

```
1 class Calci
2 {
3
4     public static void main(String[] args)
5     {
6         int sum=add(10,30);
7         System.out.println("Addition "+sum);
8         int subtract=sub(100,50);
9         System.out.println("Subtraction "+subtract);
10        int prod=mul(10,5);
11        System.out.println("Multiplication "+prod);
12        double quotient=div(134.9,7.2);
13        System.out.println("Division "+quotient);
14        int rem=mod(12,9);
15        System.out.println("Modulus "+rem);
16    }
17
18    public static int add(int a,int b)
```

```
16  
17 public static int add(int a,int b)  
18 {  
19     int res=a+b;  
20     return res;  
21 }  
22 public static int sub(int x,int y)  
23 {  
24     int res=x-y;  
25     return res;  
26 }  
27 public static int mul(int a ,int b)  
28 {  
29     int res=a*b;  
30     return res;  
31 }  
public static double div(double a, double b)
```

```
26  
27 public static int mul(int a, int b)  
28 {  
29     int res=a*b;  
30     return res;  
31 }  
32 public static double div(double a, double b)  
33 {  
34     double res=a/b;  
35     return res;  
36 }  
37 public static int mod(int x, int y)  
38 {  
39     int res=x%y;  
40     return res;  
41 }
```

onice)
ock
nd
an

```
1 class Demo1
2 {
3
4     public static void main(String[] args)
5     {
6         System.out.println("Main Starts");
7         //Atleast one method should call in main() method
8         //same method we can call for n no of times
9         demo();
10        //demo1();CTE
11        System.out.println("Main Ends");
12    }
13
14    public static void demo()
15    {
16        System.out.println("Demo Starts");
17        System.out.println("Demo Ends");
18    }
19 }
```

