;

/**
 *
 * @author mikael.morelle
 */
public class HelloWorld {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
        System.out.println("Hello World!");
    }
}
```

helloworld.HelloWorld

Search Results | Output - HelloWorld (run)

```
run:
Hello World!
BUILD SUCCESSFUL (total time: 0 seconds)
```

Hello World in Java

package helloworld;

- Each source file begins with a package definition
- What is a package ?
  - A solution to regroup different classes together
  - Will produce a JAR file
- Each instruction should be ended by a ;

```
1
2   package helloworld;
3
4   /**
5    *
6    * @author mikael.morelle
7    */
8   public class HelloWorld {
9
10      /**
11       * @param args the command line arguments
12       */
13      public static void main(String[] args) {
14          // TODO code application logic here
15          System.out.println("Hello World!");
16      }
17  }
18
```

helloworld.HelloWorld

Search Results | Output - HelloWorld (run)

```
run:
Hello World!
BUILD SUCCESSFUL (total time: 0 seconds)
```

Hello World in Java

public class HelloWorld { …… }

- Each program should be defined into a class (the class concept will be explained later in the course)
  - →Nothing outside classes
- Each class name should begin with an Uppercase letter
- Only One class per file
- name of the file = name of the class
  - →HelloWorld.java

```
1
2   package helloworld;
3
4   /**
5    *
6    * @author mikael.morelle
7    */
8   public class HelloWorld {
9
10      /**
11       * @param args the command line arguments
12       */
13      public static void main(String[] args) {
14          // TODO code application logic here
15          System.out.println("Hello World!");
16      }
17  }
18
```

helloworld.HelloWorld

Search Results | Output - HelloWorld (run)

```
run:
Hello World!
BUILD SUCCESSFUL (total time: 0 seconds)
```

Hello World in Java

public static void *main* (*String* [ ] args) {… }

- The main method, where every program begins
- A method is an “inside function” that could be executed through a class (very simplistic explanation)
- Each method name should begin with an lowercase letter
- A method always return something (here void means “no information returned”)
- A method can receive parameters (here args is an array of Strings)
- A string is a sequence of characters that exist as an object of the class **String**

```
1
2   package helloworld;
3
4   /**
5    *
6    * @author mikael.morelle
7    */
8   public class HelloWorld {
9
10      /**
11       * @param args the command line arguments
12       */
13      public static void main(String[] args) {
14          // TODO code application logic here
15          System.out.println("Hello World!");
16      }
17  }
18
```

helloworld.HelloWorld

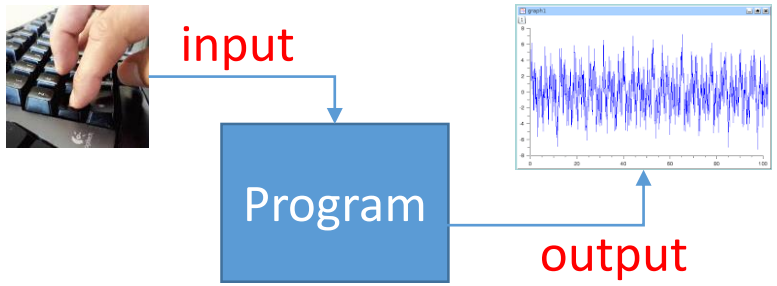
Search Results | Output - HelloWorld (run)

```
run:
Hello World!
BUILD SUCCESSFUL (total time: 0 seconds)
```

Hello World in Java

System.*out*.println (“Hello World !”) ;

- The class System provide methods for using standard input and output streams.
  - System.in: for input stream
  - System.out: for output stream



- println: method which prints a text on screen followed by a return carriage ‘\n’
- print: method which prints a text on screen without return carriage

```
1
2   package helloworld;
3
4   /**
5    *
6    * @author mikael.morelle
7    */
8   public class HelloWorld {
9
10      /**
11       * @param args the command line arguments
12       */
13      public static void main(String[] args) {
14          // TODO code application logic here
15          System.out.println("Hello World!");
16      }
17  }
18
```

helloworld.HelloWorld

Search Results | Output - HelloWorld (run)

```
run:
Hello World!
BUILD SUCCESSFUL (total time: 0 seconds)
```

Hello World in Java

Java Documentation (Javadoc)

