



Setting-up Threads

This lesson discusses how threads can be created in Java.

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.

```
1 class Demons
2     public s
```



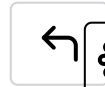
```
3      Thre
4
5
6
7
8      });
9      t.st
10     }
11 }
```



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.

```
1 class Demons
2     public s
3
4     Exec
5     Thre
6     t.st
7 }
8 }
9
10 class Execut
11
12     public voi
13     System.o
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 Demons
2     public s
3         Exec
4         exec
5         exec
6
7     }
8 }
9
10 class Execut
11
12     @Override
13     public voi
14         System.o
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

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=educative](https://hired.com/?utm_source=educative&utm_medium=lesson&utm_location=CA&utm_campaign=educative)



← Back

Next →

Next Steps

Basic Thread Handling



Mark as Completed



Report an
Issue



Ask a Question

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