Semaphore in Java

Semaphore

Java's semaphore can be **releas()-ed** or **acquire()-d** for signalling amongst threads. However the important call out when using semaphores is to make sure that the permits acquired should equal permits returned. Take a look at the following example, where a runtime exception causes a deadlock.

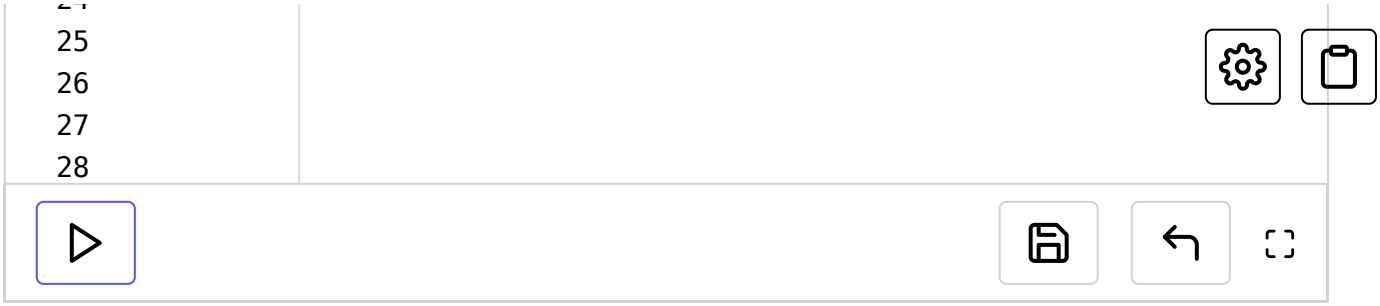
```
1  import java.
2
3  class Demons
4
5      public s
6          Inco
7      }
8  }
9
10 class Incorr
11
12     public s
13
14         fina
15
16         Thre
17
18
19
20
21
22
```

```
23
24
25
26
27
28
```

Incorrect Use of Semaphore

The above code when run would time out and show that one of the threads threw an exception. The code is never able to release the semaphore causing the other thread to block forever. Whenever using locks or semaphores, remember to unlock or release the semaphore in a **finally** block. The corrected version appears below.

```
1  import java.
2
3  class Demons
4
5      public s
6          Corr
7      }
8  }
9
10 class Correc
11
12     public s
13
14         fina
15
16         Thre
17
18
19
20
21
22
23
24
```



Running the above code will print the **`Exiting Program`** statement.

Back
Missed Signals

Next
Spurious Wakeups

Mark as Completed

Report an Issue

Ask a Question (https://discuss.educative.io/tag/semaphore-in-java__multithreading-in-java__java-multithreading-for-senior-engineering-interviews)