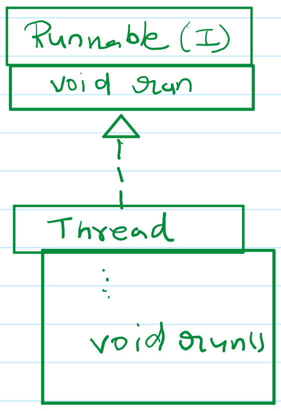
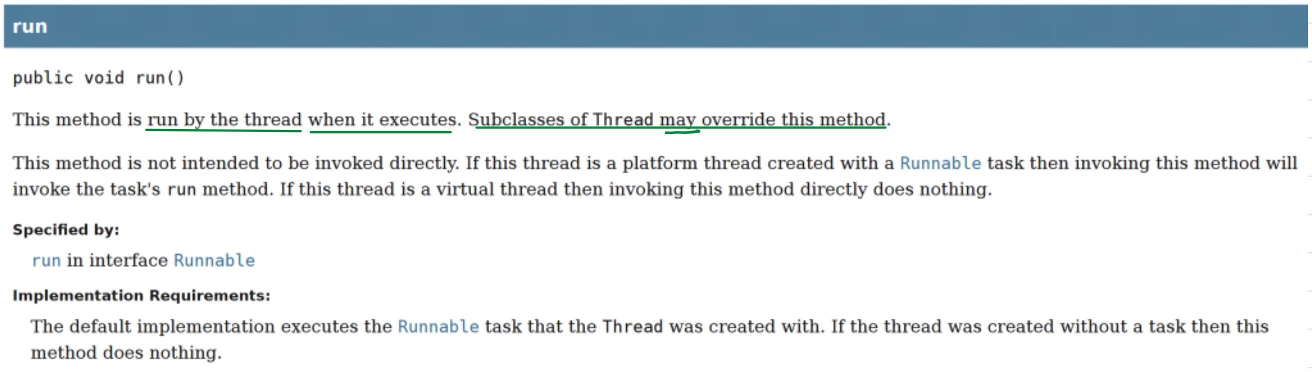
The Java Threading API

19 January 2026 18:31



neutral behavior
↓
public void run() { }



run

`public void run()`

This method is run by the thread when it executes. Subclasses of Thread may override this method.

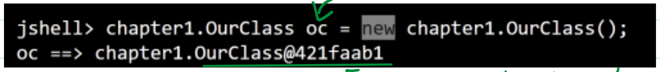
This method is not intended to be invoked directly. If this thread is a platform thread created with a `Runnable` task then invoking this method will invoke the task's `run` method. If this thread is a virtual thread then invoking this method directly does nothing.

Specified by:

`run` in interface `Runnable`

Implementation Requirements:

The default implementation executes the `Runnable` task that the `Thread` was created with. If the thread was created without a task then this method does nothing.



```
jshell> chapter1.OurClass oc = new chapter1.OurClass();
oc ==> chapter1.OurClass@421faab1
```

Object created, when Thread class is not extended.

Normal object

```
package chapter1;
public class OurClass{
    @Override
    public void run(){
        int i;
        for(i = 0; i < 100; i++){
            System.out.println("Hello");
        }
    }
}
```

java.lang.Thread.

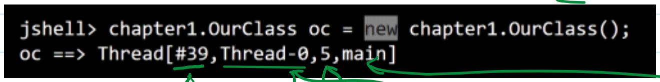
To start the thread we need to call start() method.

This method will be executed by newly created thread.
Developers should override this method with the code they want new thread

```
package chapter1;
public class OurClass extends Thread{
    @Override
    public void run(){
        int i;
        for(i = 0; i < 100; i++){
            System.out.println("Hello");
        }
```

```
            for(i = 0; i < 100; i++){
                System.out.println("Hello");
            }
        }
    }
```

Developers should override this method with the code they want new thread to run.



```
jshell> chapter1.OurClass oc = new chapter1.OurClass();
oc ==> Thread[#39,Thread-0,5,main]
```

Thread ID

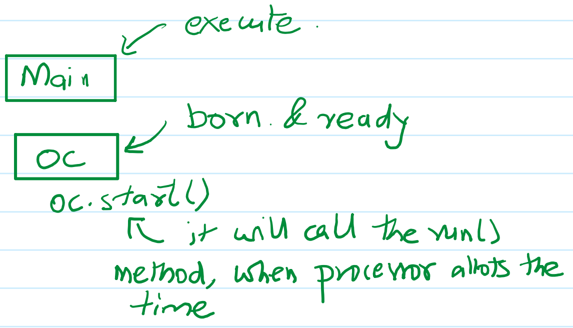
Thread name

Priority

Which thread main.

Threading using the Thread class
We consider threads as separate task that execute in parallel. These tasks are simply code executed by the thread, and this code is actually part of our program. The code may download an image from the server or may play audio file on the speakers or any other task. Because it is a code. It can be executed by our original thread. To introduce the parallelism we desire, we must create a new thread and arrange for the new thread to. execute the appropriate code.

```
package chapter1;
public class TestFirstThread {
    public static void main(String[] args) {
        OurClass oc = new OurClass();
        oc.start();
        for (int i = 0; i < 10; i++) {
            System.out.println("Main");
        }
    }
}
```


execute
Main
born & ready
oc
oc.start()
it will call the run() method, when processor allots the time

calls the run() method defined in this thread class.

run() vs main()
In essence, the run() method may be thought of as main() method of the newly formed thread. A new thread begins execution with a run() method in the same way a program begins execution with the main() method.
While the main() method receive its argument from argv parameter (which is typically set from the command line) The newly created thread must receive its arguments programmatically from the originating threads. Hence, parameters can be passed in via the constructor, the static instance variable, or any other technique designed by the developer.