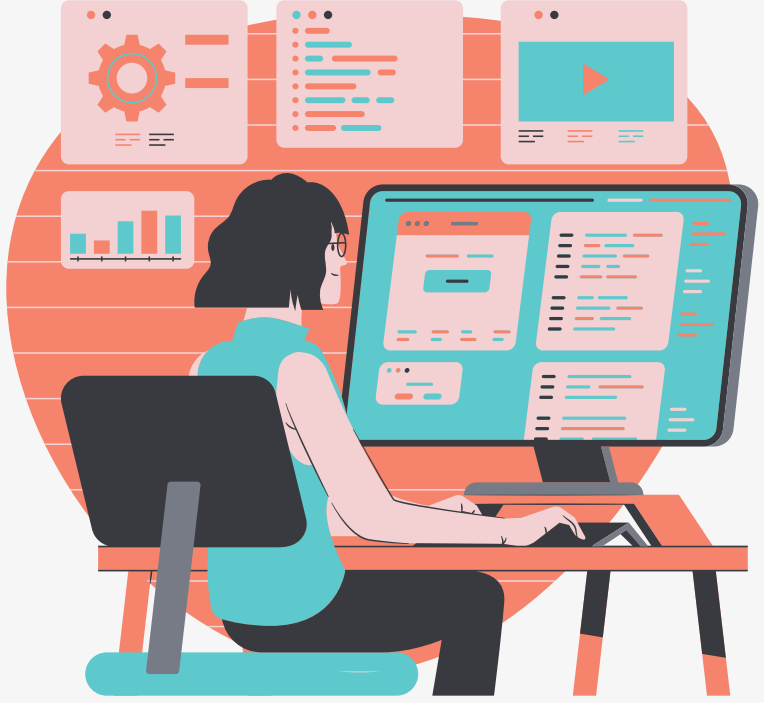

DEMYSTIFYING MULTITHREADING IN JAVA 21

Utkarsh Gupta

Creating Threads

Platform Thread

```
Thread platformThread = Thread.ofPlatform().start(() -> {
    // Code to run in the platform thread
});
```

Virtual Thread

```
Thread virtualThread = Thread.ofVirtual().start(() -> {
    // Code to run in the virtual thread
});
```

Structured Concurrency

```
void myMethod() {
    Thread.startVirtualThread(() -> {
        // Code to run in a virtual thread,
    automatically managed within this method
    });
}
```

Thread.Builder

```
Thread thread = Thread.ofPlatform()
                      .name("MyCustomThread")
                      .daemon(true)
                      .start(() -> {
                          // Code to run in the thread
                      });
```

Waiting for Threads

```
// Create virtual threads (or use Thread.ofPlatform() for platform threads)
Thread thread1 = Thread.ofVirtual().start(() -> {
    // Task 1 code
});

Thread thread2 = Thread.ofVirtual().start(() -> {
    // Task 2 code
});

// Wait for both threads to finish using join()
try {
    thread1.join();
    thread2.join();
} catch (InterruptedException e) {
    System.out.println("Thread interrupted");
}

// Proceed with code that depends on both threads being finished
System.out.println("Both threads have finished.");
```

Advanced Features

- **Thread.onSpinWait:** Hints the JVM to use a spin-wait loop for active waiting.
- **VirtualThread.interrupted:** Checks for virtual thread interruption.

Important Considerations

- Virtual threads are experimental in Java 21.
- Choose thread types based on your application's needs and resource constraints.
- Handle thread synchronization and coordination carefully to prevent race conditions and deadlocks.
- Consider using structured concurrency for clearer and more manageable code.

LIKE

&

FOLLOW

for more