```
/**
*
**/
```

2 stars for opening

2 stars for closing

- Used to describe:
  - a class
  - a method

Allow to give information about How-To, author, version, date, attributes, parameters, return value, exception thrown, …

Java Documentation will be evaluated

Comments:

1 star for opening

1 star for closing

```
1
2   package helloworld;
3
4   /**
5    *
6    * @author mikael.morelle
7    */
8   public class HelloWorld {
9
10      /**
11       * @param args the command line arguments
12       */
13      public static void main(String[] args) {
14          // TODO code application logic here
15          System.out.println("Hello World!");
16      }
17  }
18
```

helloworld.HelloWorld

Search Results | Output - HelloWorld (run)

```
run:
Hello World!
BUILD SUCCESSFUL (total time: 0 seconds)
```

- Class documentation
- Existing Constructors
- Constructors
- Methods

[String documentation]

Package **Class** Use Tree Deprecated Index Help

PREV CLASS NEXT CLASS
SUMMARY: NESTED | FIELD | CONSTR | METHOD

FRAMES NO FRAMES
DETAIL: FIELD | CONSTR | METHOD

java.helloworldpackage

Class Main

```
java.lang.Object
  └ java.helloworldpackage.Main
```

```
public class Main
extends java.lang.Object
```

It is just an Hello World program

Constructor Summary

`Main()`

Method Summary

`static void`	`main(java.lang.String[] args)` main method

Methods inherited from class java.lang.Object

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

Main

```
public Main()
```

Method Detail

main

```
public static void main(java.lang.String[] args)
```

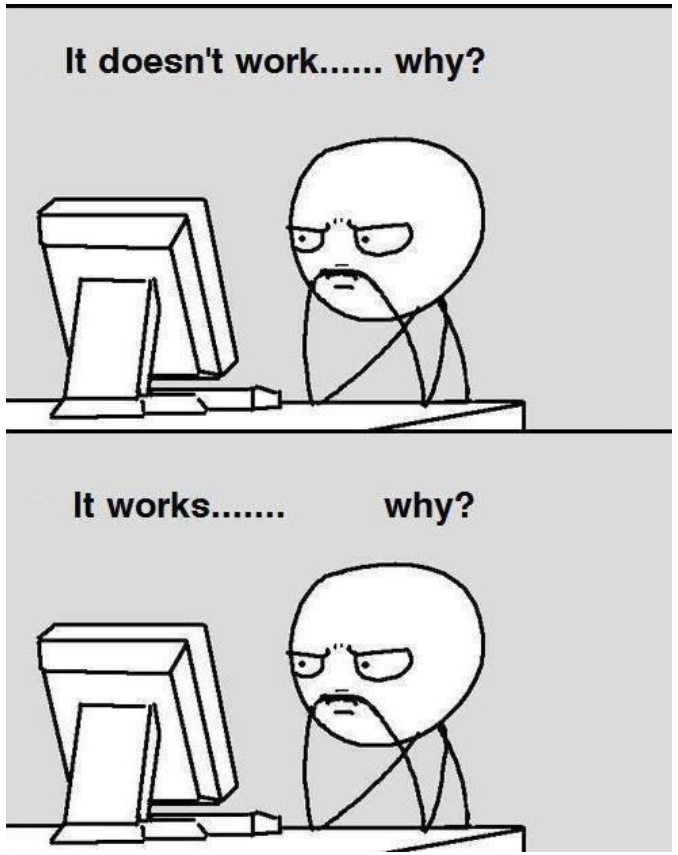
main method

Parameters:

`args` - the command line arguments

Part 5

Using simple data for simple programs



Primitive data types – Wrapper classes

8 different primitive data types exists :

