Understanding Java Keywords

- ❖ There are 50 Java language keywords.
- ❖ We cannot use any of the following as identifiers in your programs.

1) Keywords for data types: (8)

2) Keywords for flow control:(11)

3) Keywords for modifiers:(11)

4) Keywords for exception handling:(6)

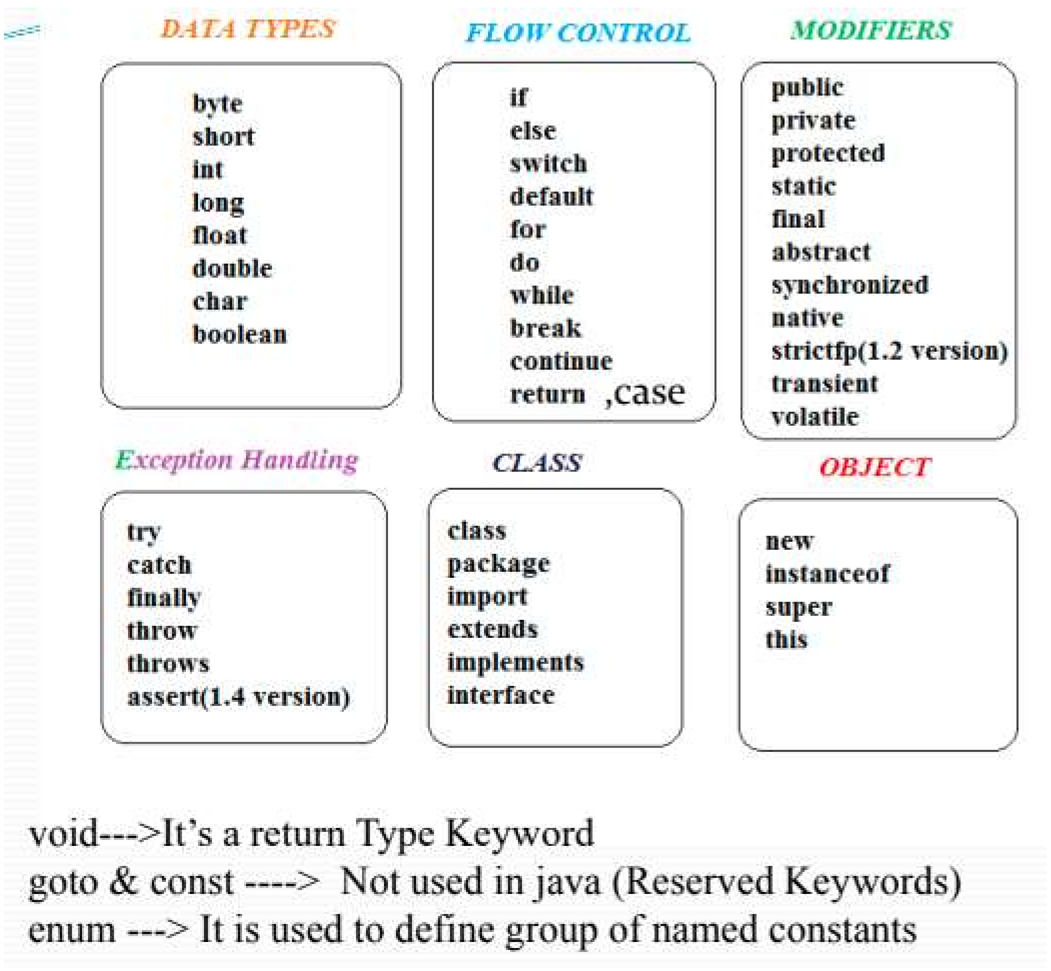
5) Class related keywords:(6)

6) Object related keywords:(4)

7) Void keyword (1)

8) Enum (1)

9) Reserved keywords (2)



DATA TYPES

byte
short
int
long
float
double
char
boolean

FLOW CONTROL

if
else
switch
default
for
do
while
break
continue
return ,case

MODIFIERS

public
private
protected
static
final
abstract
synchronized
native
strictfp(1.2 version)
transient
volatile

Exception Handling

try
catch
finally
throw
throws
assert(1.4 version)

CLASS

class
package
import
extends
implements
interface

OBJECT

new
instanceof
super
this

void--->It's a return Type Keyword
goto & const ----> Not used in java (Reserved Keywords)
enum ---> It is used to define group of named constants

Comments:

- In Java, comments are preceded by two slashes (//) in a line or enclosed between /* and */ in one or multiple lines.

