**JAVA DAY 3**

In Java, keywords are the reserved words that cannot be used as identifiers. In total there are 57 keywords in Java. One among them is “Static“.

In Java, static keyword is mainly used for memory management. It can be used with variables, methods, blocks and nested classes. It is a keyword which is used to share the same variable or method of a given class.

\*\*\*we know that for a particular class the number of object we will create, for each object one copy will be created for a particular instance member.

But if the member is static, one copy will be created and this particular copy will be shared by all objects.

In java with in a class we can assign four static member –

1. **static block** - static block is to initialize the static variables. but we can operate other operation also .

**// Java program to demonstrate the use of static blocks**

import java.util.\*;

public class A{

// static variable

static int j = 10;

static int n;

// static block

static {

System.out.println("Static block initialized.");

n = j \* 8;

}

public static void main(String[] args)

{

System.out.println("Inside main method");

System.out.println("Value of j : "+j);

System.out.println("Value of n : "+n);

}

}

**Output:**

Static block initialized

Inside main method

Value of j:10

Value of n : 80

\*\*\*\*static block will execute first before jvm invoke main().

1. **Static variables:** In a program if we use static variable or instance variables, static variables will initialise first because they are initialised by static block and static block is invoked before main().

And instance variables is initialised with in Constructor. constructor is called with in the main when we create object.

import java.util.\*;

public class A

{

// static variable

static int j = n();

// static block

static {

System.out.println("Inside the static block");

}

// static method

static int n() {

System.out.println("from n ");

return 20;

}

// static method(main !!)

public static void main(String[] args)

{

System.out.println("Value of j : "+j);

System.out.println("Inside main method");

}

}

When you execute the above program, it will execute static block and the variable in order as defined in the above program.

**Output:**

from n

Inside the static block

Value of j: 20

Inside main method

\*\*\*In a class we can create n number of static block. They are executed as per their appearance. and we don't require to call them . They are automatically called even before jvm loads the main().

\*\*\*Benefits of static keywords

* only a single memory is created and which can be shared by all the objects.

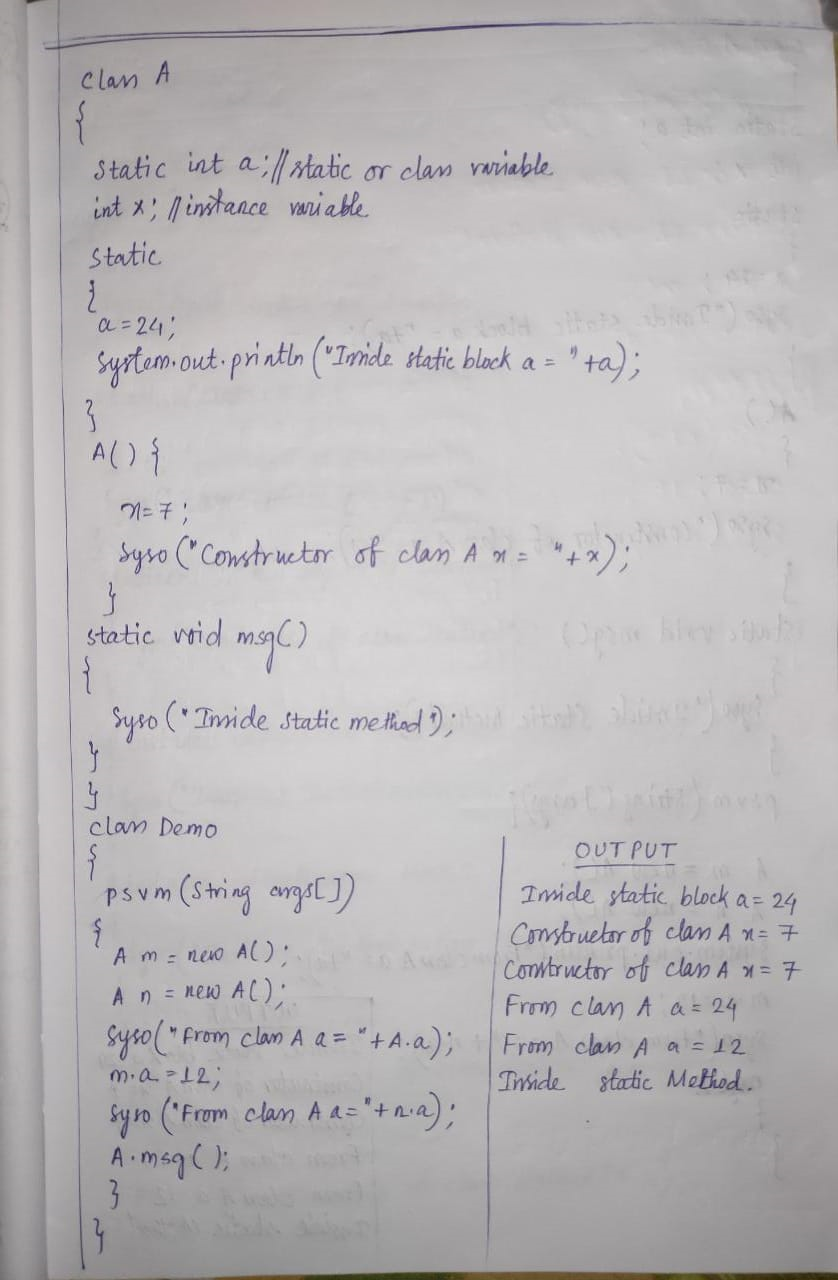
1. **static methods:** A static method is a method that belongs to a class rather than an instance of a class. This means you can call a static method without creating an object of the class. Static methods are sometimes called class methods.

