Lambda Expression

→ lambda Expressions are anonymous functions. These functions do not need a name or a class to be used. Lambda expressions are added in Java 8. Lambda Expression express instances of functional interfaces.

Syntax →

```
( ) → Expression ;
```

Params · Arrow token · function body

```
Calculator C1 = (a,b) → {
        return a+b;
};
```

2 Para

use paranthisis for multi-statements.

1 · 2

zero Parameter

```
Shape s1 = ( ) → System.out.Println("I am Square");
```

only one statement so run without Paranthisis.

Can remove Brackets ( ) for one Parameter

```
Number n1 = n → {
        if (n%2==0){
                sout("Even");
        else{
            sout ("ODD");
        }
};
```

Collections

Can sort In Increse

```
Tree<Integer,Integer> map = new TreeMap <>();
Comparator <Integer> Comp1 = (a,b) → b-a;
```

↳ reverse sort.

② Pass Comp1 for reverse & then run

```
//  incr  1P – 2 Parameter
//        2P – 1 Parameter

map.put (7,1);
map.put (77,1);
map.put (177,1);
map.put (17,1);
```

① Sout(map); → Increseing

```
Tree <Integer,Integer> map = new TreeMap <>(
(a,b) → b-a);
```

* ↳ we can Direct Pass this for sort in reverse order

Do It

* Comparable ? find functional Interface ?
if yes Implement it

→ Priority Queue +
custom class →
sort using lambda

* why this work Because Comparator is a functional Interface.

```
ArrayList <Integer> list = new ArrayList <>();
list.forEach ( );
```

→ use consumer Interface is also a functional Interface

read all this
* serialize / threading
* consumer