Concurrency

EECS 4315

www.eecs.yorku.ca/course/4315/

How many different executions?

Question

One thread prints 1 one. Another thread prints 1 two. How many different executions are there?

How many different executions?

Question

One thread prints 1 one. Another thread prints 1 two. How many different executions are there?

Answer

2.

How many different executions?

Question

One thread prints 2 ones. Another thread prints 2 twos. How many different executions are there?

How many different executions?

Question

One thread prints 2 ones. Another thread prints 2 twos. How many different executions are there?

Answer

6.

How many different executions?

Question

One thread prints 3 ones. Another thread prints 3 twos. How many different executions are there?

How many different executions?

Question

One thread prints 3 ones. Another thread prints 3 twos. How many different executions are there?

Answer

20.

How many different executions?

Question

One thread prints 1000 ones. Another thread prints 1000 twos. How many different executions are there?

How many different executions?

Question

One thread prints 1000 ones. Another thread prints 1000 twos. How many different executions are there?

Answer

20481516269894897143351625029808250443964248879813970338203826376717481862020837558289329941826102062014647663199980236924154817980045247920180475497692615785630128966343206471485115239525165122776858861153954625614790737866846415444453361761377007385567381458963007130651045595951447988874620636871851455182855117316627625366377308468293225538904974385948143175503078379644437081008516372482746279141701661988376484084354143081778594703774656518847551468074969467492380303310181872329800966856745856025254991011811352535346588879419666536749045113061100963119062703425022931559111089767339639911491 20.

How many executions?

Question

One thread prints 1000 ones. Another thread prints 1000 twos. How many different executions are there?

How many executions?

Question

One thread prints 1000 ones. Another thread prints 1000 twos. How many different executions are there?

Answer

$\binom{2000}{1000} = \frac{2000!}{1000!1000!}$.

How many executions?

Question

One thread executes n instructions. Another thread executes n instructions. How many different executions are there?

How many executions?

Question

One thread executes n instructions. Another thread executes n instructions. How many different executions are there?

Answer

At most $\binom{2n}{n}$.

How many executions?

Question

One thread executes n instructions. Another thread executes n instructions. How many different executions are there?

Answer

At most $\binom{2n}{n}$.

Question

Can there be fewer?

How many executions?

Question

One thread executes n instructions. Another thread executes n instructions. How many different executions are there?

Answer

At most $\binom{2n}{n}$.

Question

Can there be fewer?

Answer

Yes. For example, if each instruction is `x = 1` then there is only one execution.

How many executions?

Question

There are k threads. Each thread executes n instructions. How many different executions are there?

How many executions?

Answer

$$\binom{kn}{n}\binom{(k-1)n}{n}\cdots\binom{2n}{n}$$

How many executions?

Answer

$$\binom{kn}{n}\binom{(k-1)n}{n}\cdots\binom{2n}{n}$$
$$= \frac{(kn)!}{n!((k-1)n)!}\frac{((k-1)n)!}{n!((k-2)n)!}\cdots\frac{(2n)!}{n!n!}$$

How many executions?

Answer

$$\begin{aligned}
&\binom{kn}{n}\binom{(k-1)n}{n}\cdots\binom{2n}{n} \\
&= \frac{(kn)!}{n!((k-1)n)!}\frac{((k-1)n)!}{n!((k-2)n)!}\cdots\frac{(2n)!}{n!n!} \\
&= \frac{(kn)!}{(n!)^k}
\end{aligned}$$

How many executions?

Answer

$$
\begin{aligned}
&\binom{kn}{n}\binom{(k-1)n}{n}\cdots\binom{2n}{n} \\
&= \frac{(kn)!}{n!((k-1)n)!}\frac{((k-1)n)!}{n!((k-2)n)!}\cdots\frac{(2n)!}{n!n!} \\
&= \frac{(kn)!}{(n!)^k} \\
&= \frac{(kn)(kn-1)\cdots(kn-n+1)}{n!}\cdots\frac{2n(2n-1)\cdot(n+1)}{n!}\frac{n!}{n!}
\end{aligned}
$$

How many executions?

Answer

$$
\begin{aligned}
&\binom{kn}{n}\binom{(k-1)n}{n}\cdots\binom{2n}{n} \\
&= \frac{(kn)!}{n!((k-1)n)!}\frac{((k-1)n)!}{n!((k-2)n)!}\cdots\frac{(2n)!}{n!n!} \\
&= \frac{(kn)!}{(n!)^k} \\
&= \frac{(kn)(kn-1)\cdots(kn-n+1)}{n!}\cdots\frac{2n(2n-1)\cdot(n+1)}{n!}\frac{n!}{n!} \\
&\geq \left(\frac{2n(2n-1)\cdot(n+1)}{n!}\right)^{k-1}
\end{aligned}
$$

How many executions?

