

# Implementing Concurrency in Java (Java SE 11 Developer Certification 1Z0-819)

---

Introduction to Concurrency and Threads



**Maaike van Putten**  
Software Developer & Trainer

[www.brightboost.nl](http://www.brightboost.nl)



# Overview



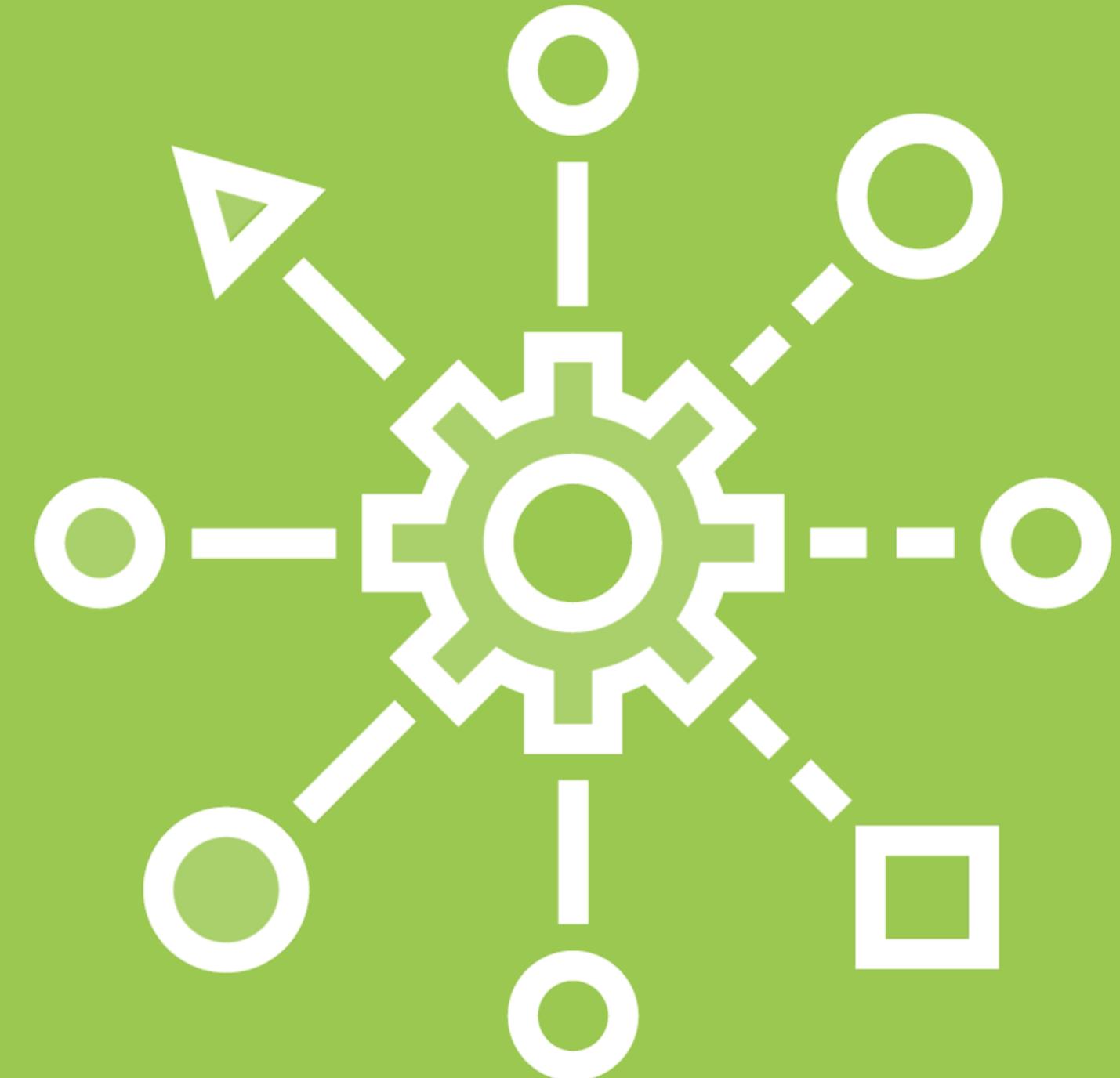
## Concurrency and threads

### Creating threads

- Using the Thread class
- Runnable
- Callable

### Sleep and interrupt





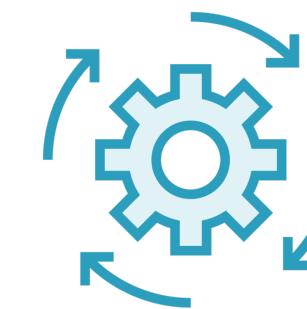
## Concurrency

Executing different paths of instructions at the same time.

