

```

1 -----TaskExecutorUsage.java-----
2 package com.ameya.test;
3
4 import java.util.concurrent.ExecutorService;
5 import java.util.concurrent.Executors;
6 import java.util.concurrent.Future;
7
8 import com.ameya.tasks.TaskOne;
9 import com.ameya.tasks.TaskTwo;
10
11 public class TestExecutorUsage {
12     private static ExecutorService executor = null;
13     private static volatile Future taskOneResults = null;
14     private static volatile Future taskTwoResults = null;
15
16     private static void checkTasks()throws Exception{
17         if(taskOneResults ==null || taskOneResults.isDone()||
18            taskOneResults.isCancelled()) {
19             taskOneResults=executor.submit(new TaskOne());
20         }
21         if(taskTwoResults ==null || taskTwoResults.isDone()||
22            taskTwoResults.isCancelled()) {
23             taskTwoResults=executor.submit(new TaskTwo());
24         }
25     }
26
27     public static void main(String[] args) {
28         executor=Executors.newFixedThreadPool(2);
29         while(true) {
30             try {
31                 checkTasks();
32                 Thread.sleep(1000);
33             }catch(Exception e) {
34                 System.out.println(e.getMessage());
35             }
36         }
37     }
38 }
39 -----TaskOne.java-----

```

```
40 package com.ameya.tasks;
41
42 public class TaskOne implements Runnable {
43
44     @Override
45     public void run() {
46         System.out.println("Executing Task One");
47         try {
48             Thread.sleep(2000);
49         } catch (InterruptedException e) {
50             e.printStackTrace();
51         }
52         System.out.println("TaskOne Terminates....");
53     }
54
55 }
56 -----TaskTwo.java-----
57 package com.ameya.tasks;
58
59 public class TaskTwo implements Runnable {
60
61     @Override
62     public void run() {
63         System.out.println("Executing Task Two");
64         try {
65             Thread.sleep(2000);
66         } catch (InterruptedException e) {
67             e.printStackTrace();
68         }
69         System.out.println("Task Two Terminates....");
70     }
71 }
72 -----TaskThree.java-----
73 package com.ameya.tasks;
74
75 public class TaskThree implements Runnable {
76
77     @Override
78     public void run() {
79         System.out.println("Executing Task Three");
80         try {
```

```
81         Thread.sleep(2000);
82     } catch (InterruptedException e) {
83         e.printStackTrace();
84     }
85     System.out.println("TaskThree Terminates....");
86 }
```

```
87
88 }
89 -----MultiRunnable.java-----
```

```
90 package com.ameya.tasks;
91
92 import java.util.List;
93
94 public class MultiRunnable implements Runnable {
95     private List<Runnable> runnables;
96     public MultiRunnable(List<Runnable> runnables) {
97         this.runnables=runnables;
98     }
99     @Override
100     public void run() {
101         for(Runnable runnable : runnables) {
102             new Thread(runnable).start();
103         }
104
105     }
106
107 }
```

```
108 -----MultiRunnableTaskExecutor.java-----
```

```
109 package com.ameya.test;
110
111 import java.util.ArrayList;
112 import java.util.List;
113 import java.util.concurrent.ArrayBlockingQueue;
114 import java.util.concurrent.BlockingQueue;
115 import java.util.concurrent.RejectedExecutionHandler;
116 import java.util.concurrent.ThreadPoolExecutor;
117 import java.util.concurrent.TimeUnit;
118
119 import com.ameya.handlers.RejectedExecutionHandlerImpl;
120 import com.ameya.tasks.MultiRunnable;
121 import com.ameya.tasks.TaskOne;
```

```

122 import com.ameya.tasks.TaskThree;
123 import com.ameya.tasks.TaskTwo;
124
125 public class MultiRunnableTaskExecutor {
126
127     public static void main(String[] args) {
128         BlockingQueue<Runnable> worksQueue=new
129             ArrayBlockingQueue<Runnable>(10);
130         RejectedExecutionHandler rejectionHandler=new
131             RejectedExecutionHandlerImpl();
132         ThreadPoolExecutor executor=new
133             ThreadPoolExecutor(3,3,10,TimeUnit.SECONDS,
134                 worksQueue,rejectionHandler);
135         executor.prestartAllCoreThreads();
136         List<Runnable> taskGroup=new ArrayList<Runnable>();
137         taskGroup.add(new TaskOne());
138         taskGroup.add(new TaskTwo());
139         taskGroup.add(new TaskThree());
140         taskGroup.add(new TaskTwo());
141         taskGroup.add(new TaskThree());
142         taskGroup.add(new TaskTwo());
143         taskGroup.add(new TaskThree());
144         worksQueue.add(new MultiRunnable(taskGroup));
145     }
146 }
147
148 -----RejectedExecutionHandlerImpl.java-----
149 package com.ameya.handlers;
150
151 import java.util.concurrent.RejectedExecutionHandler;
152 import java.util.concurrent.ThreadPoolExecutor;
153
154 public class RejectedExecutionHandlerImpl implements
155     RejectedExecutionHandler {
156
157     @Override
158     public void rejectedExecution(Runnable r, ThreadPoolExecutor
159         executor) {
160         System.out.println(r.toString()+" Has Been Rejected ! ");
161     }
162 }

```

158

159 }

160 -----