



## Quiz 3

Questions on how threads can be created

### Question # 1

*Give an example of creating a thread using the **Runnable** interface?*

The below snippet creates an instance of the **Thread** class by passing in a lambda expression to create an anonymous class implementing the **Runnable** interface.

```
Thread t = new Thread(() -> {
    System.out.println(this.getClass().getSimpleName
()));
});

t.start();
t.join();
```

```
1 class Demonstration {
2     public static void main( String args[]) throws Exception {
3
4         Thread t = new Thread(() -> {
5             System.out.println("Hello from thread !");
6         });
7
8         t.start();
9         t.join();
10
```



```
10  
11     }  
12 }
```



## Question # 2

*Give an example of a thread running a task represented by the `Callable<V>` interface?*

There's no constructor in the **Thread** class that takes in a type of **Callable**. However, there is one that takes in a type of **Runnable**. We can't directly execute a callable task using an instance of the **Thread** class. However we can submit the callable task to an executor service. Both approaches are shown below:

### Callable with Thread Class



```
// Anonymous class
Callable<Void> task = new Callable<Void>() {

    @Override
    public Void call() throws Exception {
        System.out.println("Using callable indirectly with instance of thread class");
        return null;
    }
};

// creating future task
FutureTask<Void> ft = new FutureTask<>(task);
Thread t = new Thread(ft);
t.start();
t.join();
```

## Callable with Executor Service

```
// Anonymous class
Callable<Void> task = new Callable<Void>() {

    @Override
    public Void call() throws Exception {
        System.out.println("Using callable indirectly with instance of thread class");
        return null;
    }
};

ExecutorService executorService = Executors.newFixedThreadPool(5);
executorService.submit(task);
executorService.shutdown();
```



```
1 import java.util.concurrent.Callable;
2 import java.util.concurrent.FutureTask;
3 import java.util.concurrent.ExecutorService;
4 import java.util.concurrent.Executors;
5
6 class Demonstration {
7     public static void main( String args[] ) throws Exception {
8         usingExecutorService();
9         usingThread();
10
11     }
12
13     static void usingExecutorService() {
14         // Anonymouse class
15         Callable<Void> task = new Callable<Void>() {
16
17             @Override
18             public Void call() throws Exception {
19                 System.out.println("Using callable with executor service.");
20                 return null;
21             }
22         };
23
24         ExecutorService executorService = Executors.newFixedThreadPool(5);
25         executorService.submit(task);
26         executorService.shutdown();
27     }
28
```



## Question # 3

*Give an example of representing a class using the Thread class.*

We can extend from the **Thread** class to represent our task. Below is an example of a class that computes the square roots of given numbers. The

**Task** class encapsulates the logic for the task being performed.



```
class Task<T extends Number> extends Thread {

    T item;

    public Task(T item) {
        this.item = item;
    }

    public void run() {
        System.out.println("square root is: " + Math.sqrt(item
.doubleValue()));
    }
}
```

```
1 class Demonstration {
2     public static void main( String args[] ) throws Exception{
3
4         Thread[] tasks = new Thread[10];
5         for(int i = 0;i<10;i++) {
6             tasks[i] = new Task(i);
7             tasks[i].start();
8         }
9
10        for(int i = 0;i<10;i++) {
11            tasks[i].join();
12        }
13    }
14 }
15
16 class Task<T extends Number> extends Thread {
17
18     T item;
19
20     public Task(T item) {
21         this.item = item;
22     }
23
24     public void run() {
```

```
25         System.out.println("square root is: " + Math.sqrt(item.doubleValue()))
26     }
27 }
```



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



← Back

Next →

Quiz 2

Quiz 4



Mark as Completed



Report an  
Issue



Ask a Question

([https://discuss.educative.io/tag/quiz-3\\_\\_revision-quizzes\\_\\_java-multithreading-for-senior-engineering-interviews](https://discuss.educative.io/tag/quiz-3__revision-quizzes__java-multithreading-for-senior-engineering-interviews))