The most common example of a static method is the main( ) method. Methods declared as static can have the following restrictions:

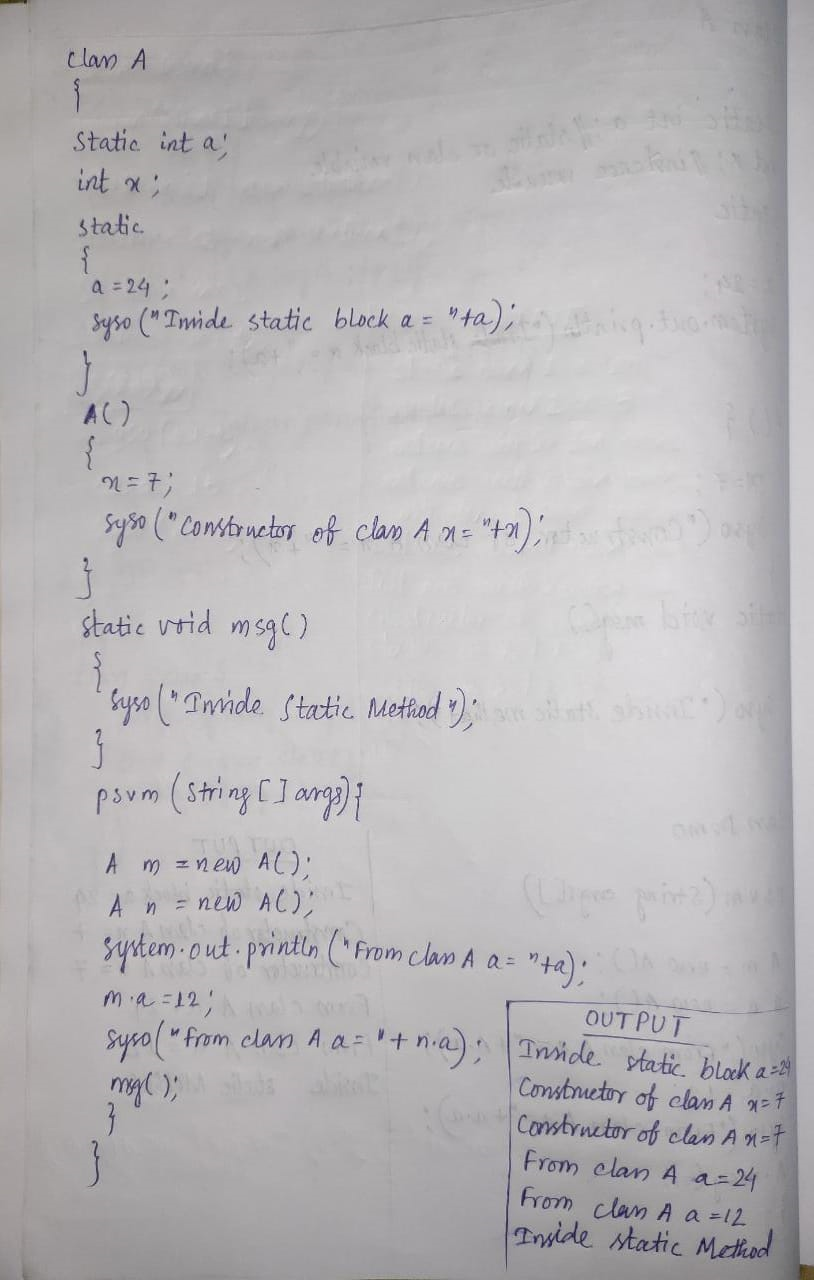
* + They can directly call other static methods only.
  + They can access static data directly.