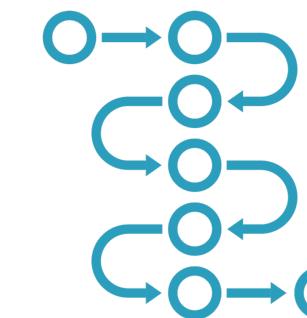


# Different Types of Concurrency

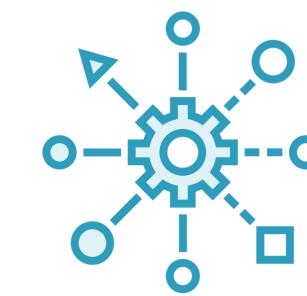
**Concurrency does not  
(necessarily) mean parallel  
execution**



**Multiprocessing – multiple  
CPUs**



**Multitasking – one CPU  
alternating between tasks**



**Multithreading – different  
parts of the same program  
using different threads**



**Computer running  
multiple programs**

**IDE listening to input and  
executing code**

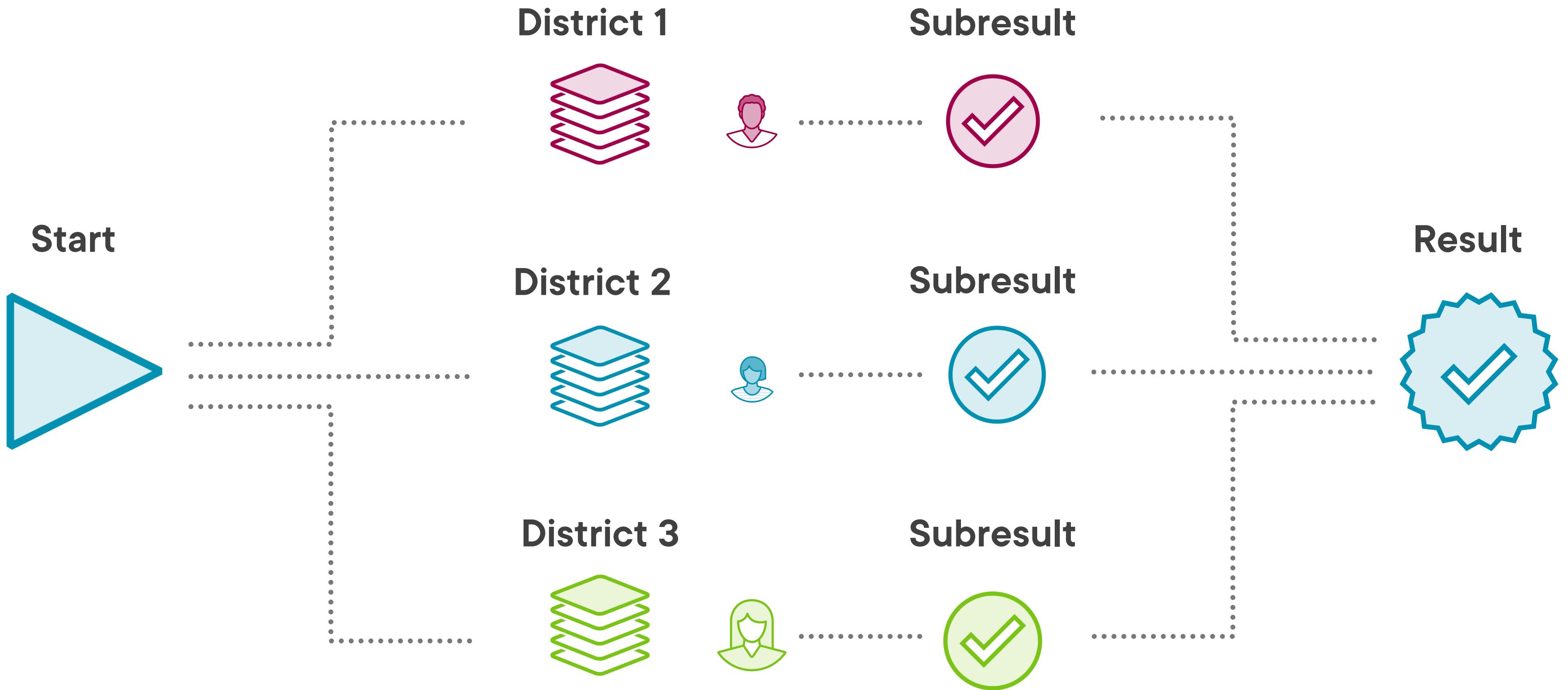
**A web server handling  
many requests at the  
same time**



