Using Functional Interface

We saw what [Functional Interfaces](http://data-structure-learning.blogspot.com/2015/06/functional-interfaces-java-8.html) are in previous post. I this post we will see how to use them using new syntax i.e. Lambda Operator.

First let us see some different ways to start a Thread in Java.

Here we are using Runnable Interface as argument.

Thread t1=**new** Thread(**new** Runnable(){

@Override

**public** **void** run() {

**for** (**int** i = 0; i <= 50; i++) {

System.out.println(i);

}

}

});

t1.start();

Second, we can implement the Runnable Interface in class and Override run() method.

**public** **class** UseRunnableInterface **implements** Runnable {

@Override

**public** **void** run() {

**for** (**int** i = 0; i < 10; i++) {

System.out.println(Thread.currentThread().getName()+" "+i);

**try** {

Thread.sleep(100);

} **catch** (InterruptedException ie) {

ie.printStackTrace();

}

}

}

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

UseRunnableInterface run1 = **new** UseRunnableInterface();

Thread t1 = **new** Thread(run1);

t1.start();

}

}

Third, we can extend the Thread class and Override run method. We don’t use this method unless we don’t have any good reason to do so. Because extending one class thread limits our extending capability as we can extend only one class in Java.

**public** **class** UseThreadClass **extends** Thread {

@Override

**public** **void** run() {

**for** (**int** i = 0; i < 10; i++) {

System.out.println(i);

**try** {

Thread.sleep(100);

} **catch** (InterruptedException ie) {

ie.printStackTrace();

}

}

}

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

UseThreadClass runner1=**new** UseThreadClass();

runner1.start();

}

}

Did you notice that Runnable Interface has only one method? And hence Runnable Interface passes the test of Functional Interface.

**public** **interface** Runnable {

**public** **abstract** **void** run();

}

So as per Java 8 we can use this method with our new Syntax. If I throw entire code here it won’t be easy to understand. Instead I will slowly develop the code line by line so it can be understood.

So let us begin.

Runnable runner = () -> {

System.***out***.println("Hello Lambda");

};

As we said Runnable is Functional Interface we can use Lambda Operator for it.

() determines that there are no parameters for the method in Runnable Interface.

-> tells what is to be done for the method.

{ }; is the scope for which the method will run.

Now let us try to use this syntax to print numbers from 1 to 10. We will use IntStream interface’s method range to create stream of 1 to 10 numbers both inclusive. Print the numbers with 100 milliseconds delay.

Runnable runner = () -> {

IntStream.*range*(1, 11).forEach(num -> {

**try** {

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

Thread.*sleep*(100);

} **catch** (InterruptedException ex) {

ex.printStackTrace();

}

});

};

**new** Thread(runner).start();

So this is how you can use Lambda Operator.

In next post we will see how to use Lambda Operator for Functional Interface without arguments.