OOP goals

Simple modeling	An accurate representation of the real world by grouping objects with their properties and actions
Robustness	Easy maintenance and bug detection. Strong typing which results in more robust code (predictable behavior of your code)
Scalability	Adding functionality comes down to establishing new connections with other objects and methods.
Reusability	Available features allowing code reusability. (inher-itance)

OOP building blocks

Class	A blueprint of an object: its properties, behavior and how it interacts with the exterior world.
Attributes & methods	The attributes are the properties of an object while the methods represents its dynamic behavior inside of your program. These attributes and methods define how the object should be accessed, its internal behavior and how it interacts other objects

OOP building blocks (cont)

Objects	An instance of a class. (the class is the type definition and the object is the variable) => all instances of a class share the same fields but have distinct data inside. In addition to its state (properties) and behavior (methods) an object has an identity which distincts it from other objects (alias/ memory address ...)

OOP Paradigms

Abstra-ction	The selection of only useful information about an object for a particular application.
Encaps ulation	Is the ability to control the access to the object's properties and methods, render them either visible (public) or hidden (visible only to internal functions).
Polymo rphisme	The definition of different executions of a methode for different inputs and for different objects

OOP Paradigms (cont)

Heritage	A description of a general\s-pecific relationship between classes. In shorts a subclass has all the properties/ actions of its super class + its own ones + its redefined ones
Compos-ition, Agregation	The creation of a class/object as a collection of other objects

JAVA general infos

JAVA platform = JVM + API JAVA

JVM : execution environment for JAVA apps.
Allows code to be machine independant as it executes inside of a virtual machine which abstracts the specefics of input\output, hardware configuration, and different OSs.
JVM has its own native language : byte code (juts like a real computer has its own instruction set)
The JVM interprets byte code and manages memeory for the programs automatically by its Garbage collector

JAVA API : libraries that abstracts diverse functionalities.

JAVA main function (entry point)

```
public class test{
public static void main(String ar[]){
// code
}
}
```

Array declaration

In java you must allocate the memory for array first by
`type [] name=new type[size]` or `type name[] = new type[size]`
Then if the type is not primitive you need to instantiate each field of the array by
`array_ nam e[i ]=new type(c ons tructor attrib utes)`
Also, we can declare arrays like this
`type array_ nam e[] ={v al1 ,va l2, val3};`

Strings

Declaration:
`String s="s ample text";`
`String s=new String ("sample text")`
Some string methods:
`s.equa ls(s2);` returns 1 is s=s2 and 0 otherwise
`s3=s1.c on cat (s2);` **<=>** `s3=s1+s2;` to concatenate strings

I/O

INPUT in the header of the class file :

```
import java.u til.Sc anner;
```

in the method:

```
Scanner sc = new Scanne r(S ystem.in);
int i = sc.nextInt();
double d = sc.nextDouble();
long l = sc.nextLong();
byte b = sc.nex tBy te(); // etc.
```

I/O (cont)

OUTPUT for single variables:

```
System.ou t.p rin tln (singl);
```

for multiple variables:

```
System.ou t.p rin tln (va r1+ " " +va r2+...);
```

the idea is to convert variables to strings and concatenate them

Garbage Collection

```
//The garbage collector deallocates memory of unreferenced objects
//example
cl obj1=new cl();
System.gc();
//the GC does nothing here since obj 1 is still referenced
cl=null;
System.gc();
//the GC deallo cates the memory previously allocated for obj1 since it's no longer referenced
//This makes the life of a programmer (for example when freeing an array of objects : a single command instead of a loop
```

Properties of static class members

Static attributes are initialized as follows: ints, floats -> **0**, bools -> **false**, references -> **null**

static methods have access only to static methods and attributes of a class (obviously).They can't have use the "**this**" reference since it doesn't make any sense.

```
Class_ nam e.s tat ic_ atr rib u te _name
Class_ nam e.s tat ic_ met hod _ name()
```

Class members visibility

Modificateur	classe	package	Sous-classe	Autres packages
public	oui	oui	oui	oui
protected	oui	oui	oui	non
	oui	oui	non	non
private	oui	non	non	non