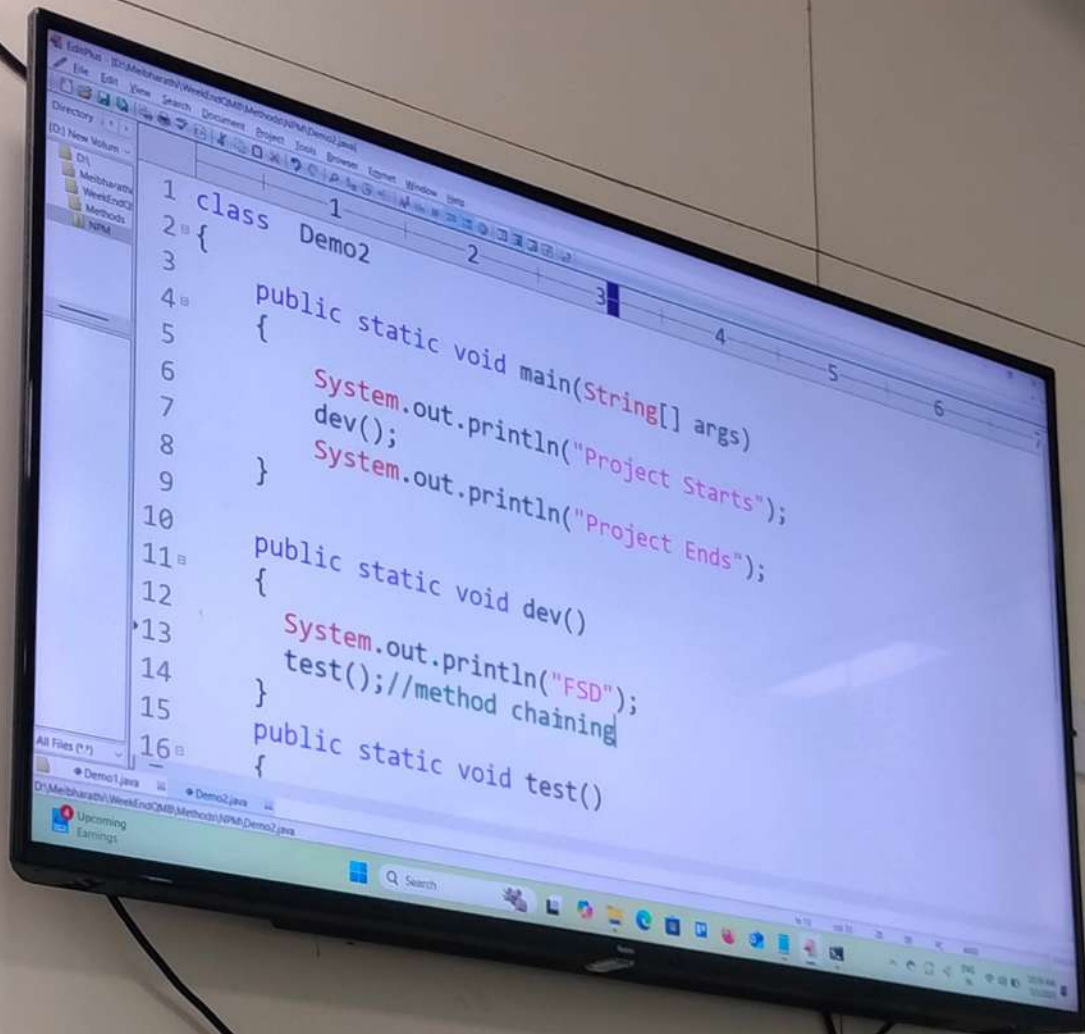

berg.
final, default.
final, (synchronised)
to the method block
PDT → Updefined
PDT class

```
1 class P1
2 {
3
4     public static void main(String[] args)
5     {
6         method(10,20); //actual args
7         method('A','a'); //type casted
8     }
9     public static void method(int a ,int b) //formal args
10    {
11        System.out.println("int a = "+a);
12        System.out.println("int b = "+b);
13    }
14 }
```

```
1 class P2
2 {
3
4     public static void main(String[] args)
5     {
6         mobileDetails("OPPO", "A15", 15000, 64);
7         mobileDetails("Iphone", "15ProMax", 110000, 128);
8         mobileDetails("RedMi", "8A Dual", 10000, 64);
9     }
10
11     public static void mobileDetails(String brand, String model, double
12     {
13         System.out.println("Mobile Brand : "+brand);
14         System.out.println("Mobile model : "+model);
15         System.out.println("Mobile Price : "+price);
16         System.out.println("Mobile storage: "+storage+"gb");
17     }
18 }
```

```
1 class Demo3
2 {
3
4     public static void main(String[] args)
5     {
6         System.out.println("Amount to pay : "+shopping());
7     }
8     public static double shopping()
9     {
10         int qty=3;double price=499,discount=10;
11         double cost=qty*price;
12
13         if (cost > 1000)
14         {
15             double totalBill=cost - (cost * 10/100);
16             return totalBill;
17         }
18     }
19 }
```

```
4 {  
5     System.out.println("Amount to pay : "+shopping());  
6 }  
7 public static double shopping()  
8 {  
9     int qty=3;double price=499,discout=10;  
10    double cost=qty*price;  
11  
12    if (cost > 1000)  
13    {  
14        double totalBill=cost - (cost * 10/100);  
15        return totalBill;  
16    }  
17    else  
18        return cost;  
19 }
```

```
1 class Demo2
2 {
3
4     public static void main(String[] args)
5     {
6         System.out.println("Project Starts");
7         dev();
8         System.out.println("Project Ends");
9     }
10
11     public static void dev()
12     {
13         System.out.println("FSD");
14         test();//method chaining
15     }
16     public static void test()
17     {
```

default.
, Synchronized
method block
T → User defined
class

```
4 {  
5  
6     System.out.println("Project Starts");  
7     dev();  
8     System.out.println("Project Ends");  
9 }  
10  
11 public static void dev()  
12 {  
13     System.out.println("FSD");  
14     test();//method chaining  
15 }  
16 public static void test()  
17 {  
18     System.out.println("S/W Testing");  
19 }
```

```
1 class Details
2 {
3
4     public static String data()
5     {
6         String name="Dilip";int id=123;long cno=8529631478L;
7         return "Name : "+name+"\nID : "+id+"\nCno : "+cno;
8     }
9     public static void main(String[] args)
10    {
11        System.out.println("*****Details*****");
12        System.out.println(data());
13        System.out.println("-----");
14        String details=data();
15        System.out.println(details);
16    }
}
```


Parameterized Method

→ [AM] [MM] void method name (formal args)
{ }

→ [AM] [MM] Datatype method name (formal args) { return }

Non Parameterized Method.

→ [AM] [MM] void method name () { }

→ [AM] [MM] Datatype method name () { return }

Syntax:

Access Modified	Memory Modified
-----------------	-----------------

{

}

→ Method

* Method &

* Method &

* Method &

→ Combin
and Metho

→ Methods.

→ block of memory

↳ Set of Instructions

→ We can create Multiple Method blocks.

↳ Inside the class block.

→ In Main() method.

↳ at least one method.

Syntax



→ Method Terminology

* Method Signature

* Method declaration

* Method definition

→ combined with method declaration and Method block (body, implementation)

Access Modifier → Access the members

→ public, private, protected, default

Memory Modifier → static, nonstatic, final, synchronized
Behavior of the method

return type → It returns the value back to the method block

→ void → Data type
→ Primitive
→ Non-Primitive

*A block of code, which is used to ex
*Basically initial methods, we write
will get executed together in the bl
*These instructions can be executed
method block.

Syntax:

```
[Access Modifier] [Memory Modifier] [return type] [method name (in formal)]  
{  
    Method statement  
    Return statement  
}
```

→ Methods.

→ block of memory

Perform the specific task.

→ Set of Instruction

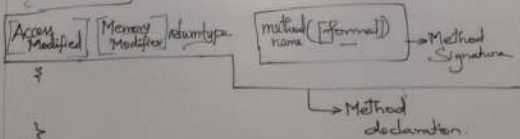
→ We can create multiple Method blocks.

→ Inside the class block.

→ In Main() method.

→ at least one method.

