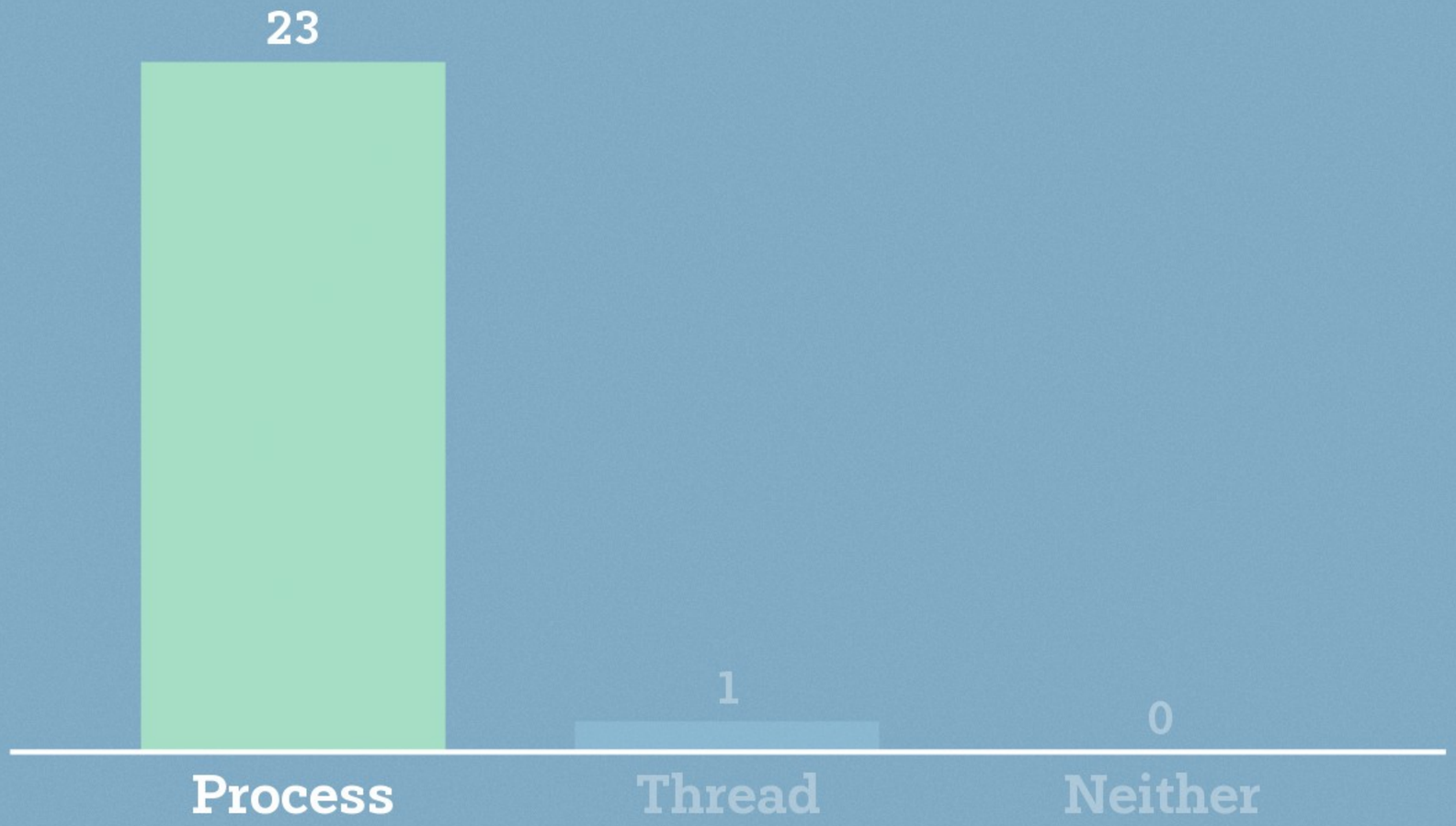
Ask me anything

Mentimeter

6 questions
4 upvotes

Does a Java VM run in a process or a thread?