Answer

$$
\begin{aligned}
&\binom{kn}{n}\binom{(k-1)n}{n}\cdots\binom{2n}{n} \\
&= \frac{(kn)!}{n!((k-1)n)!}\frac{((k-1)n)!}{n!((k-2)n)!}\cdots\frac{(2n)!}{n!n!} \\
&= \frac{(kn)!}{(n!)^k} \\
&= \frac{(kn)(kn-1)\cdots(kn-n+1)}{n!}\cdots\frac{2n(2n-1)\cdot(n+1)}{n!}\frac{n!}{n!} \\
&\geq \left(\frac{2n(2n-1)\cdot(n+1)}{n!}\right)^{k-1} \\
&= \left(\frac{2n(2n-1)\cdot(n+1)}{n(n-1)\cdots 2}\right)^{k-1}
\end{aligned}
$$

How many executions?

Answer

$$
\begin{aligned}
&\binom{kn}{n}\binom{(k-1)n}{n}\cdots\binom{2n}{n} \\
&= \frac{(kn)!}{n!((k-1)n)!}\frac{((k-1)n)!}{n!((k-2)n)!}\cdots\frac{(2n)!}{n!n!} \\
&= \frac{(kn)!}{(n!)^k} \\
&= \frac{(kn)(kn-1)\cdots(kn-n+1)}{n!}\cdots\frac{2n(2n-1)\cdot(n+1)}{n!}\frac{n!}{n!} \\
&\geq \left(\frac{2n(2n-1)\cdot(n+1)}{n!}\right)^{k-1} \\
&= \left(\frac{2n(2n-1)\cdot(n+1)}{n(n-1)\cdots 2}\right)^{k-1} \\
&\geq n^{k-1}
\end{aligned}
$$

How many executions?

Question

There are k threads. Each thread executes n instructions. How many different executions are there?

Answer

In the worst case, more than n^{k-1}.

Conclusion

The number of different executions may grow exponential in the number of threads.

Java code

Assume that a `Printer` prints its name once.

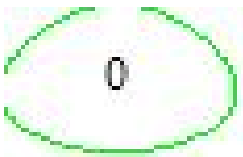
```
public static void main(String[] args) {
  Printer one = new Printer("1");
  one.run();
}
```

Executions

Question

Draw the state-transition diagram.

Executions



Java code

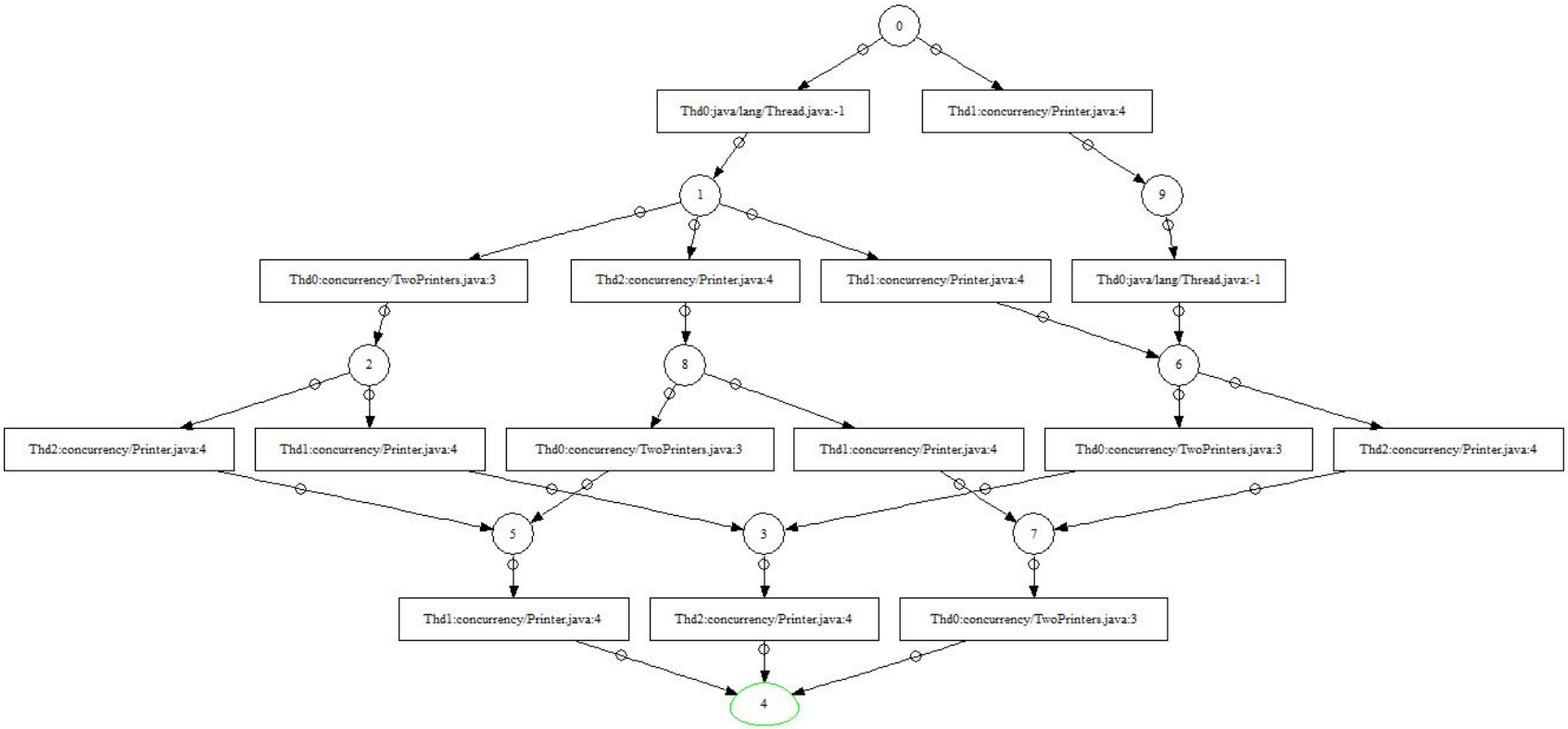
```
public static void main(String[] args) {
  Printer one = new Printer("1");
  Printer two = new Printer("2");
  one.start();
  two.start();
}
```