- When the compiler sees //, it ignores all text after // in the same line.

- When it sees /*, it scans for the next */ and ignores any text between /* and */.

Variables:

- Variables are nothing but reserved memory locations to store values.
- Variables are divided into three types

1. Instance variables
2. Static variables
3. Local variables

Instance Variable:

The variables which are declared within the class but outside of any method or block or constructor are called "Instance Variables"

- Instance variables can't be accessed from static area

Static Variable:

- A variable which is declared static is known as static variable.
- Static variables will be created at the time of class loading and destroyed at the time of class unloading

hence the scope of the static variable is exactly same as the scope of the .class file.

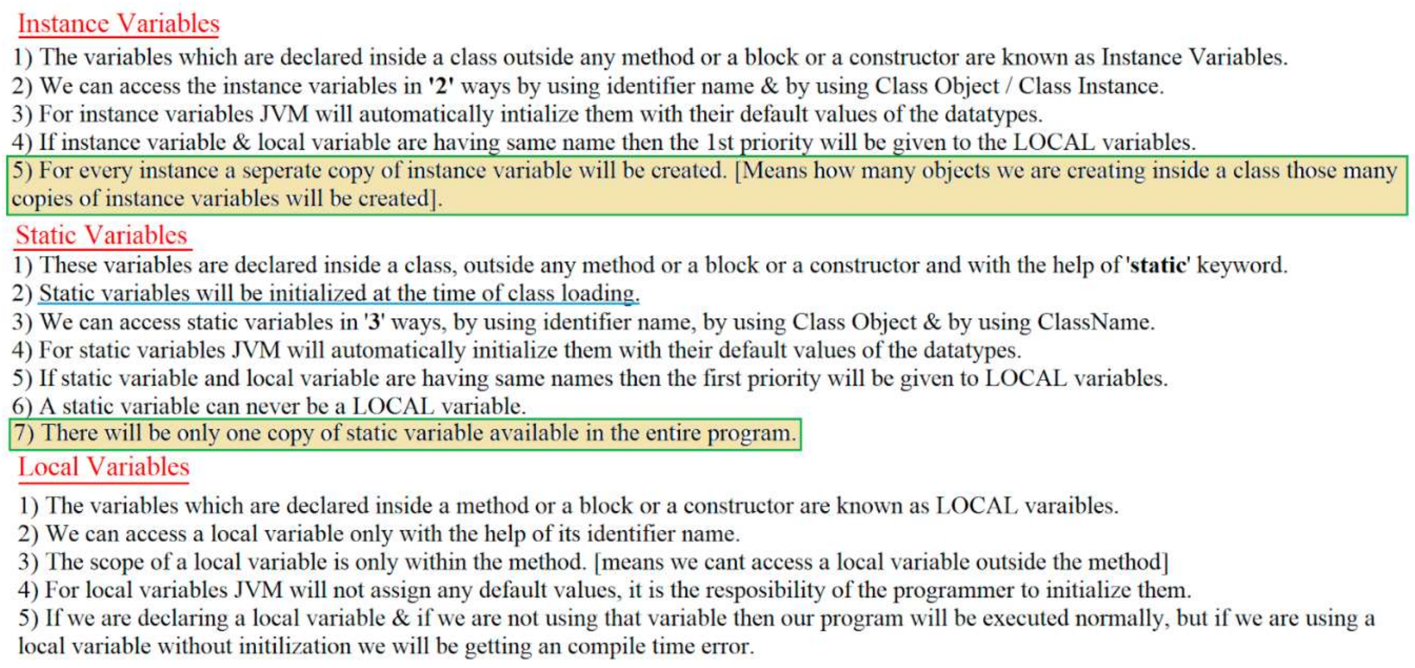
- ❖ Static variables can never be local variables.
- ❖ Static variables can be accessed from both instance and static areas directly.

Local Variable:

- ❖ Variables which are declared inside a method or block or constructors such type of variables are called local variables.
- ❖ The scope of the local variables is exactly same as the scope of the block in which we declared.
- ❖ The only valid modifier for local variables is final.