Syntax



Method Terminology

* Method Signature

* Method declaration

* Method definition

→ Combined with method declaration and Method block (body, implementation)

Access Modifier → Access the members.

→ public, private, protected, default

Memory Modifier → static, nonstatic, final, synchronized

Behavior of the method

return type → It returns the value back to the method block

→ void → Data type → Primitive (PDT) → User defined (UPDT) can

Methods :

- * A block of code, which is used
- * Basically initial methods, which will get executed together in
- * These instructions can be executed in a method block.

Syntax:

```
[Access Modifier] [Memory  
Arguments/Parameters]]  
{  
    Method statement  
}
```

Methods :

- *A block of code, which is used to execute a specific task.
- *Basically initial methods, we write instructions and all the instruction will get executed together in the block.
- *These Instruction can be executed multiple time as they are saved in the method block.

Syntax:

```
[Access Modifier] [Memory Modifier] returnType methodName([Formal  
Arguments/Parameters])  
{  
    Method statement  
    Return statement  
}
```

members
ed, default.
final, synchronized
the method block
DT variable

Method Terminology :
-->Method Signature
-->Method declaration
-->Method Definition

Modifier :

Modifier are the keyword which specify a special property of the block of code.

-->We Have two types of Modifier.

- *Access Modifier
- *Memory Modifier

Access Modifier:

-->Access Modifier specifies who can access the code.
(or)

-->Access Modifier is used to accessibility of all the Members in our Class.

We have public, protected, private, default.
public : open to all, anyone can access and execute the code.
protected: We can access this code only within the project.
private: the program can be used only within the class.
[default] : Not written any keywords is called default access, here we can say that the access is equal to protected Keyword Access .

Memory Modifier:
-->Memory Modifier is an Characteristic of the method or Behavior of the Method.
-->If static keyword is written then we call it as static method , else non static method.

default.
(Synchronized)
method block
-> User defined
class

The value to be returned and the return type should be of same type or properly type casted.

Parameters Intro:

Basically in methods we are saving a small set of program so that by passing different inputs and by using them at different times, we can minimize the complexity of a program.

Parameters:

Arguments are input that we can pass for executing a method. Basically arguments is written of two types.

*Non Parameter Method:

It doesn't have Any formal arguments.

*Parameterized Method:

It have formal arguments in parameter.

me.
default.
(synchronized)

Method block
(synchronized)

Memory Modifier:

-->Memory Modifier is an Characteristic of the method or Behavior of the Method.

-->If static keyword is written then we call it as static method , else non static method.

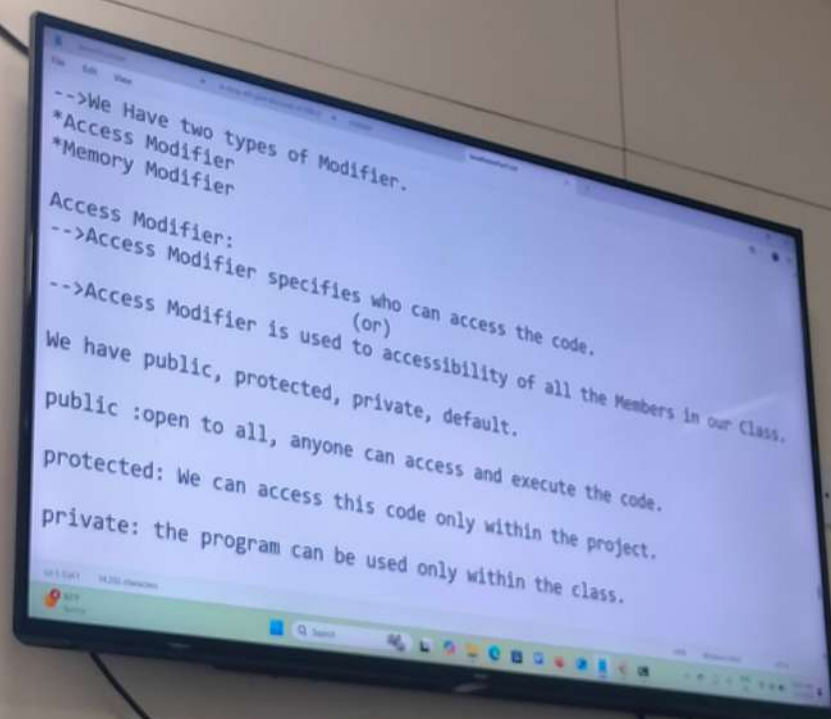
Return Type:

It is the data type of value we can expect as an output from a method.

If we don't want to return any value the return type is void. In void we don't have return statement.

Note :

The value to be returned and the return type should be of same type or properly type casted.



Return Type:

It is the data type of value we can expect as an output from a method.

If we don't want to return any value the return type is void. In void we don't have return statement.

Note :

The value to be returned and the return type should be of same type or properly type casted.

Parameters Intro:

Basically in methods we are saving a small set of program so that by passing different inputs and by using them at different times, we can minimize the complexity of a program.

Parameters:

Arguments are input that we can pass for executing a method

monica

black

black

black