1. **static class** : Java allows a class to be defined within another class. These are called Nested Classes. Classes can be static which most developers are aware of, henceforth some classes can be made static in Java.

But if we want to access static member of a class, with in static member of another class, then we require **class\_name.member\_name**



\*\*\*If we want to access static member of a class, with in the same class static member we don't require any object, we don't require any class name, we can use them directly.



\*\*\*\*What will happen if I create a class and within class i declare a static block only ... will it run??

class Joy{

static int j;

static

{

j=10;

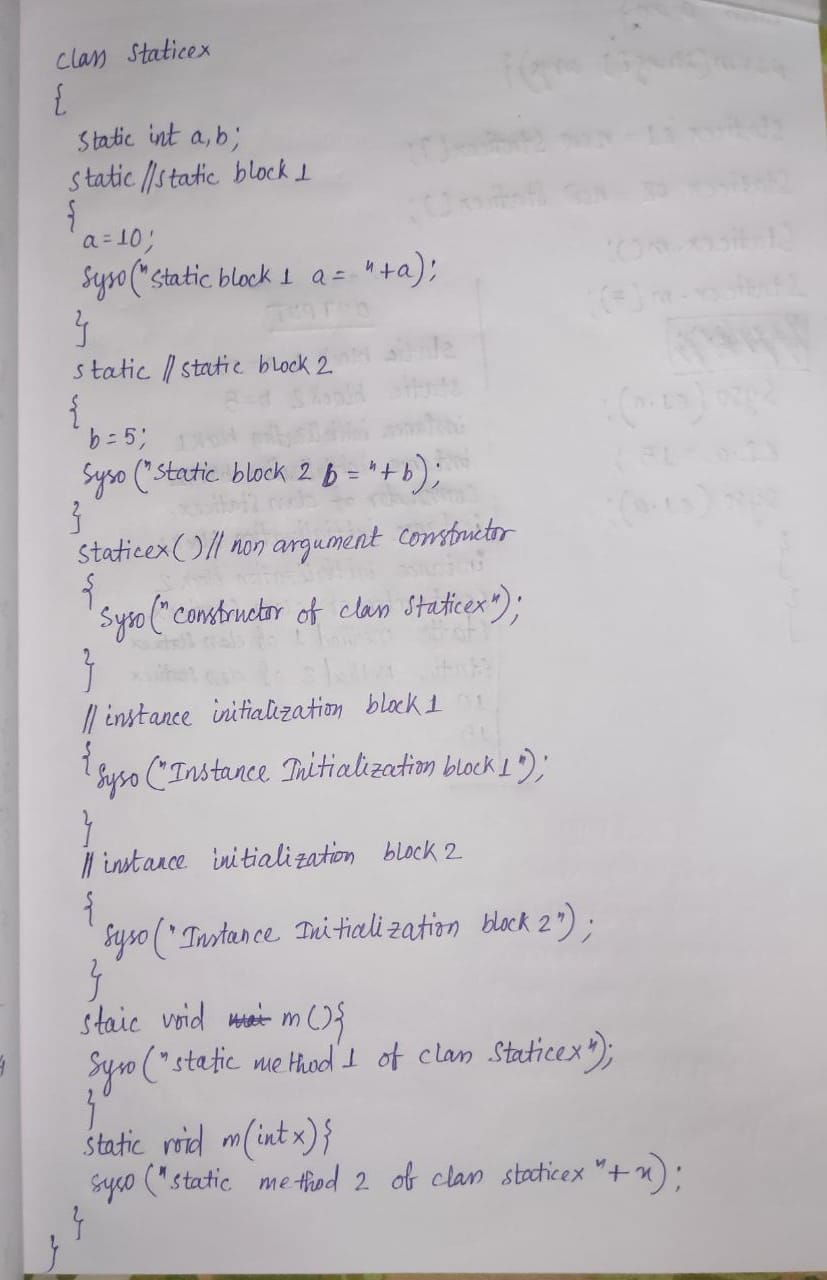
System.out.println(j);

