

Stay safe, friends. Learn to code from home. Use our free 2,000 hour curriculum.

13 DECEMBER 2019 / [#JAVA](#)

Multithreading in Java: How to Get Started with Threads



Aditya Sridhar

I Love Technology and like to Explore new Technologies



What is a Thread?

A thread is a lightweight process. Any process can have multiple threads running in it.

For example in a web browser, we can have one thread which will load the user interface and another thread which will actually retrieve all the data that needs to be displayed in that interface.

What is MultiThreading?

Multithreading enables us to run multiple threads concurrently.

For example in a web browser, we can have one thread which handles the user interface, and in parallel we can have another thread which fetches the data to be displayed.

So multithreading improves the responsiveness of a system.

What is Concurrency?

Concurrency in the context of threads enables us to run multiple threads at the same time.

But do the threads really run at the same time?

Single Core Systems

The **Thread Scheduler** provided by the JVM decides which thread runs at any given time. The scheduler gives a small time slice to each thread.

So at any given time we have only one thread which is actually running in the processor. But because of the time slicing we get the feeling that multiple threads are running at the same time.

Multi Core Systems

Even in multiple core systems the thread scheduler is involved. But since we have multiple cores, we can actually have multiple threads running at the exact same time.

For example if we have a dual core system, then we can have 2 threads running at the exact same time. The first thread will run in the first core, and the second thread will run in the second core.

Why is Multithreading needed?

Multithreading enables us to improve the responsiveness of a system.

For example in a web browser, if everything ran in a single thread, then system would be completely unresponsive whenever data was being fetched to display. For example, if it takes 10 seconds to fetch the data, then in that 10 seconds we wont be able to do anything else in the web browser like opening new tabs, or even closing the web browser.

So running different parts of a program in different threads concurrently helps improve the responsiveness of a system.

How to write Multithreaded Programs



We can create threads in Java using the following

- Extending the thread class
- Implementing the runnable interface
- Implementing the callable interface
- By using the executor framework along with runnable and callable tasks

We will look at callables and the executor framework in a separate blog. In this article I will be mainly focussing on extending the thread class and implementing the runnable interface.

Extending the Thread Class

In order to create a piece of code which can be run in a thread, we create a class and then extend the **thread** class. The task being done by this piece of code needs to be put in the **run()** function.

In the below code you can see that **worker** is a class which extends the **thread** class, and the task of printing numbers 0 to 5 is being done inside the **run()** function.

```
class Worker extends Thread {  
  
    @Override  
    public void run() {  
        for (int i = 0; i <= 5; i++) {  
            System.out.println(Thread.currentThread().getName() + ": " +  
                                i);  
        }  
    }  
}
```

In the above code `Thread.currentThread().getName()` is used to get the name of the current thread which is running the code.

In order to create a **thread**, we just need to create an instance of the worker class. And then we can start the thread using the **start()** function.

```
public class ThreadClassDemo {  
    public static void main(String[] args) {  
        Thread t1 = new Worker();  
        Thread t2 = new Worker();  
        Thread t3 = new Worker();  
        t1.start();  
        t2.start();  
        t3.start();  
    }  
}
```

In the above code, we are creating 3 threads (t1,t2 and t3) from the worker class. Then we are starting the threads using the **start()** function.

Here is the final code for creating a thread by extending a thread class:

```
class Worker extends Thread {  
  
    @Override  
    public void run() {  
        for (int i = 0; i <= 5; i++) {
```

```
}  
  
}  
  
public class ThreadClassDemo {  
    public static void main(String[] args) {  
        Thread t1 = new Worker();  
        Thread t2 = new Worker();  
        Thread t3 = new Worker();  
        t1.start();  
        t2.start();  
        t3.start();  
    }  
}
```

Here is the output we get by running the above code:

```
<terminated> ThreadClassDemo [Java Application] C:\Program Files\Java\jdk1.8.0_11\bin\javaw.exe  
Thread-1: 0  
Thread-2: 0  
Thread-0: 0  
Thread-2: 1  
Thread-1: 1  
Thread-2: 2  
Thread-0: 1  
Thread-2: 3  
Thread-1: 2  
Thread-2: 4  
Thread-0: 2  
Thread-2: 5  
Thread-1: 3  
Thread-1: 4  
Thread-1: 5  
Thread-0: 3  
Thread-0: 4  
Thread-0: 5
```

You can see that all the 3 threads have printed the numbers from 0 to 5.

run in any particular sequence


Implementing the Runnable Interface

In order to create a piece of code which can be run in a thread, we create a class and then implement the **Runnable** interface. The task being done by this piece of code needs to be put in the **run()** function.

In the below code you can see that **RunnableWorker** is a class which implements **Runnable** interface, and the task of printing numbers 0 to 4 is being done inside the **run()** function.

```
class RunnableWorker implements Runnable{

    @Override
    public void run() {
        for (int i = 0; i <= 4; i++) {
            System.out.println(Thread.currentThread().getName() + ": " + i);
        }
    }
}
```



In order to create a thread, first we need to create an Instance of **RunnableWorker** which implements the **Runnable** interface.