```
boolean booleanVar;
//-----------------------------------------
byte byteVar;
char charVar;
short shortVar;
int intVar;
long longVar;
//-----------------------------------------
float floatVar;
double doubleVar;
//-----------------------------------------
```

- `boolean booleanVar;` : {false; true}
- Integer values:
  - `byte byteVar;` : $\{-2^7; 2^7-1\}$: {-128; +127}: 1 Byte
  - `char charVar;` : $\{0; 2^{16}-1\}$: {All characters, classics, symbols, alphabet, greek, …}: 2 Bytes
  - `short shortVar;` : $\{-2^{15}; 2^{15}-1\}$: {-32,768; +32,767}: 2 Bytes
  - `int intVar;` : $\{-2^{31}; 2^{31}-1\}$: {-2,147,483,648; +2,147,483,647}: 4 Bytes
  - `long longVar;` : $\{-2^{63}; 2^{63}-1\}$: {-9,223,372,036,854,775,908; +9,223,372,036,854,775,907} : 8 Bytes
- Floating-point values:
  - `float floatVar;` : {±3.40282347E+38F}: *IEEE 754* standard coding: 4 Bytes
  - `double doubleVar;` : {±1.79769313486231570E+308}: *IEEE 754* standard coding: 8 Bytes

Everything else is complex !

Primitive data types – Wrapper classes

For each primitive data type, a complex form exists

These are specific classes called ‘Wrapper Classes’

```
boolean booleanVar;
//-----------------------------------
byte byteVar;
char charVar;
short shortVar;
int intVar;
long longVar;
//-----------------------------------
float floatVar;
double doubleVar;
//-----------------------------------
```

```
Boolean booleanVar;
//-----------------------------------------
Byte byteVar;
Character charVar;
Short shortVar;
Integer intVar;
Long longVar;
//-----------------------------------------
Float floatVar;
Double doubleVar;
//-----------------------------------------
```

lowerCase

Primitive data type

UpperCase

Complex data type

Objects of a Complex type

Primitive data types – Wrapper classes

<u>Advantages of using wrapper classes</u> :

- Access to type conversion functions (From or To String for example)

```
Boolean booleanVar;
//-------------------------------------------
Byte byteVar;
Character charVar;
Short shortVar;
Integer intVar;
Long longVar;
//-------------------------------------------
Float floatVar;
Double doubleVar;
//-------------------------------------------
```

```
intVar.
    byteValue()                          byte
    compareTo(Integer anotherInteger)    int
    doubleValue()                        double
    equals(Object obj)                   boolean
    floatValue()                         float
    getClass()                           Class<?>
    hashCode()                           int
    intValue()                           int
    longValue()                          long
    notify()                             void
    notifyAll()                          void
    shortValue()                         short
    toString()                           String
    wait()                               void
    wait(long timeout)                   void
    wait(long timeout, int nanos)        void
```

- Possibility to set a value to null when not defined yet

```
Integer yearOfDeath = null;
```

Class String (1/5)

```
8   /**
9    *
10   * @author Mikael Morelle
11   * @version 1.1
12   */
13  public class Main
14  {
15
16      public static void main(String[] args)
17      {
18          String mot=new String();
19
20          mot="Mon 1er cours de Java";
21
22          System.out.println(mot.);
23      }
24
25  }
26
27
28
29
30
31
32
33
34
```

```
split(String regex, int limit)                          String[]
startsWith(String prefix)                               boolean
startsWith(String prefix, int toffset)                  boolean
subSequence(int beginIndex, int endIndex)               CharSequence
substring(int beginIndex)                               String
substring(int beginIndex, int endIndex)                 String
toCharArray()                                           char[
toLowerCase()                                           String
toLowerCase(Locale locale)                              String
toString()                                              String
toUpperCase()                                           String
toUpperCase(Locale locale)                              String
trim()                                                  String
valueOf(Object obj)                                     String
valueOf(boolean b)                                      String
valueOf(char c)                                         String
valueOf(char[] data)                                    String
```

Usages

```
run:
BUILD SUCCESSFUL (total time: 0 seco
```

java.lang.String

public String toUpperCase()

Converts all of the characters in this String to upper case using the rules of the default locale. This method is equivalent to toUpperCase(Locale.getDefault()).

Note: This method is locale sensitive, and may produce unexpected results if used for strings that are intended to be interpreted locale independently. Examples are programming language identifiers, protocol keys, and HTML tags. For instance, "title".toUpperCase() in a Turkish locale returns "TİTLE", where 'İ' is the LATIN CAPITAL LETTER I WITH DOT ABOVE character. To obtain correct results for locale insensitive strings, use toUpperCase(Locale.ENGLISH).

Class String (2/5)

```java
//Definition of all the variables used
String name = "Albert Einstein";
int indexOfSpaceCharacter;
String firstName, lastName;

