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

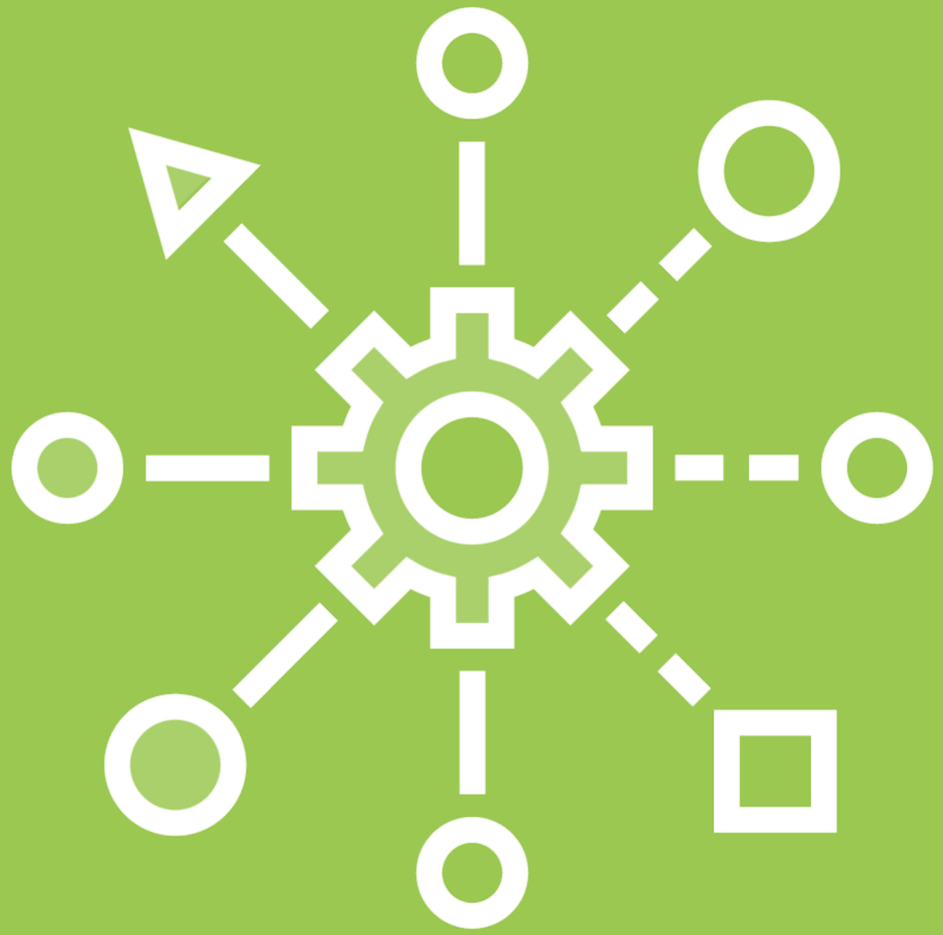
Overview

Concurrency and threads

Creating threads

- Using the Thread class
- Runnable
- Callable

Sleep and interrupt

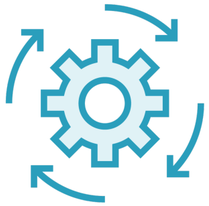


Concurrency

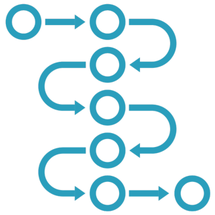
Executing different paths of instructions at the same time.