# Counting Votes without Concurrency



# Counting Votes with Concurrency



# Concurrency: Pros and Cons

Pros	Cons
Decreasing waiting and response time	Shared resources need to be handled carefully
Optimal usage of resources	Managing data integrity can be more difficult
Higher efficiency	Managing memory is more difficult
Meeting user expectations	Concurrency problems: deadlock, livelock, race condition
Overall: improved performance	Switching threads comes with a cost





# What Are Threads?

**Execution of instructions**

**Smallest unit of execution**

**Threads perform tasks**

**Processes consist of multiple threads**



# Demo



**Create a new instance of the Thread class**  
**Start the Thread**

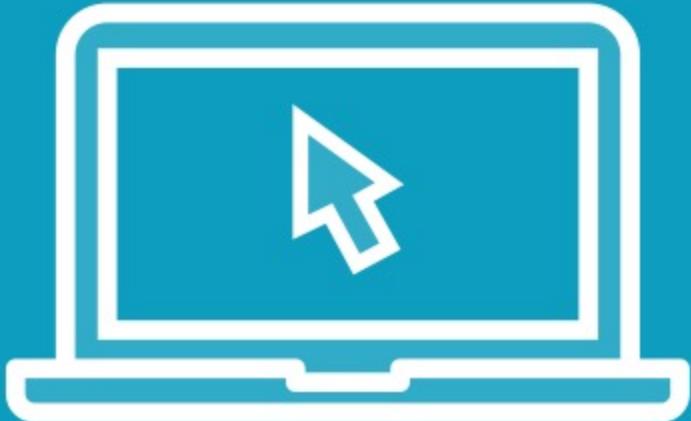


# Careful! What Happens Here?

```
public class RiddleThread extends Thread {  
    public void run(){  
        System.out.println("I'm the task of CustomThread!");  
        System.out.println("The thread id in task: " + Thread.currentThread().getId());  
    }  
  
    public static void main(String[] args) {  
        RiddleThread ct = new RiddleThread();  
        ct.run();  
        System.out.println("The thread id in main: " + Thread.currentThread().getId());  
    }  
}
```



Demo



**Create a Thread extending Thread**  
**Start the Thread**



Demo



**Implement Runnable**

**Start the Thread**

**Implement Callable**



# Runnable vs Callable

<b>Runnable</b>	<b>Callable</b>
<b>Functional interface</b>	<b>Functional interface</b>
<b>Represents a task to be executed by a thread</b>	<b>Represents a task to be executed by a thread</b>
<b>Abstract method run</b>	<b>Abstract method call</b>
<b>Returns void</b>	<b>Returns specified generic type</b>
<b>Cannot throw exception</b>	<b>Can throw exception</b>
<b>Submit returns Future with null</b>	<b>Submit returns Future with generic type</b>