NOTE

➢For the static and instance variables it is not required to perform initialization explicitly, JVM will provide default values. But for the local variables JVM won't provide any default values compulsory we should perform initialization explicitly before using that variable.



Instance Variables

1) The variables which are declared inside a class outside any method or a block or a constructor are known as Instance Variables.
2) We can access the instance variables in **'2'** ways by using identifier name & by using Class Object / Class Instance.
3) For instance variables JVM will automatically intialize them with their default values of the datatypes.
4) If instance variable & local variable are having same name then the 1st priority will be given to the LOCAL variables.
5) For every instance a seperate copy of instance variable will be created. [Means how many objects we are creating inside a class those many copies of instance variables will be created].

Static Variables

1) These variables are declared inside a class, outside any method or a block or a constructor and with the help of '**static**' keyword.
2) Static variables will be initialized at the time of class loading.
3) We can access static variables in '**3**' ways, by using identifier name, by using Class Object & by using ClassName.
4) For static variables JVM will automatically initialize them with their default values of the datatypes.
5) If static variable and local variable are having same names then the first priority will be given to LOCAL variables.
6) A static variable can never be a LOCAL variable.
7) There will be only one copy of static variable available in the entire program.

Local Variables

1) The variables which are declared inside a method or a block or a constructor are known as LOCAL varaibles.
2) We can access a local variable only with the help of its identifier name.
3) The scope of a local variable is only within the method. [means we cant access a local variable outside the method]
4) For local variables JVM will not assign any default values, it is the resposibility of the programmer to initialize them.
5) If we are declaring a local variable & if we are not using that variable then our program will be executed normally, but if we are using a local variable without initilization we will be getting an compile time error.

Note:

1. **Every number in java by default consider as int.**
2. **Every decimal value in java by default is considered double.**

Arithmetical operations results is given either in int or long format.

Instance Variables

1) The variables which are declared inside a class, outside any method or a block or a constructor are known as Instance variables.
2) We can access the instance variables in ‘2’ ways by using
   1.identifier name
   2.Class Object / Class Instance
3) For instance variable JVM will automatically initialize them with their default values of the datatypes.
4) If instance variable and local variable are having same name, then the 1st priority will be given to the LOCAL variables.
5) For every instance a separate copy of instance variable will be created. (Means how many objects we are creating inside a class those many copies of instance variables will be created).