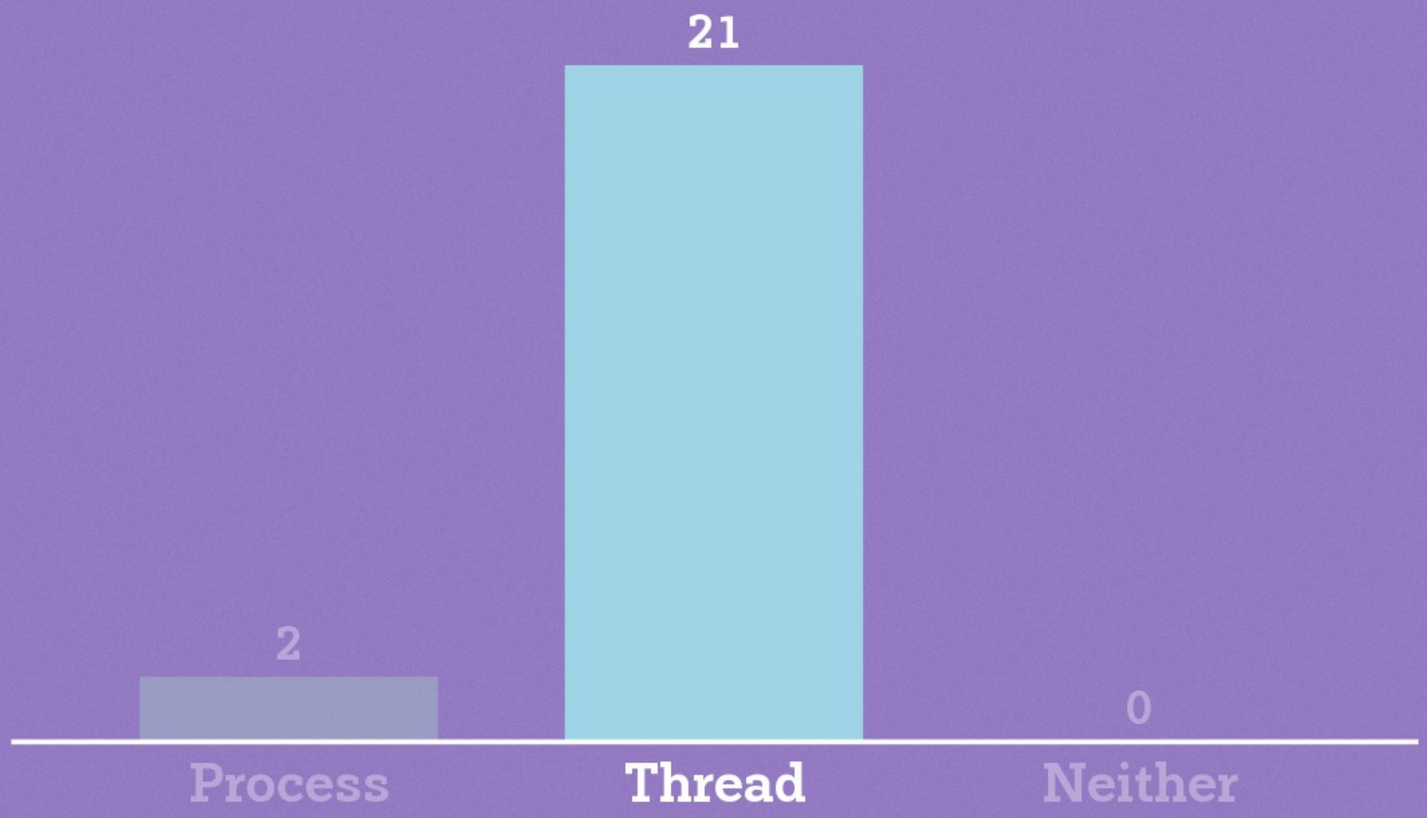
Mentimeter



24

Does a sequential Java program run in a process or a thread?