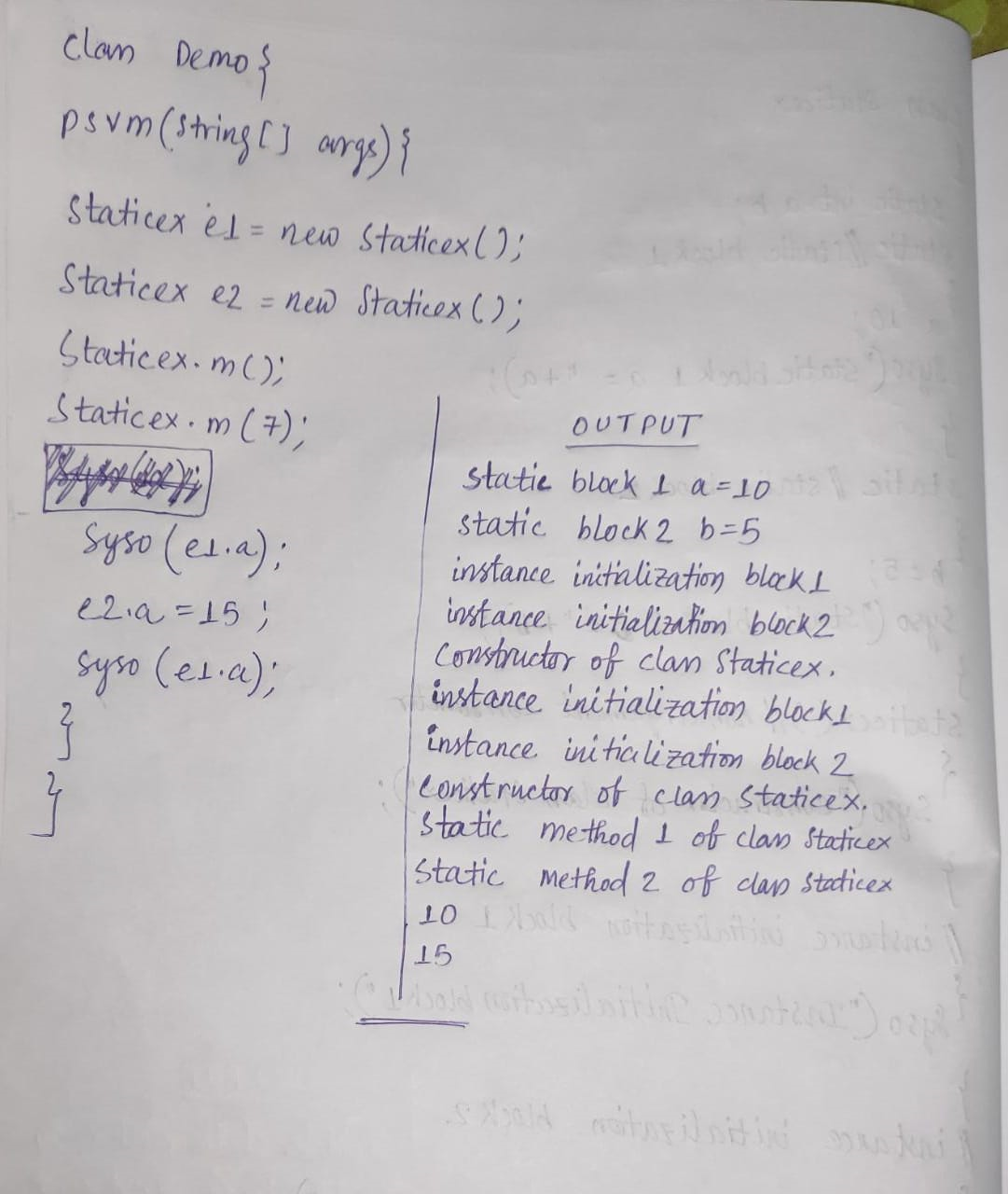
}

}

Ans: It will compile but it will not run because there is no main() in the class.

\*\* Instance initialization block: It is kind of a block used to initialize the instance variables like constructor but if both present then at the time object creation instance initialization block will be invoked first then the Constructor.





\*\* static method overloading is also possible.

NESTED CLASS:

In java, it is possible to define a class within another class, such classes are known as nested classes.

1) static nested class : Nested classes that are declared static are called static nested classes.

This classes only use the static member of the outer class.

**Syntax:**

class OuterClass

{

...

static class NestedClass

{

...

}

}

2) Inner class: An inner class is a non static nested class. It can use all the members of outer class.

**Syntax:**

class OuterClass

{

...

class NestedClass

{

...

}

}

\*\*\*\* To create an object of static nested class we need the name of outer class and also need to create the object of an outer class.

// outer class

class OuterClass

{

// static member

static int outer\_x = 10;

// instance(non-static) member

int outer\_y = 20;

// private member

private static int outer\_private = 30;

// static nested class

static class StaticNestedClass

{

void display()

{

// can access static member of outer class

System.out.println("outer\_x = " + outer\_x);

// can access display private static member of outer class

System.out.println("outer\_private = " + outer\_private);

// The following statement will give compilation error

// as static nested class cannot directly access non-static member

// System.out.println("outer\_y = " + outer\_y);

}

}

}

// Driver class

public class StaticNestedClassDemo

{

public static void main(String[] args)

{

// accessing a static nested class