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

Concurrency: Pros and Cons

Pros

Decreasing waiting and response time

Optimal usage of resources

Higher efficiency

Meeting user expectations

Overall: improved performance

Cons

Shared resources need to be handled carefully

Managing data integrity can be more difficult

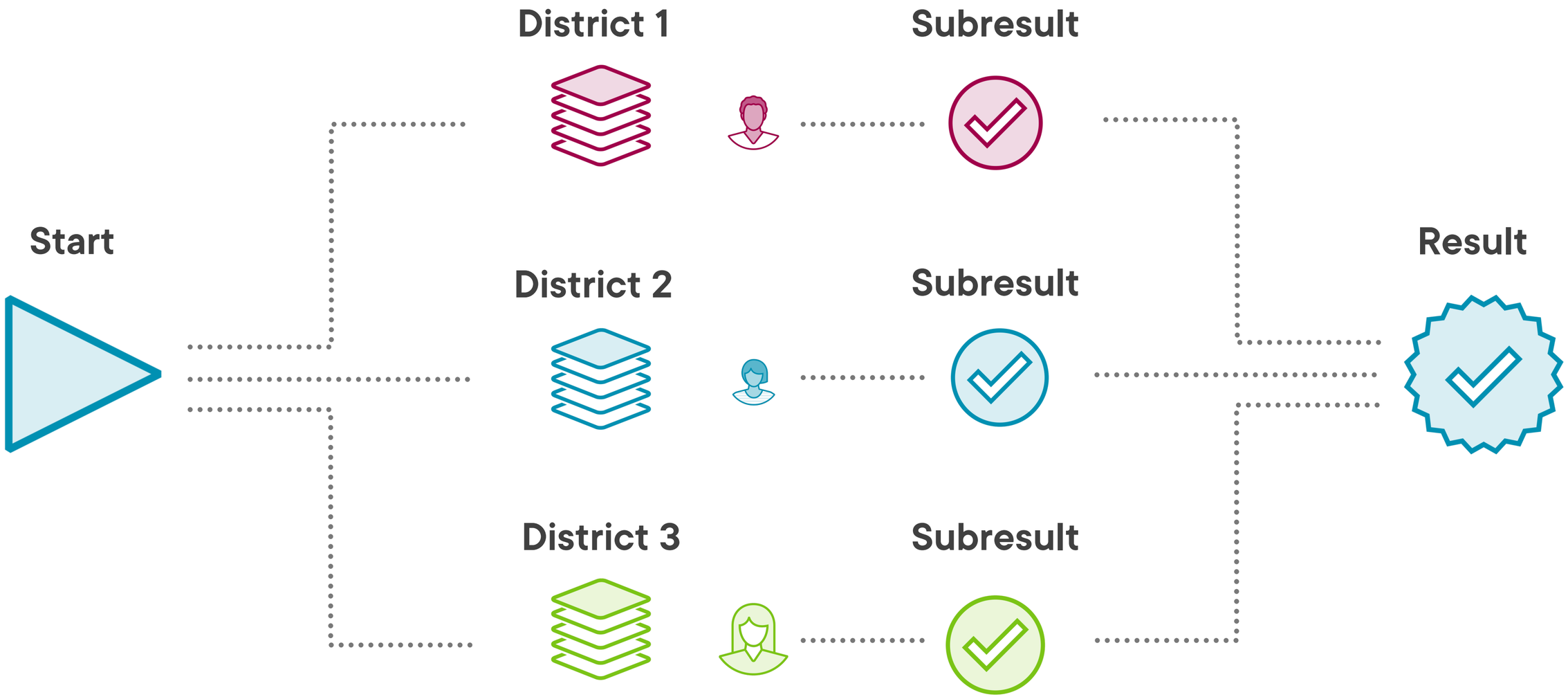
Managing memory is more difficult

Concurrency problems: deadlock, livelock, race condition

Switching threads comes with a cost

Counting Votes without Concurrency

Counting Votes with Concurrency



What Are Threads?