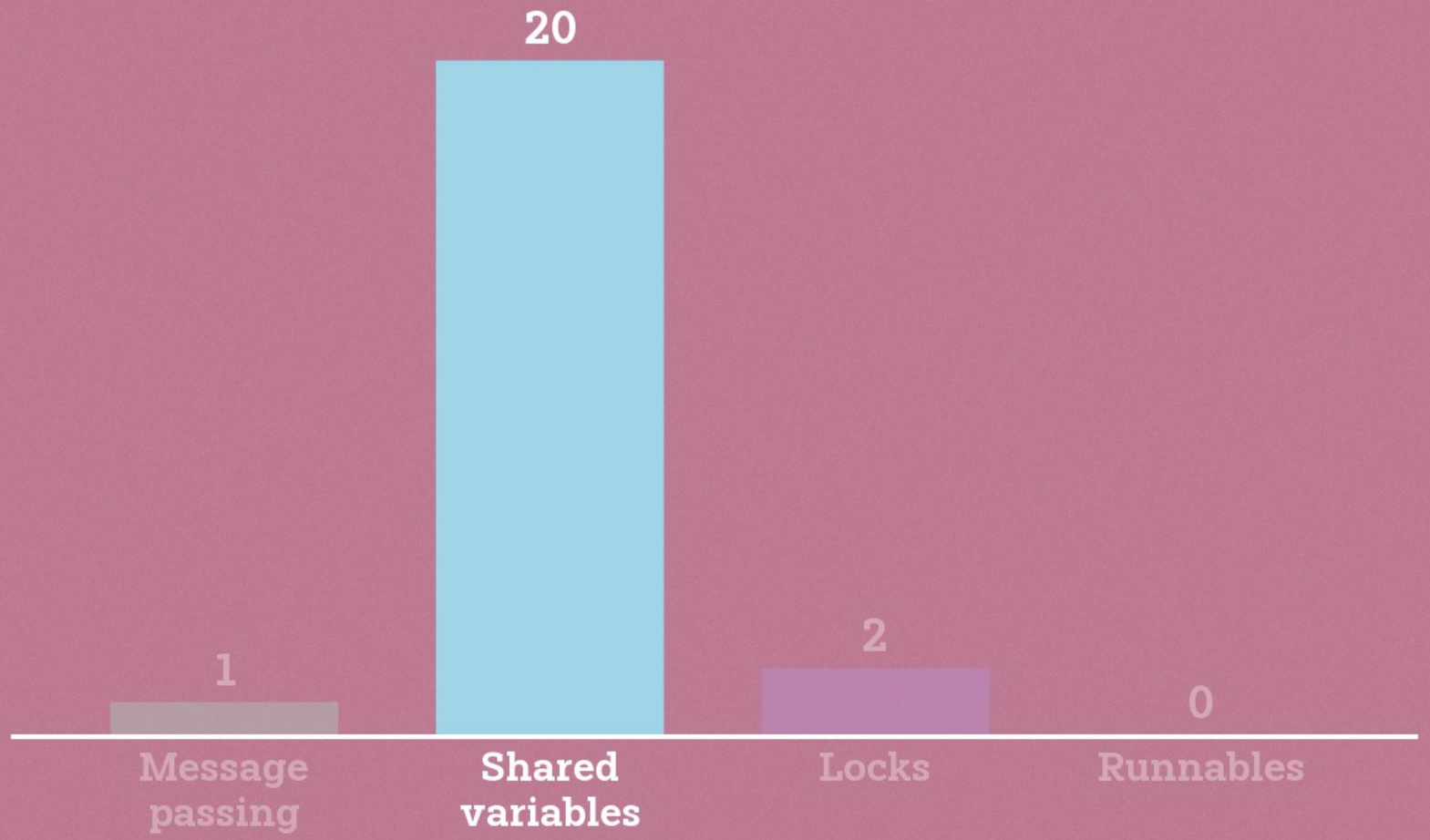
Mentimeter



23

How do Java threads share information?