# Sleep and Interrupt



**Threads can go to sleep...**



**And then be interrupted.**



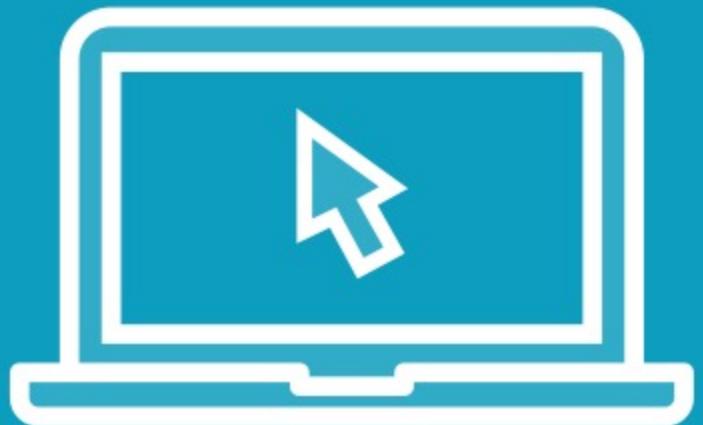


# Join

**Wait for another thread to complete  
Indefinitely or a certain amount of time  
.join()**



Demo



**Sleep**

**Join**

**Interrupt**

**InterruptedException**



Up Next:  
Synchronized and Locks

---



# Synchronized and Locks

---



**Maaike van Putten**  
Software Developer & Trainer

[www.brightboost.nl](http://www.brightboost.nl)



# Overview



## Thread interference

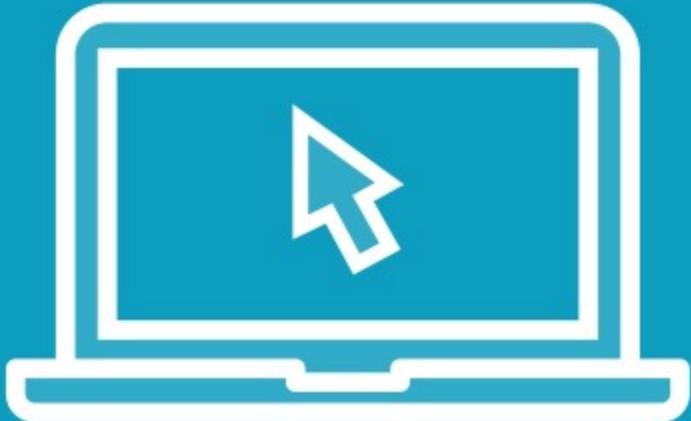
### Synchronized

- Methods
- Objects

### Lock framework



Demo



**Let's count to 10 concurrently**  
**What's going on?!**



# Synchronized Keyword



**Only one thread at a time**



**Methods can be synchronized**



**Objects (locks) can be synchronized**



# Demo



**Let's fix our problem with the synchronize keyword**

- On method
- Lock object



# Limitations of Synchronized Keyword



- Threads are waiting**
- No way to check whether the lock is available**
- If lock remains, waiting thread waits forever**





# ReentrantLock Interface

Protect a piece of code using the ReentrantLock interface and use more advanced functions.



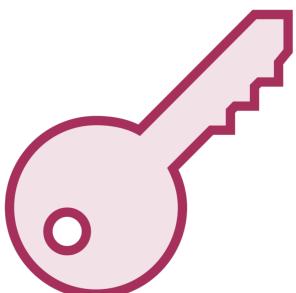
# Methods on ReentrantLock Interface



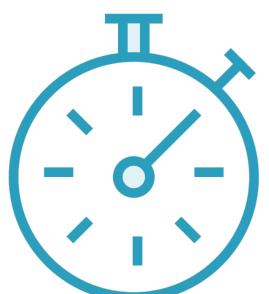
**void lock()**



**void unlock()**



**boolean tryLock()**



**boolean tryLock(long, TimeUnit)**



Demo



**Let's fix our counting concurrently  
problem with ReentrantLock**



Up Next:  
ExecutorService and Thread Pools

---

