



Setting-up Threads

This lesson discusses how threads can be created in Java.

We'll cover the following



- Creating Threads
- Runnable Interface
- Subclassing Thread class

Creating Threads#

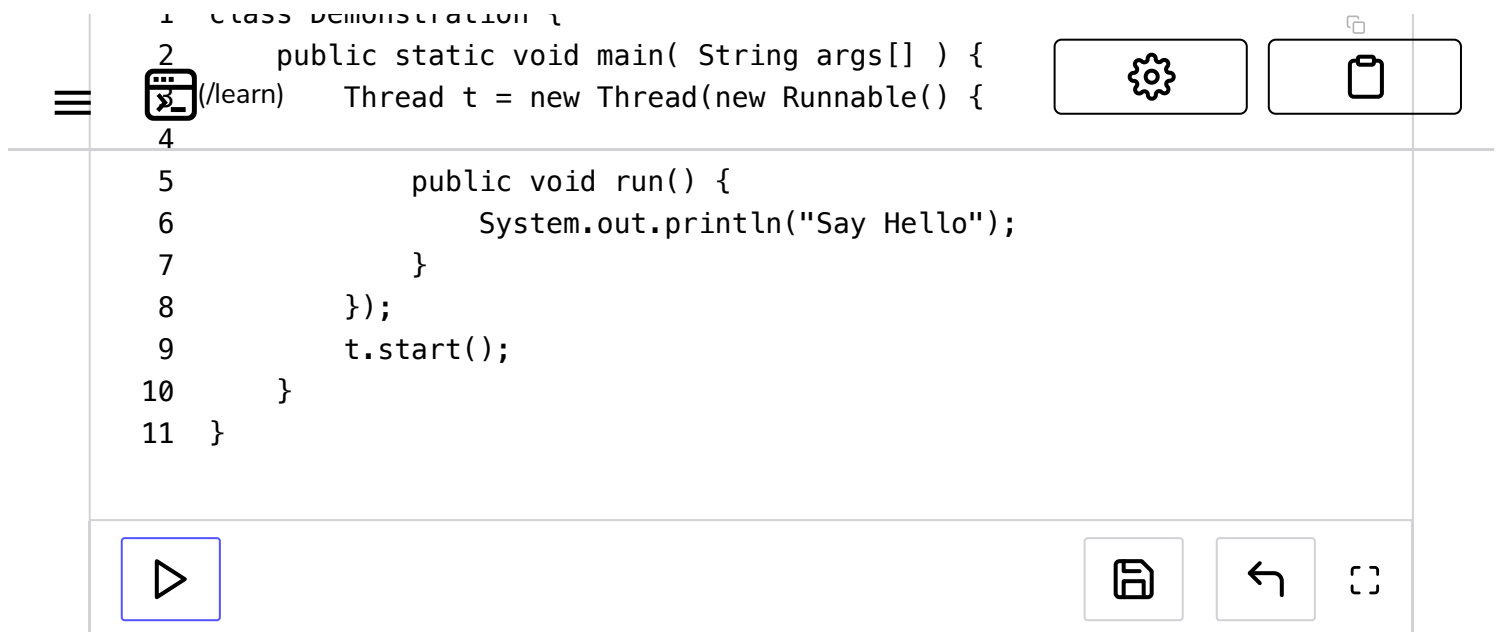
To use threads, we need to first create them. In the Java language framework, there are multiple ways of setting up threads.

Runnable Interface#

When we create a thread, we need to provide the created thread code to execute or in other words we need to tell the thread what *task* to execute. The code can be provided as an object of a class that implements the Runnable

(<https://docs.oracle.com/javase/8/docs/api/java/lang/Runnable.html>) interface. As the name implies, the interface forces the implementing class to provide a `run` method which in turn is invoked by the thread when it starts.