//Code section
indexOfSpaceCharacter = name.indexOf(" ");

firstName = name.substring(0, indexOfSpaceCharacter);

lastName = name.substring(indexOfSpaceCharacter+1);
lastName = lastName.toUpperCase();

name = lastName.concat(" ").concat(firstName);

//Will print "EINSTEIN Albert" on the standard output stream
System.out.println(name);
```

vaapplication1.ScannerTest

- JavaApplication1 (run)

```
run:
EINSTEIN Albert
BUILD SUCCESSFUL (total time: 0 seconds)
```

Class String (3/5)

- Example of a simple concatenation of 3 instances of class Strings:

```
String firstName = "Albert";
String separator = " ";
String lastName = "Einstein";
String name = firstName + separator + lastName;

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

```
run:
Albert Einstein
BUILD SUCCESSFUL (total time: 0 seconds)
```

- Automatic conversion of type:

```
String outputText = "The solution is ";
outputText = outputText + 42;
System.out.println(outputText);
// Set the outputText variable to "The solution is 42"
```

```
run:
The solution is 42
BUILD SUCCESSFUL (total time: 0 seconds)
```

Class String (4/5)

```java
String sentence1 = "Hello.";
String sentence2 = "Hello everyone";

    System.out.print("Test if s1 == s2 : ");
    System.out.println(sentence1==sentence2);
    System.out.println("Test if s1 equals s2 : " + sentence1.equals(sentence2));
```

elloworld.HelloWorld > main >

h Results | Output - HelloWorld (run) | Javadoc

```
run:
Test if s1 == s2 : false
Test if s1 equals s2 : false
BUILD SUCCESSFUL (total time: 0 seconds)
```

Class String (4/5)

```
String sentence1 = "Hello.";
String sentence2 = "Hello everyone";

    System.out.print("Test if s1 == s2 : ");
    System.out.println(sentence1==sentence2);
    System.out.println("Test if s1 equals s2 : " + sentence1.equals(sentence2));
```

elloworld.HelloWorld > main >

h Results | Output - HelloWorld (run) | Javadoc

```
run:
Test if s1 == s2 : false
Test if s1 equals s2 : false
BUILD SUCCESSFUL (total time: 0 seconds)
```

```
String sentence1 = "Hello.";
String sentence2 = "Hello.";

System.out.print("Test if s1 == s2 : ");
System.out.println(sentence1==sentence2);
System.out.println("Test if s1 equals s2 : " + sentence1.equals(sentence2));
```

helloworld.HelloWorld > main >

ch Results | Output - HelloWorld (run) | Javadoc

```
run:
Test if s1 == s2 : true
Test if s1 equals s2 : true
BUILD SUCCESSFUL (total time: 0 seconds)
```

Class String (5/5)

```java
String sentence1 = "Hello.";
String sentence2 = "Hello.";

System.out.print("Test if s1 == s2 : ");
System.out.println(sentence1==sentence2);
System.out.println("Test if s1 equals s2 : " + sentence1.equals(sentence2));
```

helloworld.HelloWorld > main >

ch Results | Output - HelloWorld (run) | Javadoc

```
run:
Test if s1 == s2 : true
Test if s1 equals s2 : true
BUILD SUCCESSFUL (total time: 0 seconds)
```

Class String (5/5)

```java
String sentence1 = "Hello.";
String sentence2 = "Hello.";

System.out.print("Test if s1 == s2 : ");
System.out.println(sentence1==sentence2);
System.out.println("Test if s1 equals s2 : " + sentence1.equals(sentence2));
```

helloworld.HelloWorld > main >

ch Results | Output - HelloWorld (run) | Javadoc

```
run:
Test if s1 == s2 : true
Test if s1 equals s2 : true
BUILD SUCCESSFUL (total time: 0 seconds)
```

```java
String sentence1 = "Hello.";
String sentence2 = new String("Hello.");

System.out.print("Test if s1 == s2 : ");
System.out.println(sentence1==sentence2);
System.out.println("Test if s1 equals s2 : " + sentence1.equals(sentence2));
```

elloworld.HelloWorld > main >

h Results | Output - HelloWorld (run) | Javadoc

```
run:
Test if s1 == s2 : false
Test if s1 equals s2 : true
BUILD SUCCESSFUL (total time: 0 seconds)
```