Lamda Expression

Lamda Expression:

Ealier for a function few things were mandatory,such as

1>Access Modifier.

2>Return statement.

3>Name
Eg:

```
public void test(){

System.out.println("Hello');

return;

}
```

But for a Lamda expression these things are not required,so the above function is written as:

```
()->{ System.out.println("Hello'); }
```

The Operator that is used to denote the lamda expression is ->

Now if have a single statement then parenthesis are optional so no need to write that as well.

```
()->System.out.println("Hello');
```

But if we have more than on statement within the function then we need to give parenthesis as well.

Now the next question that come to mind is were to use this lamda expression, **so lamda expression can only be used for functional interface**,In case you don't know what are functional interface please read Functional interface post.

So if we have a Functional interface ,then their we can use Lamda expression.

```
package com.technicalstack.basic;

interface Interf{

public void f1();
}

public class Test {

public static void main(String[] args) {
Interf i = ()->System.out.println("Lamda Expression");
i.f1();
}
```

}
Here Interf is a functional interface which has f1() as a abstract function,

The implementation of this abstract function is defined in main class,

so Interf i = ()->System.out.println("Lamda Expression");

is the way lamda expression can we written as no name,no modifier and no return type is required and has only one statement in it.

The important thing to note here is that to hold the output of lamda expression the corresponding funtional interface reference is requried, here **Interf i ,**

When we run the above code the output is Lamda Expression