Java Syntax overview

Note :
Java does not use indentation, but it is advised to use it to keep your code clear.
Each instruction line must end with ‘;’ (blocks ‘{ }’ are not instruction line)

-Declaration of a variable :

- global variable of a class :

```java
public int value1;
```

Common types : int, float, int[], int[][],…, float[], float[][],…, String, boolean

- local variable

```java
int value1;
```

-Display a message :

```java
System.out.print("Hello World!!!");
System.out.println("Hello World!!!");        // ln is for new line
System.out.println("iteration "+value1);     // concatenation
```

-Operators

addition/subtraction :   + , -
multiplication/division : * , /
increment/decrement :   i++; i--; i+=2; i-=2;
NOT, OR, AND : ! , || , &&   (note : OR uses symbol of keys ‘Alt Gr’+’6’)
equal, inequal, comparisons : ==,!=, >, <, >=, <=

- functions and procedures

```java
// procedure : declared with void
public void myProcedure( int p1, float p2 , … ){
        // instructions
}

// function (can return int, float, int[][],… or any object
public int myFunction( int p1, float p2 , … ){
        // instructions
        return value;  // a function must ends with a return
}
```

-Conditional structures

```java
if (condition){                // condition is a boolean expression (True or False)
        // instructions
}
else{                          // else part is facultative
        // instructions
}


if (condition1){               // multiple conditions
        // instructions
}
else if (condition2){
        // instructions
}
else{
        // instructions
}
```

-Loop structures :

```
        - for loops :
        for (int i=0;i<max;i++){
                // instructions
        }

        for (int i=0;i<max;i+=2){     // increment of 2
                // instructions
        }

        for (int i=max;i>=0;i--){     // decrement is also possible
                // instructions
        }

        int i
        for (i=0;i<max;i++){          // iterator can be declared outside
                // instructions
        }

        - while loops :
        while (condition){            // condition is a boolean expression (True or False)
                // instructions
        }
```

-Structure of a *class* :

```
class MyClass{

        // declaration of global variables
        public int value1;
        public float value2;
        public int[][] table1;

        // object instance constructor
        void MyClass(int v1, … ){            // constructor can use parameters

                // initialization of values
                value1=v1;
                value2=0;
                table1=new int[50][v1];
        }

        // procedure and functions
        public void myProcedure( int p1, float p2 , … ){
                // instructions
        }

        public int myFunction( int p1, float p2 , … ){
                // instructions
                return value;
        }
}
```

-Create object instances (in outside code) :

```
MyClass instance1=new MyClass(4);
MyClass instance2=new MyClass(5);
```

-Read instance value and use methods and functions (from outside code) :

```
int val1=instance1.value1;
int val2=instance2.value1;

int val3=instance1.myFunction(5,6);
```

-Dynamic tables : the ArrayLists

-declaration and initialization :
(common types : *Integer*, *Float*, *String*, *boolean*, and any object including tables)

```
public ArrayList<Integer> list;
list=new ArrayList<Integer>();
```

-get, add, insert, remove, clear :

```
val1=list.get(i);
list.add(val2);
list.insert(index, val2);
list.remove(index);
list.clear();
length=list.size();
```