Executions

Question

Draw the state-transition diagram.

Executions


0
Thd0:java/lang/Thread.java:-1
Thd1:concurrency/Printer.java:4
1
9
Thd0:concurrency/TwoPrinters.java:3
Thd2:concurrency/Printer.java:4
Thd1:concurrency/Printer.java:4
Thd0:java/lang/Thread.java:-1
2
8
6
Thd2:concurrency/Printer.java:4
Thd1:concurrency/Printer.java:4
Thd0:concurrency/TwoPrinters.java:3
Thd1:concurrency/Printer.java:4
Thd0:concurrency/TwoPrinters.java:3
Thd2:concurrency/Printer.java:4
5
3
7
Thd1:concurrency/Printer.java:4
Thd2:concurrency/Printer.java:4
Thd0:concurrency/TwoPrinters.java:3
4

Counter class

Problem

Implement the class `Counter` with attribute `value`, initialized to zero, and the methods `increment` and `decrement`.

Counter class

Problem

Implement the class `Counter` with attribute `value`, initialized to zero, and the methods `increment` and `decrement`.

Question

Can multiple threads share a `Counter` object and use methods such as `increment` and `decrement` concurrently?

Counter class

Problem

Implement the class `Counter` with attribute `value`, initialized to zero, and the methods `increment` and `decrement`.

Question

Can multiple threads share a `Counter` object and use methods such as `increment` and `decrement` concurrently?

Answer

Yes, but, as before, if two threads invoke `increment` concurrently, the counter may only be incremented by one (rather than two).

Synchronized methods

Methods such as `increment` should be executed atomically. This can be accomplished by declaring the method to be `synchronized`.

A lock is associated with every object. For threads to execute a `synchronized` method on such the object, first its lock needs to be acquired.

Synchronized methods

Methods such as `increment` should be executed atomically. This can be accomplished by declaring the method to be `synchronized`.

A lock is associated with every object. For threads to execute a `synchronized` method on such the object, first its lock needs to be acquired.

```
public synchronized void increment() {
  this.value++;
}
```

Resource class

Problem

Implement the class `Resource` with attribute `available`, initialized to true, and the methods `acquire` and `release`.

Wait and notify

The `Object` class contains the following three methods:

- `wait`: causes the current thread to wait for this object's lock until another thread wakes it up.
- `notify`: wakes up a single thread waiting on this object's lock; if there is more than one waiting, an arbitrary one is chosen; if there are none, nothing is done.
- `notifyAll`: wakes up all threads waiting on this objects lock.

