DAY-29

25 July 2023 15:35

WHERE CAN YOU ACCESS STATIC MEMBERS DIRECTLY ?

- INSIDE THE STATIC BLOCK ?

  YES

- INSIDE THE NON STATIC BLOCK

  YES

- INSIDE THE STATIC METHOD

  YES

- INSIDE THE NON STATIC METHOD

  YES

WHERE CAN YOU ACCESS NON STATIC MEMBERS DIRECTLY

- INSIDE THE STATIC BLOCK

  NO

- INSIDE NON STATIC BLOCK

  YES

- INSIDE THE STATIC METHOD

  NO

- INSIDE THE NON STATIC METHOD

  YES

WHY CAN WE ACCESS STATIC MEMBERS IN NON STATIC MEMBERS

WHENEVR OBJECT IS CREATED AND NON STATIC MEMBERS ARE LOADED, ONE COPY OF CLASS MEMORY WILL BE GIVEN TO OBJECT WHICH IS CREATED THAT IS WHY WE CAN ACCESS STATIC MEMBERS DIRECTLY IN NON STATIC METHODS , AND ALSO WE CAN ACCESS STATIC MEBRS USING OBJECT REFERENCE , BUT IS NOT RECOMMENDED

STATIC BLOCKS GENERALLY USED TO INTIALISE THE STATIC MEMBERS OF A CLASS

```
class Test1
{
```

```
	static int x;

	static
	{
		x = 10;  //value of x is 10
	}

	public static void main(String[] args)
	{
		System.out.println("value of x is "+x);
	}

}
```

NON - STATIC BLOCKS GENERALLY USED TO INTIALISE THE NON STATIC MEMBERS OF A CLASS

```
class Test1
{

	 int x;


	{
		x = 100;  //value of x is 100
	}

	public static void main(String[] args)
	{
		Test1 a1 = new Test1();

		System.out.println("value of x is "+ a1.x);
	}

}
```

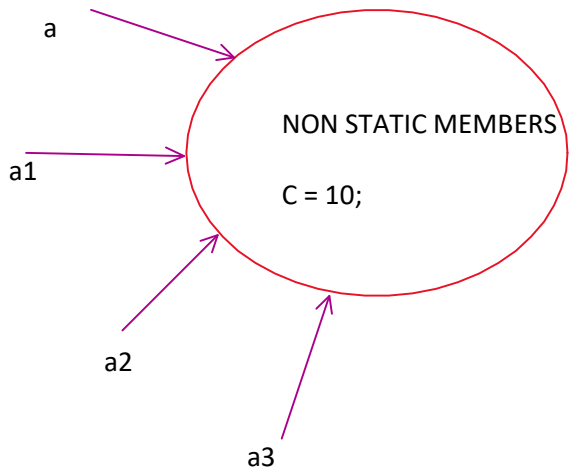
*****DIFFERENCE BETWEEN STATIC BLOCK AND NON STATIC BLOCK

STATIC BLOCK	NON STATIC BLOCK
IT IS EXECUTED DURING CLASS LOADING TIME AUTOMATICALLY	IT IS EXECUTED AUTOMATICALLY DURING OBJECT CREATION TIME
IT IS EXECUTED EVEN BEFORE THE MAIN METHOD	IT IS NOT EXECUTED BEFORE MAIN METHOD
A PARTICULAT STATIC BLOCK WILL BE EXECUTED ONLY ONCE PER CLASS	A PARTICULAR NON STATIC BLOCK IS EXECUTED AS MANY TIMES WE CREATE THE OBJECT OF A CLASS
STATIC BLOCK ARE USED TO INTIALIZE STATIC MEMBER OF A CLASS	NON STATIC BLOCK ARE USED TO INTIALIZE NON STATIC MEMBERS OF A CLASS

STATIC BLOCK WILL BE PRECEED WITH STATIC KEYWORD	NONS TATIC BLOCK WILL NON BE PRECEEDED WITH STATIC KEYWORD

MULITPLE REFERENCE POINTING TO THE SAME OBEJCT

```
class Test1
{

	int c = 10;

	public static void main(String[] args)
	{
		Test1 a = new Test1();

		Test1 a1 = a;

		Test1 a2 = a1;

		Test1 a3 = a2;

		a2.c = 50;

		System.out.println(a2.c);
		System.out.println(a1.c);
		System.out.println(a3.c);
	}

}
```



