The Lambda expression is **nameless function/anonymous function**. That means the function which does not have name, return type ,access modifiers.

Syntax:

([par1,par2,etc…]) ->{

Statements;

}

1. A lambda expression have zero or more than zero parameters. If zero number of parameters available then we have to use empty parathensis().
2. The lambda expression body can contain multiple statements. If more than one statements present then we have to enclose inside with in curly braces. If one statement present then curly braces are optional.

**Syntax to invoking the lambda expression:**

funtionalInterfaceName var-name= Lambda expression;

var-name.abstractmethodname();

**Example with out lamba expression:**

1. Interface A{

Public void sum(int a,int b);

}

Class Demo implements A{

Public void sum(int a, int b)

{

System.out.println(“ sum:”+ (a+b));

}

}

Public class Test{

Public static void main(String [] args){

A a1=new Demo();

A1.sum(2,3);

}

}

Output:

---------

Sum: 5.

**Example with Lambda Expression:**

**In**terface A{

Public void sum(int a,int b);

}

Class Test{

Public static void main(String [] args){

A a1=(a,b) 🡪 System.out.println(“Sum:”+a+b);

A1. Sum(2,3);

}

}

Output:

---------

Sum:5

**Usages Of Lambda Expression in Multi Threading:**

**1.Without Lambda Expression**

Class MyThread implements Runnable{

public void run(){

System.out.println("This is customThread");

}

}

class sample {

public static void main(String arg[]){

MyThread m1=new MyThread();

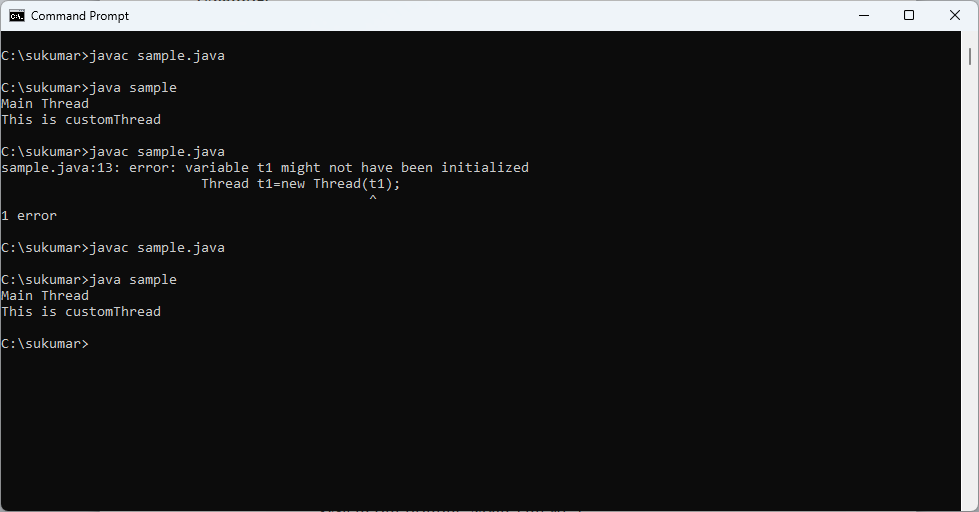
Thread t1=new Thread(m1);

System.out.println("Main Thread");

t1.start();

}

}



**2.With Lambda Expression:**

**class** suku {

**public** **static** **void** main(String arg[]){

Runnable m1=()->

System.***out***.println("This is customThread");

Thread t1=**new** Thread(m1);

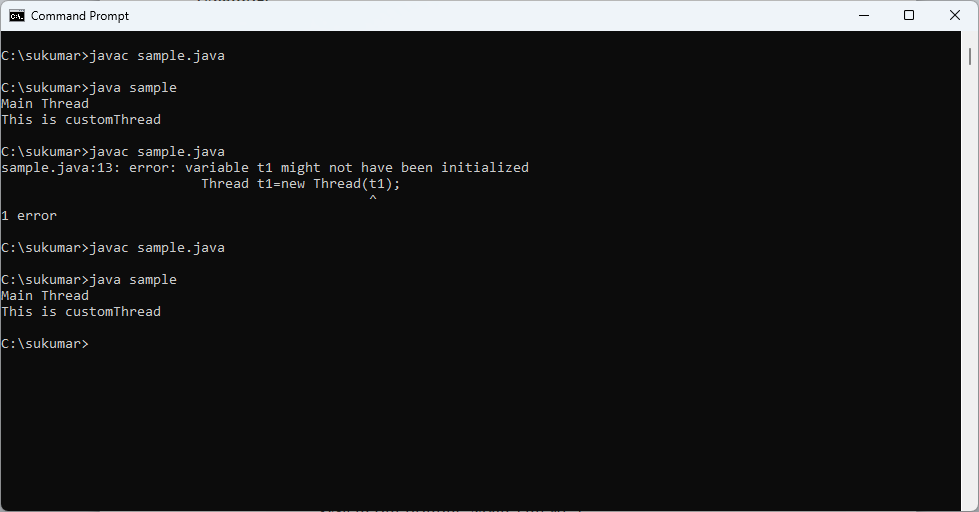
System.***out***.println("Main Thread");

t1.start();

}

}

**Output:**

****

**Uses of Lambda Expression in Collection:**

**1.with out Lambda Expression**

**import** java.util.Comparator;

**import** java.util.TreeSet;

**class** Abc **implements** Comparator

{

**public** **int** compare(Object o1,Object o2)

{

Emp e1=(Emp)o1;

Emp e2=(Emp)o2;

**if** (e1.eno > e2.eno) {

**return** -1;

}

**else**

{

**return** 1;

}

}

}

**class** Emp{

String ename;

**int** eno;

**public** Emp(String ename,**int** eno) {

**this**.ename=ename;

**this**.eno=eno;

}

**public** String toString()

{

**return** **this**.ename+" "+**this**.eno;

}

}

**public** **class** Practice {

**public** **static** **void** main(String[] args) {

TreeSet t1=**new** TreeSet(**new** Abc());

t1.add(**new** Emp("suku",7));

t1.add(**new** Emp("sv",8));

t1.add(**new** Emp("sumasuha",9));

System.***out***.println(t1);

}

}

Output:

[sumasuha 9, sv 8, suku 7]

**With Lambda Expression:**

**import** java.util.Comparator;

**import** java.util.TreeSet;

**class** Emp{

String ename;

**int** eno;

**public** Emp(String ename,**int** eno) {

**this**.ename=ename;

**this**.eno=eno;

}

**public** String toString()

{

**return** **this**.ename+" "+**this**.eno;

}

}

**public** **class** suku {

**public** **static** **void** main(String[] args) {

Comparator c1=(o1,o2)->

{

Emp e1=(Emp)o1;

Emp e2=(Emp)o2;

**if** (e1.eno > e2.eno){

**return** -1;

}

**else**

{

**return** 1;

}

};

TreeSet t1=**new** TreeSet(c1);

t1.add(**new** Emp("suku",7));

t1.add(**new** Emp("sv",8));

t1.add(**new** Emp("sumasuha",9));

System.***out***.println(t1);

}

}

**Output:**

[sumasuha 9, sv 8, suku 7]

**Advantages By Lambda Expression:**

By lambaexpression, code reduced.

When I used lambda expression, we can completely remove the functional interface implementation class. So code can be reduced.