Static Variables

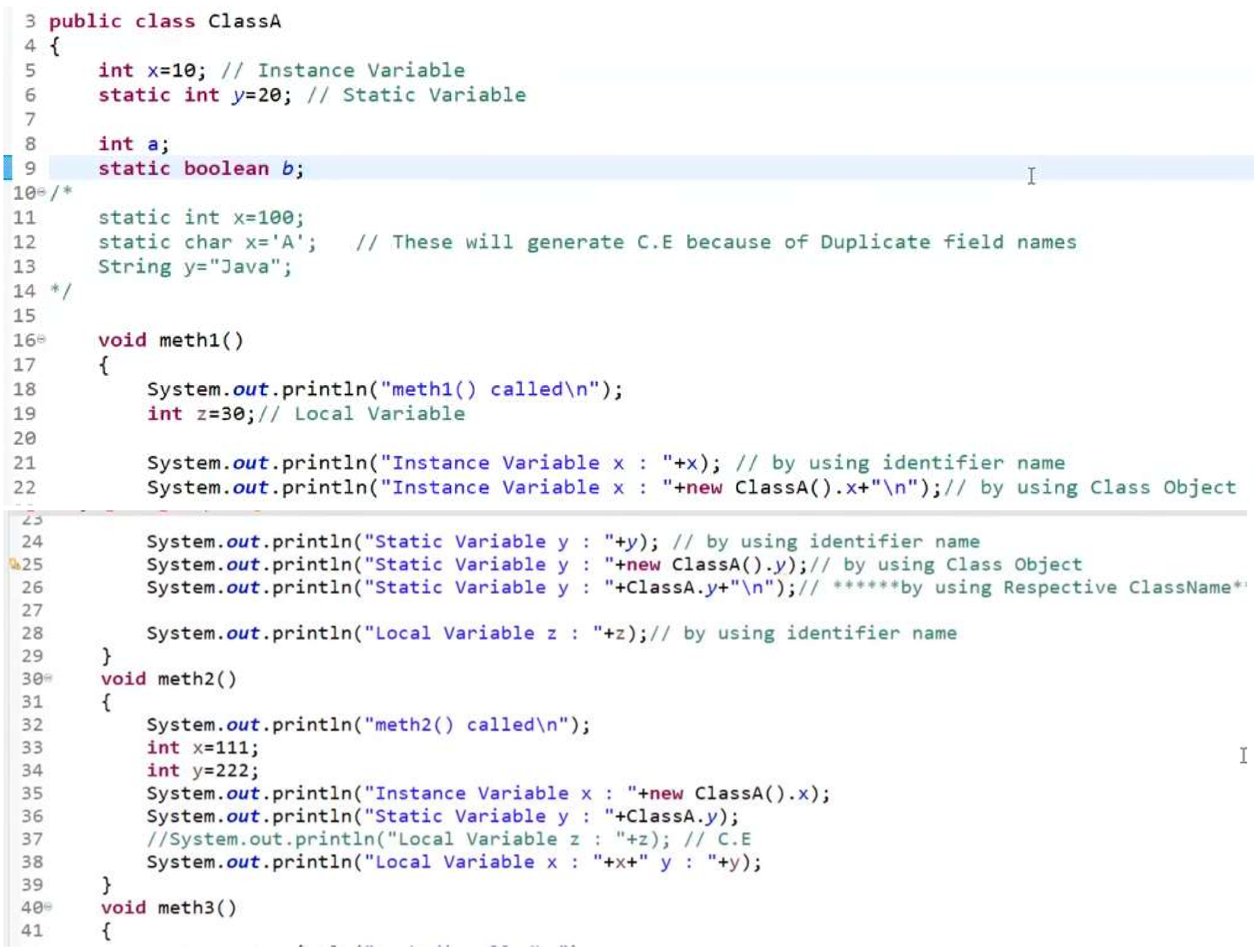
1) These variables are declared inside a class, outside any method or a block or a constructor and with the help of ‘**static**’ keyword.
2) Static variables will be initialized at the time of class loading.

3) We can access static variables in ‘**3**’ ways.
    1. By using identifier name
    2. By using class Object
    3. By using Class Name
4) For static variables JVM will automatically initialize them with their default values of the datatypes.
5) If static variable and local variable are having same names, then the first priority will be given to LOCAL variable.
6) A static variable can never be a LOCAL variable.
7) There will be only one copy of static variable available in the entire program.

