What is Thread and Runnable in Java? Example

What is Thread in Java?

Thread in Java is an independent path of execution that is used to run two tasks in parallel. When two threads run in parallel that is called multithreading in Java. Java is multithreaded from the start and has excellent support of Thread at language level e.g. `java.lang.Thread` class, synchronized keyword, volatile and final keyword make writing concurrent programs easier in Java than any other programming language like C++. Being multi-threaded is also a reason for Java's popularity and being the number one programming language.

On the other hand, if your program divides a task between two threads it also brings a lot of programming challenges and issues related to synchronization, deadlock, thread safety, and race conditions.

In short answer to the question of *what is Thread in Java* can be given like "Thread is a class in Java but also a way to execute something in parallel independently in Java".

Thread in Java requires a task that is executed by this thread independently and that task can be either Runnable or Callable which we will see in the next section along with an example of How to use multiple threads in Java. The difference between Thread and Runnable in Java is also a popular thread interview question in Java.

What is Runnable in Java?

Runnable represents a task in Java that is executed by Thread. `java.lang.Runnable` is an interface and defines only one method called `run()`. When a Thread is started in Java by using `Thread.start()` method it calls the `run()` method of Runnable task which was passed to Thread during creation.

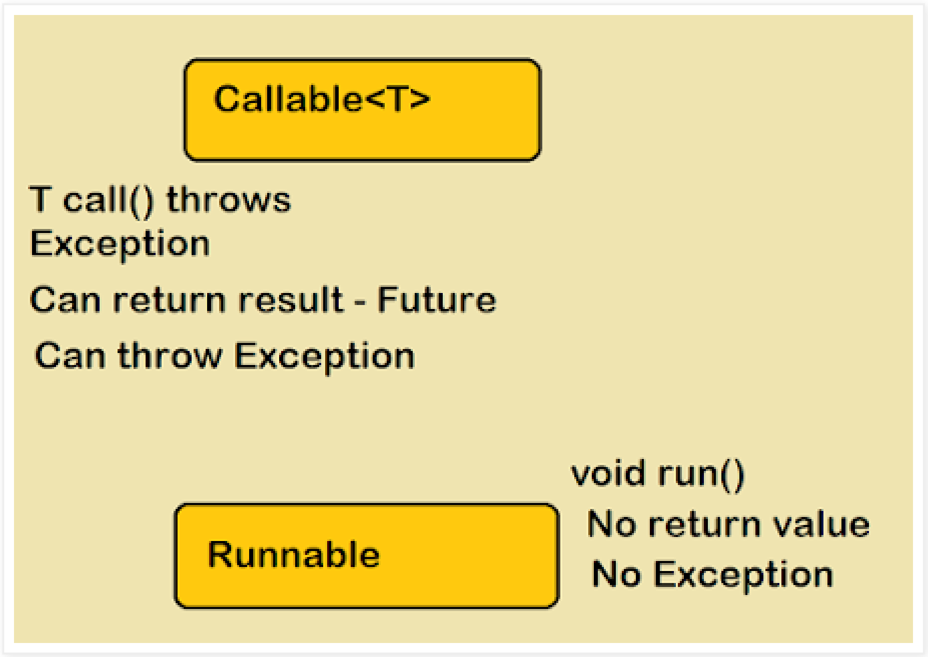
Code written inside the `run()` method is executed by this newly created thread. Since the `start()` method internally calls the `run()` method it has been a doubt among Java programmers that why not directly call the `run()` method.

This is also asked as what is the difference between `start()` and `run()` method in Java. Well, when you call the `Runnable` interface `run()` method directly, no new Thread will be created and the task defined inside the `run()` method is executed by calling thread.

There is another interface added in Java 1. 5 called Callable which can also be used in place of the `Runnable` interface in Java.

The `Callable` provides additional functionality over `Runnable` in terms of returning the result of the computation. Since the return type of run() method is void it can not return anything which is sometimes necessary.

On the other hand `Callable` interface defines the call() method which has a return type as `Future` which can be used to return the result of computation from Thread in Java. You can also see these Java Concurrency and Multithreading courses to learn more about Runnable, Callable, and other concurrency classes in Java.



Thread Example in Java.

Here is a simple example of Thread in Java. In this Java program, we create two Thread objects and pass them to two different `Runnable` instances which are implemented using the Anonymous class in Java.

We have also provided names to each thread as “Thread A” and “Thread B”, the name is optional and if you don’t give the name, Java will automatically provide a default name for your Thread like “Thread 0” and “Thread 1”.

When we start a thread using the `start()` method it calls the `run()` method which has code for printing the name of Thread two times for Thread A and three times for Thread B.

```
/**
 * Java Program to demonstrate how to use
Thread in Java with Example
 * Here two threads are provided Runnable
interface implementation using
 * anonymous class and when started they will
print Thread's name.
 * @author
 */
public class ThraedExample{

    public static void main(String args[]){

        //two threads in Java which runs in Parallel
        Thread threadA = new Thread(new Runnable(){
            public void run(){
                for(int i =0; i<2; i++){
                    System.out.println("This is thread :
" + Thread.currentThread().getName());
                }
            }
        }, "Thread A");

        //Runnable interface is implemented using Anonymous Class
        Thread threadB = new Thread(new Runnable(){
            public void run(){
                for(int i =0; i<3; i++){
                    System.out.println("This is thread :
" + Thread.currentThread().getName());
                }
            }
        }, "Thread B");
        //starting both Thread in Java
        threadA.start(); //start will call run method in new
thread
        threadB.start();


    }

}
```

Output

```
This is thread : Thread A
This is thread : Thread A
This is thread : Thread B
This is thread : Thread B
This is thread : Thread B
```

That’s all on What is Thread in Java, **What is Runnable in Java,** and How to use Thread in Java with Example