Then we can create a new thread by creating an instance of the **Thread** class and passing the instance of **RunnableWorker** as the argument. This is shown in the code below:

```
public class RunnableInterfaceDemo {  
  
    public static void main(String[] args) {  
        Runnable r = new RunnableWorker();  
        Thread t1 = new Thread(r);  
        Thread t2 = new Thread(r);  
        Thread t3 = new Thread(r);  
  
        t1.start();  
        t2.start();  
        t3.start();  
    }  
}
```

The above code creates a runnable instance r. Then it create 3 threads (t1, t2 and t3) and passes r as the argument to the 3 threads. Then the **start()** function is used to start all 3 threads.

Here is the complete code for creating a thread by implementing the runnable interface:

```
class RunnableWorker implements Runnable{  
  
    @Override  
    public void run() {  
        for (int i = 0; i <= 4; i++) {  
            System.out.println(Thread.currentThread().getName() + ": "  
        }  
    }  
}  
  
public class RunnableInterfaceDemo {
```



```
Thread t1 = new Thread(r);
Thread t2 = new Thread(r);
Thread t3 = new Thread(r);

t1.start();
t2.start();
t3.start();

}

}
```

On running the above code, we will get the following output. The sequence of the output will change every time the code is run.

```
<terminated> RunnableFunctionalInterfaceDemo [Java Application] C:\Program F
Thread-2: 0
Thread-0: 0
Thread-1: 0
Thread-0: 1
Thread-2: 1
Thread-0: 2
Thread-1: 1
Thread-0: 3
Thread-2: 2
Thread-0: 4
Thread-1: 2
Thread-2: 3
Thread-1: 3
Thread-2: 4
Thread-1: 4
```

Implementing the runnable interface is a better option than extending the thread class since we can extend only one class, but we can implement multiple interfaces in Java.

Runnable Interface in Java 8

it has only one function, **run()**.

The below code shows how we can create a runnable instance in Java 8.

```
public class RunnableFunctionalInterfaceDemo {

    public static void main(String[] args) {

        Runnable r = () -> {
            for (int i = 0; i <= 4; i++) {
                System.out.println(Thread.currentThread().getName() + ":");
            }
        };

        Thread t1 = new Thread(r);
        Thread t2 = new Thread(r);
        Thread t3 = new Thread(r);

        t1.start();
        t2.start();
        t3.start();
    }
}
```

Here, instead of creating a class and then implementing the runnable interface, we can directly use a lambda expression to create a runnable instance as shown below:

```
Runnable r = () -> {
    for (int i = 0; i <= 4; i++) {
        System.out.println(Thread.currentThread().getName() + ": " + i);
    }
};
```

Code

The code in this article is available in the following GitHub repo:

<https://github.com/aditya-sridhar/basic-threads-demo>

Congrats 😊

You now know how to create threads by extending the thread class and by implementing the runnable interface.

I will discuss the thread life cycle and challenges while using threads in my next blog post.

My Website: <https://adityasridhar.com/>

Feel free to connect with me on LinkedIn or follow me on Twitter

If you read this far, tweet to the author to show them you care.

[Tweet a thanks](#)

Learn to code for free. freeCodeCamp's open source curriculum has helped more than 40,000 people get jobs as developers.

[Get started](#)

[Donate](#)

freeCodeCamp is a donor-supported tax-exempt 501(c)(3) nonprofit organization (United States Federal Tax Identification Number: 82-0779546)

Our mission: to help people learn to code for free. We accomplish this by creating thousands of videos, articles, and interactive coding lessons - all freely available to the public. We also have thousands of freeCodeCamp study groups around the world.

Donations to freeCodeCamp go toward our education initiatives, and help pay for servers, services, and staff.

You can [make a tax-deductible donation here](#).

Our Nonprofit

[About](#)[Alumni Network](#)[Open Source](#)[Shop](#)[Support](#)[Sponsors](#)[Academic Honesty](#)[Code of Conduct](#)[Privacy Policy](#)[Terms of Service](#)[Copyright Policy](#)

Trending Guides

[2019 Web Developer Roadmap](#)[Python Tutorial](#)[CSS Flexbox Guide](#)[JavaScript Tutorial](#)[Python Example](#)[HTML Tutorial](#)[Linux Command Line Guide](#)[JavaScript Example](#)[Git Tutorial](#)[React Tutorial](#)[Java Tutorial](#)[Linux Tutorial](#)[CSS Tutorial](#)[jQuery Example](#)[SQL Tutorial](#)[CSS Example](#)[React Example](#)

[Bootstrap Example](#)

[How to Set Up SSH Keys](#)

[WordPress Tutorial](#)

[PHP Example](#)