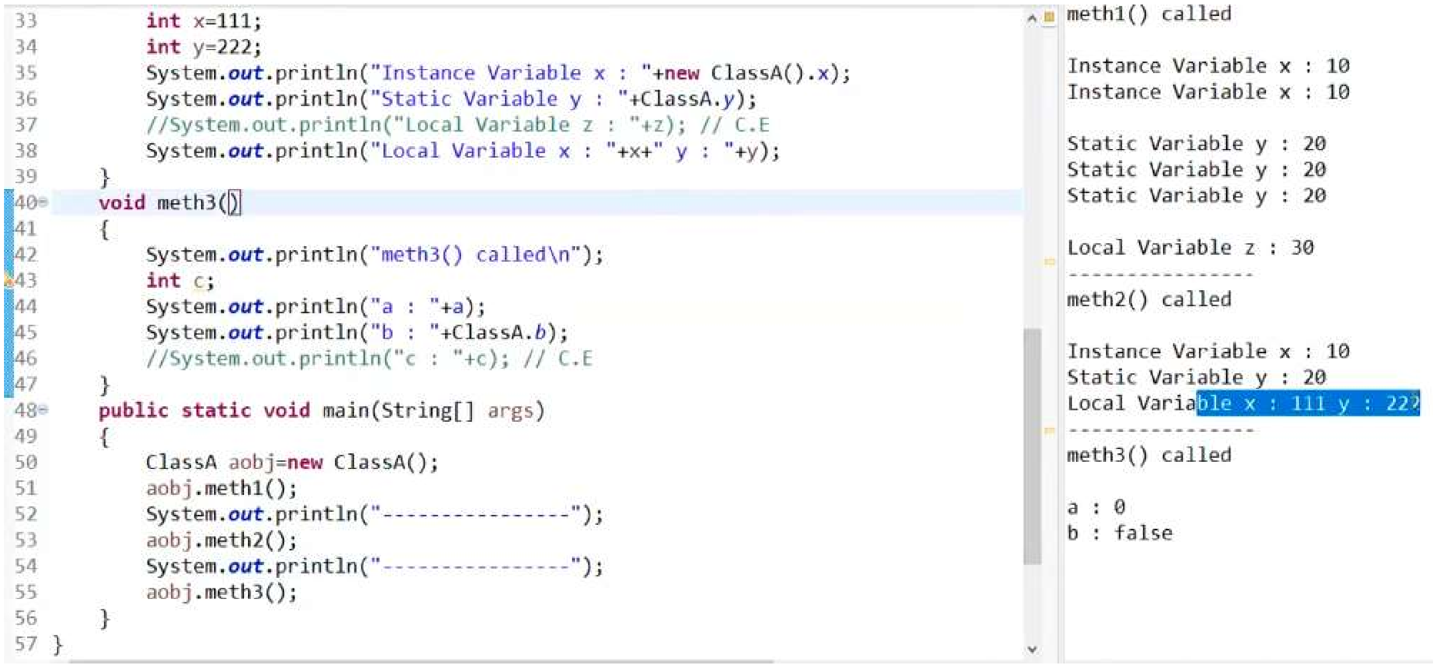
Local Variables

1) The variables which are declared inside a method or a block or a constructor are known as LOCAL variables.
2) We can access a local variable only with the help of its identifiers’ name.
3) The scope of a local variable is only within the method. [means we can’t access a local variable outside the method].
4) For local variables JVM will not assign any default values, it is the responsibility of the programmer to initialize them.

5) If we are declaring a local variable and if we are not using that variable then our program will be executed normally, but if we are using a local variable without initialization, we will be getting a compile time error.



```
public class ClassA
{
    int x=10; // Instance Variable
    static int y=20; // Static Variable

    int a;
    static boolean b;
/*
    static int x=100;
    static char x='A';   // These will generate C.E because of Duplicate field names
    String y="Java";
*/

    void meth1()
    {
        System.out.println("meth1() called\n");
        int z=30;// Local Variable

        System.out.println("Instance Variable x : "+x); // by using identifier name
        System.out.println("Instance Variable x : "+new ClassA().x+"\n");// by using Class Object

        System.out.println("Static Variable y : "+y); // by using identifier name
        System.out.println("Static Variable y : "+new ClassA().y);// by using Class Object
        System.out.println("Static Variable y : "+ClassA.y+"\n");// ******by using Respective ClassName*

        System.out.println("Local Variable z : "+z);// by using identifier name
    }
    void meth2()
    {
        System.out.println("meth2() called\n");
        int x=111;
        int y=222;
        System.out.println("Instance Variable x : "+new ClassA().x);
        System.out.println("Static Variable y : "+ClassA.y);
        //System.out.println("Local Variable z : "+z); // C.E
        System.out.println("Local Variable x : "+x+" y : "+y);
    }
    void meth3()
    {
```



```java
        int x=111;
        int y=222;
        System.out.println("Instance Variable x : "+new ClassA().x);
        System.out.println("Static Variable y : "+ClassA.y);
        //System.out.println("Local Variable z : "+z); // C.E
        System.out.println("Local Variable x : "+x+" y : "+y);
    }
    void meth3()
    {
        System.out.println("meth3() called\n");
        int c;
        System.out.println("a : "+a);
        System.out.println("b : "+ClassA.b);
        //System.out.println("c : "+c); // C.E
    }
    public static void main(String[] args)
    {
        ClassA aobj=new ClassA();
        aobj.meth1();
        System.out.println("----------------");
        aobj.meth2();
        System.out.println("----------------");
        aobj.meth3();
    }
}
```

```
meth1() called

Instance Variable x : 10
Instance Variable x : 10

Static Variable y : 20
Static Variable y : 20
Static Variable y : 20

Local Variable z : 30
----------------
meth2() called

Instance Variable x : 10
Static Variable y : 20
Local Variable x : 111 y : 222
----------------
meth3() called

a : 0
b : false
```