OuterClass.StaticNestedClass nestedObject = new OuterClass.StaticNestedClass();

nestedObject.display();

}

}

Output:

outer\_x = 10

outer\_private = 30

\*\*\*\*To create an object of inner class we need to make object of outer class and using this object we can make object of the inner class.

InnerClass();

**Example**

// outer class

class OuterClass

{

// static member

static int outer\_x = 10;

// instance(non-static) member

int outer\_y = 20;

// private member

private int outer\_private = 30;

// inner class

class InnerClass

{

void display()

{

// can access static member of outer class

System.out.println("outer\_x = " + outer\_x);

// can also access non-static member of outer class

System.out.println("outer\_y = " + outer\_y);

// can also access private member of outer class

System.out.println("outer\_private = " + outer\_private);

}

}

}

// Driver class

public class InnerClassDemo

{

public static void main(String[] args)

{

// accessing an inner class

OuterClass outerObject = new OuterClass();

OuterClass.InnerClass innerObject = outerObject.new InnerClass();

innerObject.display();

}

}

Output:

outer\_x = 10

outer\_y = 20

outer\_private = 30

\* outer class can't be static

**Difference between static and inner(non-static nested) classes**

Static nested classes do not directly have access to other members(non-static variables and methods) of the enclosing class because as it is static, it must access the non-static members of its enclosing class through an object. That is, it cannot refer to non-static members of its enclosing class directly. Because of this restriction, static nested classes are seldom used.

Non-static nested classes (inner classes) has access to all members(static and non-static variables and methods, including private) of its outer class and may refer to them directly in the same way that other non-static members of the outer class do.