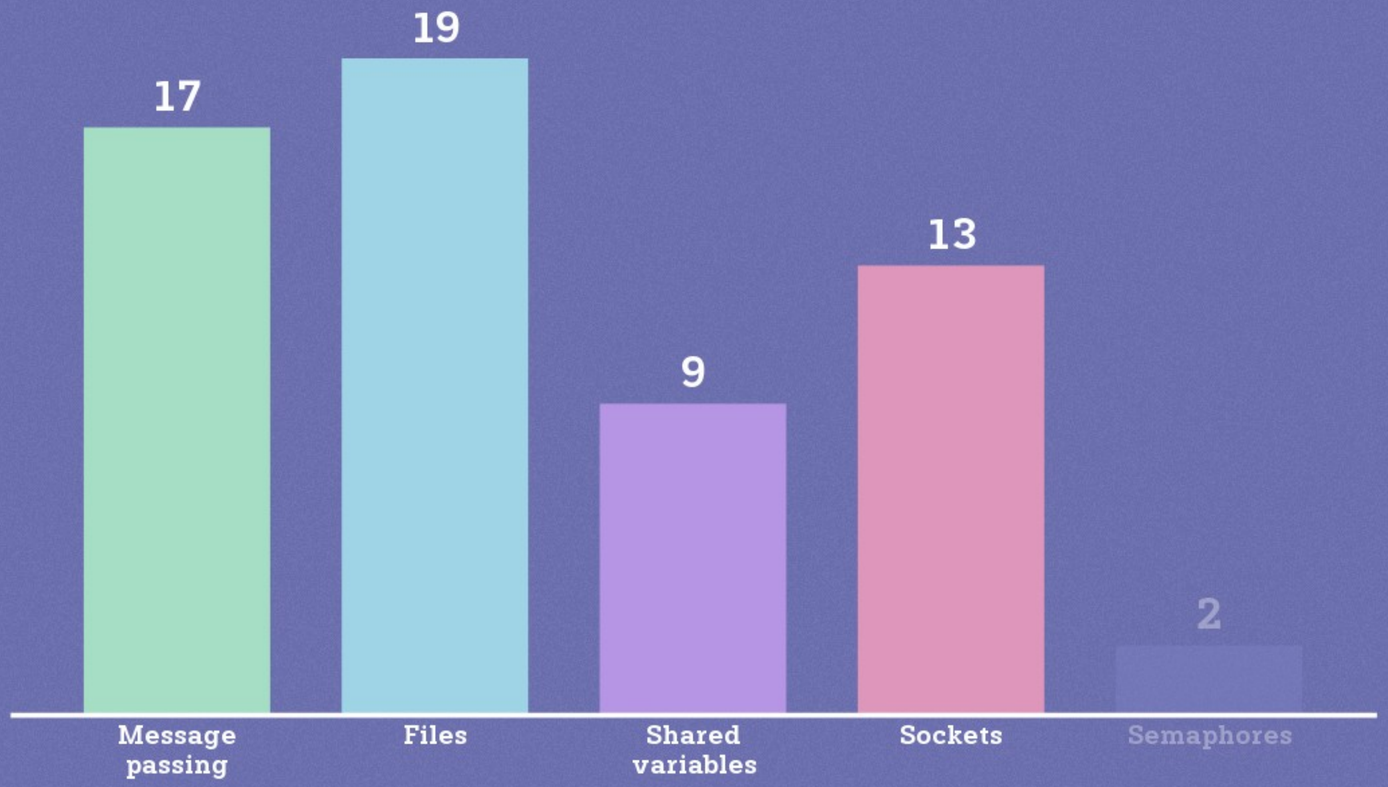
Mentimeter



23

How can OS (Unix) process share information?

