Concurrent Programming in Java

February 14, 2011

Printer

Question

Develop a Java class called `Printer` that implements `Runnable` and prints the thread's name 1000 times.

Answer

```
public class Printer implements Runnable
{
   public void run()
   {
      final int NUMBER = 1000;
      for (int i = 0; i < NUMBER; i++)
      {
         System.out.println(...);
      }
   }
}
```

Name of Thread

Question

The `Thread` has a name, but in this case `Printer` is not a `Thread`. How do we get the name of the `Thread`?

Name of Thread

Question

The `Thread` has a name, but in this case `Printer` is not a `Thread`. How do we get the name of the `Thread`?

Answer

```
Thread.currentThread().getName()
```

Two Concurrent Printers

Question

Develop an app that creates two `Printers` with names 1 and 2 and run them concurrently.

Two Concurrent Printers

```
public class PrinterTest
{
   public static void main(String[] args)
   {
      new Thread(new Printer(), "1").start();
      new Thread(new Printer(), "2").start();
   }
}
```

Semaphores

The Java class library contains the class `java.util.concurrent.Semaphore`.

The method `acquire` represents the P-operation and the method `release` represents the V-operation.

The Readers-Writers Problem

The readers and writers problem, due to Courtois, Heymans and Parnas, is another classical concurrency problem. It models access to a database. There are many competing threads wishing to read from and write to the database. It is acceptable to have multiple threads reading at the same time, but if one thread is writing then no other thread may either read or write. The problem is how do you program the reader and writer threads?

Shared Variables

```
int readers = 0;
semaphore mutex = 1;
semaphore token = 1;
```

Reader

```
P(mutex);
readers++;
if (readers == 1)
  P(token);
V(mutex);
read
P(mutex);
readers--;
if (readers == 0)
  V(token);
V(mutex);
```

Writer

```
P(token);
 write
V(token);
```