ASSIGNMENTS

- WHAT IS CLASS ?
- WHAT IS OBJECT OR INSTANCE OF A CLASS ?
- HOW ARE STATIC AND NON STATIC VARIABLES DIFFERENCE FROM LOCAL VARIABLE ?
- WHAT IS STATIC VARIABLE ? EXPLAIN ?
- WHAT IS NON STATIC VARIABLE ? EXPLAIN
- HOW TO ACCESS STATIC MEMBERS ? EXPLAIN
- HOW TO CREATE INSTANCE OF A CLASS ?
- WHAT IS DIFFERENCE BETWEEN STATIC AND NON STATIC MEMBERS ?
- WHEN TO USE STATIC VARIABLES AND WHEN TO USE NON STATIC VARIABLES ? EXPLAIN
- WHAT ARE REFERENCE VARIABLES ?
- EXPLAIN BLOCKS AND NON STATIC BLOCKS

WHAT HAPPENS IF YOU ASSIGN A NULL VALUE TO A REFERENCE VARIABLE ?

```
class Test1
{
```

```
    double x = 9.16;

    public static void main(String[] args)
    {
        Test1 a1 = null;

        System.out.println(a1.x);
    }

}
```

OUTPUT : NullPointerException

CONSTRUCTORS

- THESE ARE SPECIAL MEMBER FUNCTIONS (METHODS) HAVING THE SAME AS THAT OF THE CLASS NAME

SYNTAX :

```
ACCESS_MODIFIER CLASS_NAME(OPTIONAL PARAMEMTERS)
{
    //CODE TO EXECUTE

}
```

NOTE :

ACCESS MODIFIERS

THERE ARE 4 TYPES OF ACCESS MODIFIERS , THEY ARE

1. PUBLIC
2. PROTECTED
3. PRIVATE
4. DEFAULT

PROPERTIES OF CONSTRUCTORS

- CONSTRUCTORS ARE USED TO INTIALIZE MEMBERS OF AN OBEJCT
- CONSTRUCTORS WILL NOT HAVE ANY RETURN TYPE
- CONSTRUCTORS WILL NOT HAVE ANY MODIFIERS (STATIC AND NON STATIC)

```
ACCESS_MODIFIER CLASS_NAME()
{
   //CODE TO EXECUTE
}
```

ACCESS MODIFERS CAN BE PUBLIC/PRIVATE/PROTECTED/DEFAULT BUT, WIDELY USED IS PUBLIC FOR CONSTRUCTORS

- THE CONSTRUCTORS ARE EXECUTED AUTOMATICALLY WHEN WE CREATE OBJECT/INSTANCE OF CLASS. WHENVER JVM ENCOUNTERS OBEJCT CREATION STATEMENT, AN OBJECT WILL BE CREATED, ALL BIB STATIC MEMBERS WILL BE LOADED AND IF THERE IS ANY CONSTRUCTOR DEFINED BY THE USER , IT WILL BE AUTOMATICALLY EXECUTED.

```
class Test1
{

	public Test1()
	{
		System.out.println("inside the constructor");
	}

	public static void main(String[] args)
	{
		Test1 a1 = new Test1();
	}

}
```

```
class Test1
{
	static
	{
		System.out.println("inside the static block");
	}

	{
		System.out.println("inside the non-static block");
	}

	public Test1()
	{
		System.out.println("inside the constructor");
	}

	public static void main(String[] args)
	{
		System.out.println("inside the main method");

		Test1 a1 = new Test1();
	}

}
```

OUTPUT :

```
inside the static block
inside the main method
inside the non-static block
inside the constructor
```

PRIORITY

1. STATIC BLOCK
2. MAIN METHOD
3. NON STATIC BLOCK
4. CONSTRUCTOR

TYPES OF CONSTRUCTORS

THERE ARE TWO TYPES, THEY ARE

1. USER DEFINED CONSTRUCTORS
2. DEFAULT CONSTRUCTORS

USER DEFENIED CONSTRUCTORS

1. PARAMETERIZED
2. NON PARAMTERIZED

DEFAULT CONSTRUCTORS

- THIS IS ALSO CALLED AS PREDEFINED CONSTRUCTORS OR COMPILER DEFINED CONSTRUCTORS OR COMPILER DEFINED CONSTRUCTOR
- COMPILER DEFINED CONSTRUCTOR WILL ALWAYS BE NOT ARGUMENTED OR WITHOUT ANY PARAMETER
- THE USER DEFINED CONSTRUCTOR CAN BE WITH OR WITHOUT PARAMETERS
- EACH TIME WE CREATE AN OBEJCT OF A CLASS , IT IS MANDATORY FOR THE CONSTRUCTOR TO GET EXECUTED
- SO, IF A CLASS IS NOT HAVING USER DEFINED CONSTRCUTOR TO INTIALIZE NON STATIC MEMBERS OF A CLAS, IT WILL HAVE COMPILER DEFINED CONSTRCUTOR FOR SURE.

COMPILER DEFINED CONSTRCUTOR

```
class Test1
{
	int x;
	double y;

	public static void main(String[] args)
	{

		Test1 a1 = new Test1();

		System.out.println(a1.x);
		System.out.println(a1.y);

	}

}
```