Mentimeter



23

What happens if you execute the run() method of a Java Runnable?

Mentimeter

run() is executed once

you should not do that

It'll run in a blocking manner

the method is executed in the current thread (squentially)

The method is executed in the current thread

You will run the run() in the current thread.

It runs in the main thread instead spinning up a new thread

You should start instead

It will run in the current thread (no new thread will be generated)

18

What happens if you execute the run() method of a Java Runnable?

Mentimeter

The run() method is executed in the current thread

the run method is executed within the same thread.

the run method is executed in the current thread. it's just a normal method call.

run is called in the current thread

The method is calles as any other method in java

There are good usecases for both methods.

generally runnable implementation since we can only extend once but implement multiple interfaces

implement runnable because otherwise you can't extend other classes

println is snchronized

18

A Java thread can either subclass Thread or implement Runnable. Which is better, and why?

Mentimeter

implement Runnable to allow the class to extend a different class (there's only single inheritancein java)

Implementing Runnable allows you to inherit from another superclass (if you want)

Interface all the way

implement Runnable, so we may inherit from another class

implement Runnable

Implement runnable, you can still inherit from other classes and composition is more flexible than inheritance

Runnable as in Java it is only possible to inherit from one class

Implementing the runnable because then we are not limited by inheritance from the Thread clss

Runnable is more flexible as Java has single inheritance. So inheriting Thread would prevent it from inheriting anything else.

13

A Java thread can either subclass Thread or implement Runnable. Which is better, and why?

Use Runnable usually, unless you actually want to modify basic thread behaviour, because then you can extend another usefull class if you want to

Both are valid. But you might have to extend another class, in that case implementing Runnable will allow you to do so.

depends on the usecase, if we need inheritence we implement runnable, if we build a separate class we can extend thread

depends on the objective. If we want to inherit from some other class (that doesn't inherit Thread) then we might use the interface runnable

ThreadDemo: Why are the print lines of the Hare and Tortoise never interleaved?

Mentimeter

```
/Users/oscar/Library/Java/JavaVirtua
0 Tortoise
0 Hare
1 Tortoise
1 Hare
2 Tortoise
2 Hare
3 Tortoise
3 Hare
4 Hare
4 Tortoise
5 Hare
5 Tortoise
6 Hare
6 Tortoise
7 Hare
7 Tortoise
8 Tortoise
8 Hare
9 Hare
9 Tortoise
DONE: Tortoise
DONE: Hare

Process finished with exit code 0
```

println is an atomic action

Because printline of system.out is synchronized

public void println(...) { synchronized(this) {...}}

Because the println statement is synchronized

println() has the synchronized keyword

println is synchronized

becuase a print is atomic

Because the JVM protects the output buffer from these two threads (they can not access the output buffer simultaneously)

Because writing a single line is an atomic action

12

ThreadDemo: Why are the print lines of the Hare and Tortoise never interleaved?

Mentimeter

```
/Users/oscar/Library/Java/JavaVirtua
0 Tortoise
0 Hare
1 Tortoise
1 Hare
2 Tortoise
2 Hare
3 Tortoise
3 Hare
4 Hare
4 Tortoise
5 Hare
5 Tortoise
6 Hare
6 Tortoise
7 Hare
7 Tortoise
8 Tortoise
8 Hare
9 Hare
9 Tortoise
DONE: Tortoise
DONE: Hare

Process finished with exit code 0
```

Writing lines to STDOUT is likely ensured to be atomic.

println is synchronized

synchronized constructor calls only make sense if you sideffect in constuctors which is generally a very bad idea

12

What is strange about this code?

Mentimeter

```
synchronized (new Lock()) {
  ...
}
```

The object we synchronize on is generated each time, so this is useless

there is no reference to the lock, which makes it useless as it can not be mutated

everytime you call the method a new Lock is created, which defeats its purpose

why would you use a lock with synchronized, if you can just lock the lock itself (which is what a lock is for...)

