Difference between Thread.start() and Thread.run() in Java

Difficulty Level : Basic Last Updated : 23 Jan, 2019

In Java's multi-threading concept, **start()** and **run()** are the two most important methods. Below are some of the differences between the Thread.start() and Thread.run() methods:

1. **New Thread creation:** When a program calls the *start()* method, a new thread is created and then the *run()* method is executed. But if we directly call the *run()* method then no new thread will be created and *run()* method will be executed as a normal method call on the current calling thread itself and no multi-threading will take place.
   Let us understand it with an example:

```
class MyThread extends Thread {
    public void run()
    {
        System.out.println("Current thread name: "
                           + Thread.currentThread().getName());
        System.out.println("run() method called");
    }
}

class GeeksforGeeks {
    public static void main(String[] args)
    {
        MyThread t = new MyThread();
        t.start();
    }
}
```

Output:

```
Current thread name: Thread-0
run() method called
```

As we can see in the above example, when we call the *start()* method of our thread class instance, a new thread is created with default name *Thread-0* and then *run()* method is called and everything inside it is executed on the newly created thread.
Now, let us try to call *run()* method directly instead of *start()* method:

```
class MyThread extends Thread {
    public void run()
    {
        System.out.println("Current thread name: "
                           + Thread.currentThread().getName());

        System.out.println("run() method called");
    }
}

class GeeksforGeeks {
    public static void main(String[] args)
    {
        MyThread t = new MyThread();
        t.run();
    }
}
```

Output:

```
Current thread name: main
run() method called
```

As we can see in the above example, when we called the *run()* method of our MyThread class, no new thread is created and the *run()* method is executed on the current thread i.e. *main* thread. Hence, no multi-threading took place. The *run()* method is called as a normal function call.

2. **Multiple invocation:** In Java's multi-threading concept, another most important difference between *start()* and *run()* method is that we can't call the *start()* method twice otherwise it will throw an *IllegalStateException* whereas *run()* method can be called multiple times as it is just a normal method calling.
   Let us understand it with an example:

```
class MyThread extends Thread {
    public void run()
    {
        System.out.println("Current thread name: "
                           + Thread.currentThread().getName());

        System.out.println("run() method called");
    }
}
```

```
class GeeksforGeeks {
    public static void main(String[] args)
    {
        MyThread t = new MyThread();
        t.start();
        t.start();
    }
}
```

Output:

```
Current thread name: Thread-0
run() method called
Exception in thread "main" java.lang.IllegalThreadStateException
    at java.lang.Thread.start(Thread.java:708)
    at GeeksforGeeks.main(File.java:11)
```

As we can see in the above example, calling *start()* method again raises *java.lang.IllegalThreadStateException*.
Now, let us try to call *run()* method twice:

```
class MyThread extends Thread {
    public void run()
    {
        System.out.println("Current thread name: "
                           + Thread.currentThread().getName());
        System.out.println("run() method called");
    }
}

class GeeksforGeeks {
    public static void main(String[] args)
    {
        MyThread t = new MyThread();
        t.run();
        t.run();
    }
}
```

Output:

```
Current thread name: main
run() method called
Current thread name: main
run() method called
```

As we can see in the above example, calling *run()* method twice doesn't raise any exception and it is executed twice as expected but on the *main* thread itself.

Summary

start()	run()
Creates a new thread and the run() method is executed on the newly created thread.	No new thread is created and the run() method is executed on the calling thread itself.
Can't be invoked more than one time otherwise throws *java.lang.IllegalStateException*	Multiple invocation is possible
Defined in *java.lang.Thread* class.	Defined in *java.lang.Runnable* interface and must be overriden in the implementing class.