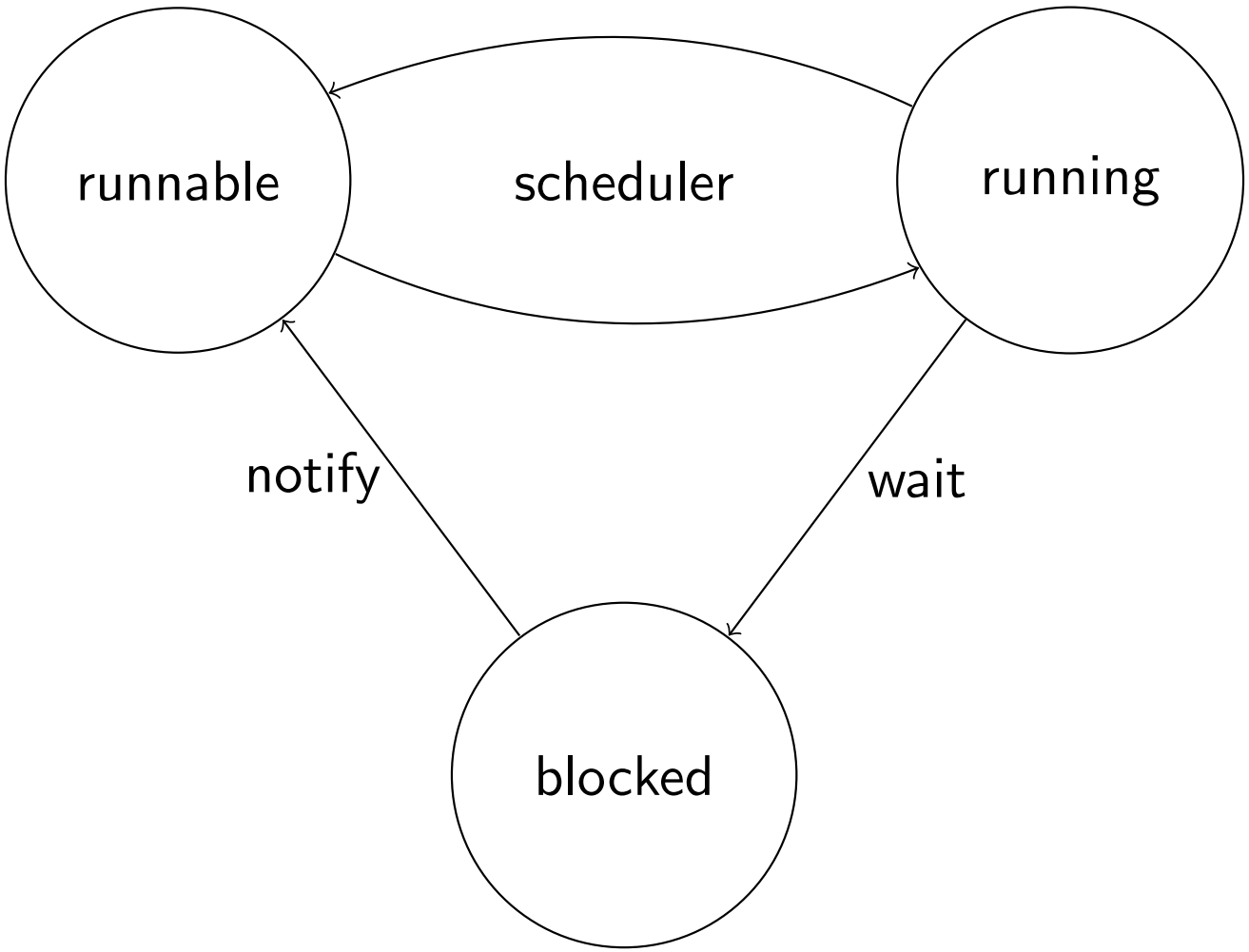
Wait and notify

The `Object` class contains the following three methods:

- `wait`: causes the current thread to wait for this object's lock until another thread wakes it up.
- `notify`: wakes up a single thread waiting on this object's lock; if there is more than one waiting, an arbitrary one is chosen; if there are none, nothing is done.
- `notifyAll`: wakes up all threads waiting on this objects lock.

Since every class extends the class `Object`, these methods are available to every object.

States of a thread



User class

```
public class User extends Thread {
  private Resource resource;

  public User(Resource resource) {
    super();
    this.resource = resource;
  }

  public void run() {
    super.run();
    this.resource.acquire();
    this.resource.release();
  }
}
```

Main method

```
final Resource resource = new Resource();
final int USERS = 2;
final User[] users = new User[USERS];
for (int i = 0; i < USERS; i++) {
  users[i] = new User(resource);
}
for (int i = 0; i < USERS; i++) {
  users[i].start();
}
```

Configuration file

```
target=Main
classpath=<folder that contains Main.class>
listener=listeners.StateSpaceWithThreadInfo
native_classpath=<folder that contains
  listener/StateSpaceWithThreadInfo.class>
```

State space