We can't use this object for another synchronized block.

A new lock is instatiated every time the code execution reaches the synchronized block

We are trying to protect the lock?

synchronized anonymous instanciation calls only make sense in side-effecting consturctors on eg an abstract class which is a very bad idea anyway.

It's Java.

10

What is strange about this code?

Mentimeter

We aren't attaching an object when declaring the "New Lock()"

```
synchronized (new Lock()) {
  ...
}
```

10

Why is it an error to call wait() outside a synchronized method or block?

It is unclear what it should wait for.

For what is it waiting then?

it'll keep waiting.

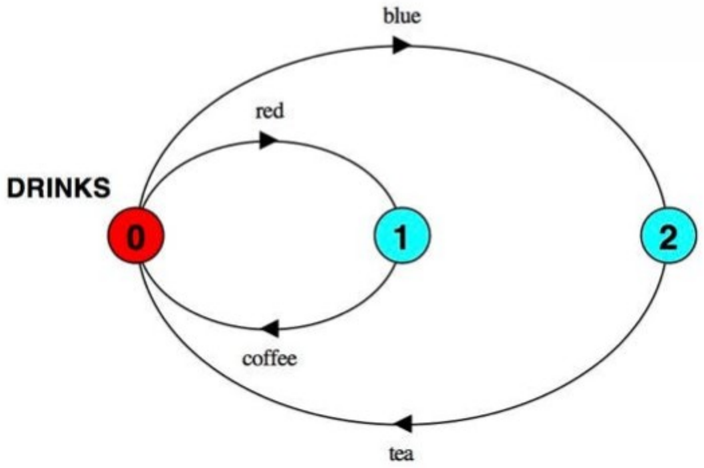
cause there is no resource to be released.. so that makes no sense!

There won't be anything waking the thread afterwards

What are the traces of the DRINKS process?

Mentimeter

```
DRINKS =
 ( red  -> coffee -> DRINKS
 | blue -> tea    -> DRINKS
 ).
```



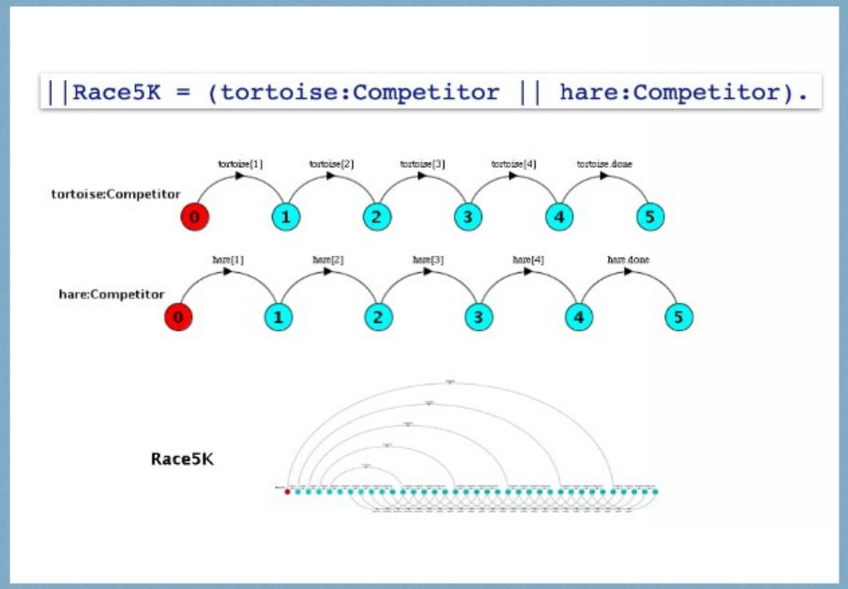
((red cofffee) | (blue tea))*

((red,coffe)| (blue,tea))^\omega

2

How many possible states are there for the Race5K?