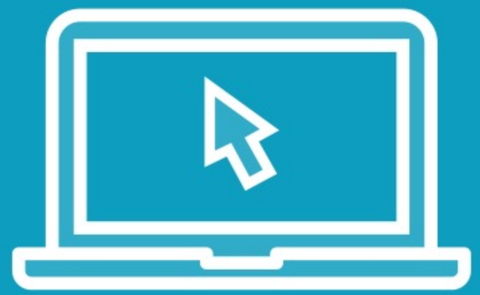
Execution of instructions

Smallest unit of execution

Threads perform tasks

Processes consist of multiple threads

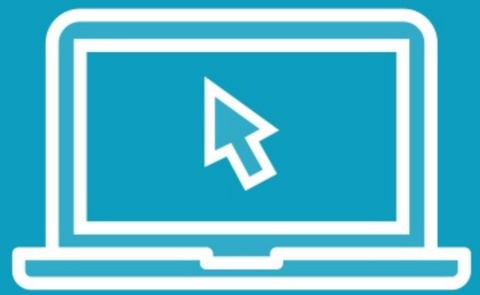
Demo



Create a new instance of the Thread class

Start the Thread

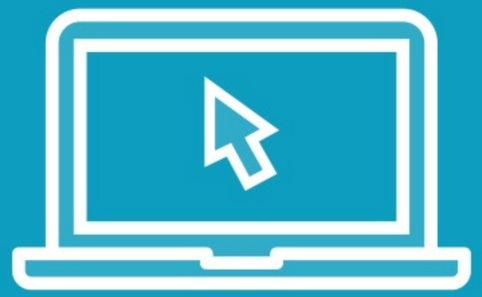
Demo



Create a Thread extending Thread

Start the Thread

Demo



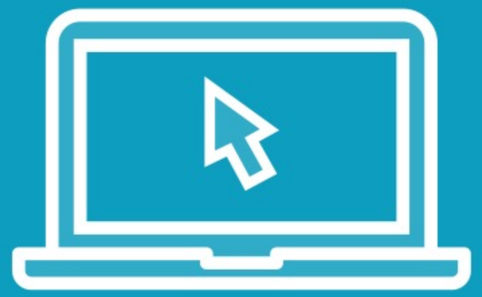
Create a Thread using Runnable

Start the Thread

Careful! What Happens Here?

```
// call run instead of start
<div class="row carousel-indicators">
    <div style="background-color:red;" class="col-md-4" data-target="#homeCarousel" data-slide-to="0" class="active">
    </div>
    <div style="background-color:green;" class="col-md-4" data-target="#homeCarousel" data-slide-to="1"
class="active">
    </div>
</div>
```

Demo



Let’s see how to create a Thread using Callable

Start the Thread

Runnable vs Callable

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

Sleep and Interrupt

Threads can go to sleep...

And then be interrupted.

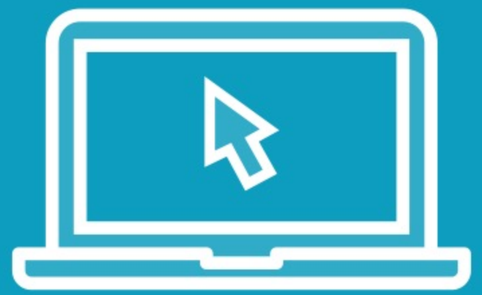
Join

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

Overview

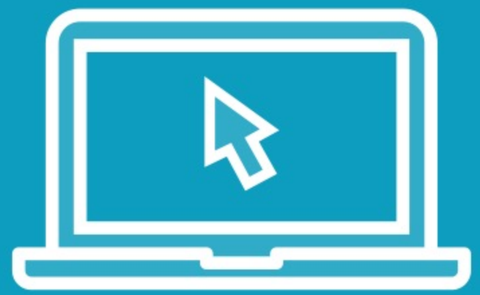
Thread interference

Synchronized

- Methods
- Objects

Lock framework

Demo



Let’s count to 10 concurrently

What’s going on?!

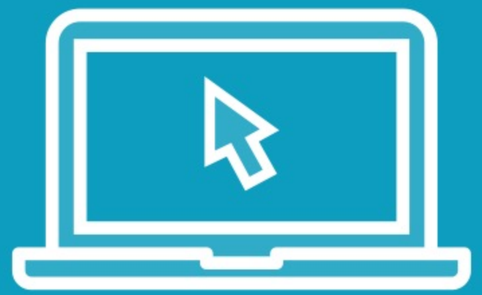
Synchronized Keyword

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 on Objects

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

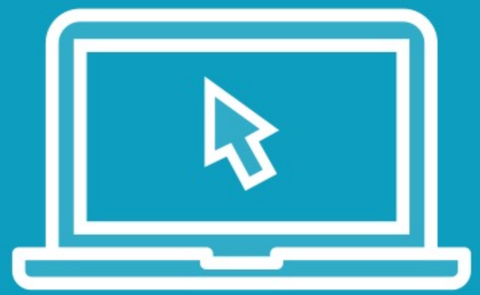
void lock()

void unlock()

boolean tryLock()

boolean tryLock(long, TimeUnit)

Demo



Let’s fix our counting concurrently problem with ReentrantLock

Up Next:
ExecutorService and Thread Pools