



Quiz 3

Questions on how threads can be created

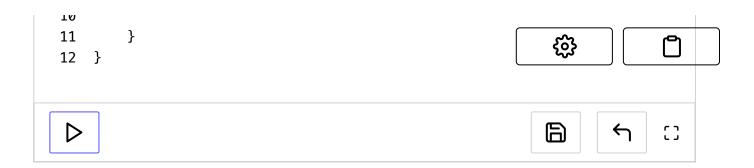
Question # 1

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();
```



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





```
// Anoymous class
Callable<Void> task = new Callable<Void>() {
     @Override
     public Void call() throws Exception {
          System.out.println("Using callable indirectl
y 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

```
// Anoymous class
Callable<Void> task = new Callable<Void>() {
    @Override
    public Void call() throws Exception {
        System.out.println("Using callable indirectl)
y with instance of thread class");
        return null;
    }
};

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



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

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

25 26 } 27 }	System.out.println("square root is: " + Ma	th.sqrt(i	tem.do	oub LeVa	Lue()
\triangleright			\leftarrow	[]	

Interviewing soon? We've partnered with Hired so that companies apply to your utm_source=educative&utm_medium=lesson&utm_location=CA&utm_campage.