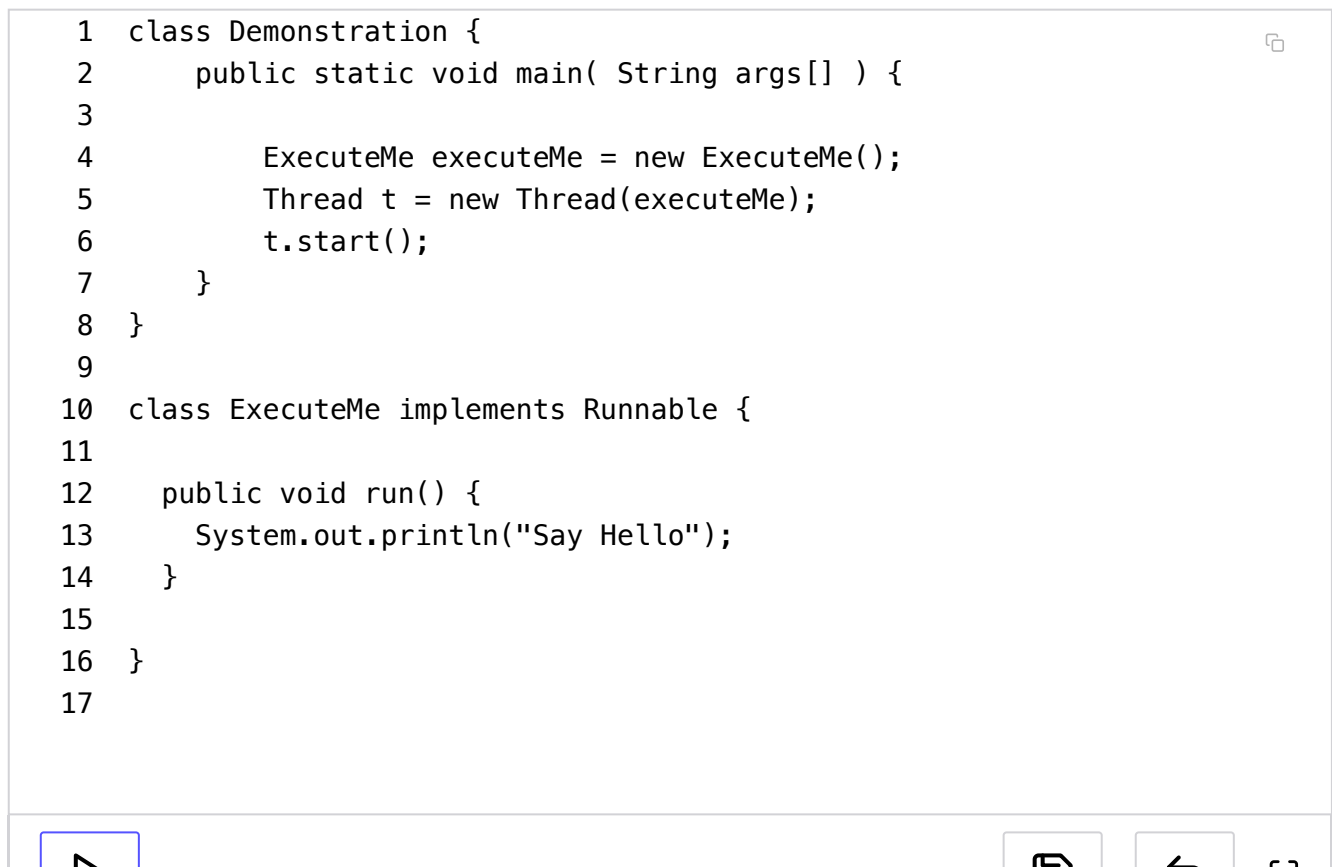
The runnable interface is the basic abstraction to represent a logical task in Java.



The image shows a code editor window with a Java code snippet. The code defines a class named `Demonstration` with a `main` method. Inside the `main` method, a `Thread` object `t` is created, passing an anonymous `Runnable` implementation to its constructor. The `run` method of the anonymous class prints "Say Hello". The IDE interface includes a menu icon on the left, a toolbar with a settings gear and a clipboard icon on the right, and a bottom toolbar with a play button, a save icon, a back arrow, and a full-screen icon.

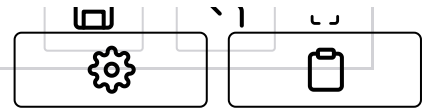
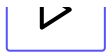
```
1 class Demonstration {
2     public static void main( String args[] ) {
3         Thread t = new Thread(new Runnable() {
4
5             public void run() {
6                 System.out.println("Say Hello");
7             }
8         });
9         t.start();
10    }
11 }
```

We defined an anonymous class inside the `Thread` class's constructor and an instance of it is instantiated and passed into the `Thread` object. Personally, I feel anonymous classes decrease readability and would prefer to create a separate class implementing the `Runnable` interface. An instance of the implementing class can then be passed into the `Thread` object's constructor. Let's see how that could have been done.



The image shows a code editor window with a Java code snippet. The code defines a class named `Demonstration` with a `main` method. Inside the `main` method, an instance of a class named `ExecuteMe` is created and passed to the `Thread` constructor. The `ExecuteMe` class implements the `Runnable` interface and has a `run` method that prints "Say Hello". The IDE interface includes a menu icon on the left, a toolbar with a settings gear and a clipboard icon on the right, and a bottom toolbar with a play button, a save icon, a back arrow, and a full-screen icon.

```
1 class Demonstration {
2     public static void main( String args[] ) {
3
4         ExecuteMe executeMe = new ExecuteMe();
5         Thread t = new Thread(executeMe);
6         t.start();
7     }
8 }
9
10 class ExecuteMe implements Runnable {
11
12     public void run() {
13         System.out.println("Say Hello");
14     }
15
16 }
17
```



Subclassing Thread class#

The second way to set-up threads is to subclass the Thread (<https://docs.oracle.com/javase/7/docs/api/java/lang/Thread.html>) class itself as shown below.

```
1 class Demonstration {
2     public static void main( String args[] ) throws Exception {
3         ExecuteMe executeMe = new ExecuteMe();
4         executeMe.start();
5         executeMe.join();
6
7     }
8 }
9
10 class ExecuteMe extends Thread {
11
12     @Override
13     public void run() {
14         System.out.println("I ran after extending Thread class");
15     }
16
17 }
18
```



The con of the second approach is that one is forced to extend the Thread (<https://docs.oracle.com/javase/7/docs/api/java/lang/Thread.html>) class which limits code's flexibility. Passing in an object of a class implementing the Runnable (<https://docs.oracle.com/javase/8/docs/api/java/lang/Runnable.html>) interface may be a better choice in most cases.

In next lesson, we'll study ways of manipulating threads



Interviewing soon? We've partnered with Hired so that companies apply to you
[utm_source=educative&utm_medium=lesson&utm_location=CA&utm_campaign=](#)



← Back

Next →

Next Steps

Basic Thread Handling



Completed



Report
an Issue



Ask a Question

(https://discuss.educative.io/tag/setting-up-threads__java-concurrency-reference__java-multithreading-for-senior-engineering-interviews)