Mentimeter



36

36

36

10-choose-5 (as in binomial coefficient)

6^2

36

36

36

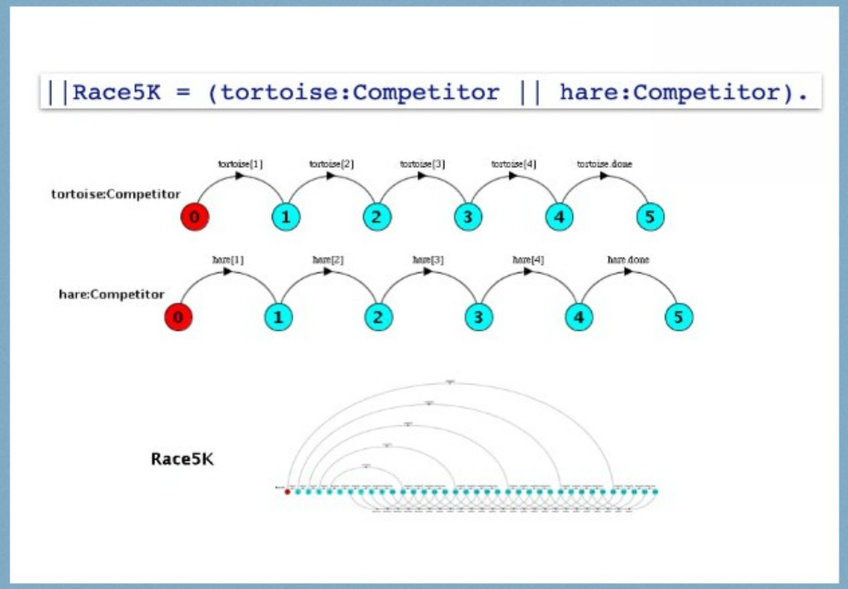
36

10

How many possible states are there for the Race5K?

Mentimeter

the cartesian product (6*6)

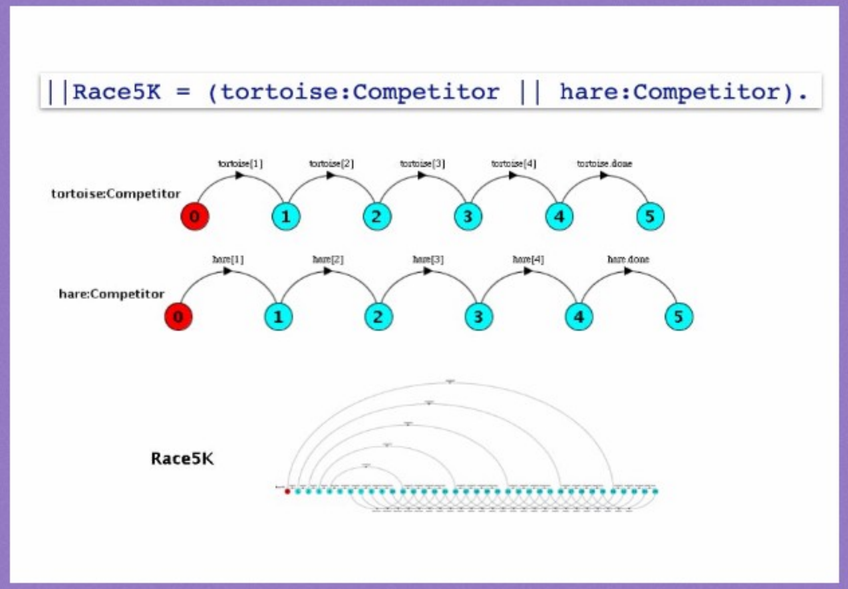


10

How many possible traces are there? (Interleavings of tortoise and hare.)

Mentimeter

Oops, 10-choose-5 here. Was one too early ;)



1

Last chance for questions