23 Concurrency

Objectives

In this chapter you'll:

- Understand concurrency, parallelism and multithreading.
- Learn the thread life cycle.
- Use `ExecutorService` to launch concurrent threads that execute `Runnable`s.
- Use `synchronized` methods to coordinate access to shared mutable data.
- Understand producer/consumer relationships.
- Use JavaFX's concurrency APIs to update GUIs in a thread-safe manner.
- Compare the performance of `Arrays` methods `sort` and `parallelSort` on a multi-core system.
- Use parallel streams for better performance on multi-core systems.
- Use `CompletableFuture`s to execute long calculations asynchronously and get the results in the future.

Outline

    1. 23.2.1 *New* and *Runnable* States
    2. 23.2.2 *Waiting* State
    3. 23.2.3 *Timed Waiting* State
    4. 23.2.4 *Blocked* State
    5. 23.2.5 *Terminated* State
    6. 23.2.6 Operating-System View of the *Runnable* State
    7. 23.2.7 Thread Priorities and Thread Scheduling
    8. 23.2.8 Indefinite Postponement and Deadlock

3. 23.3 Creating and Executing Threads with the `Executor` Framework
4. 23.4 Thread Synchronization
    1. 23.4.1 Immutable Data
    2. 23.4.2 Monitors
    3. 23.4.3 Unsynchronized Mutable Data Sharing
    4. 23.4.4 Synchronized Mutable Data Sharing—Making Operations Atomic
5. 23.5 Producer/Consumer Relationship without Synchronization
6. 23.6 Producer/Consumer Relationship: `ArrayBlockingQueue`
7. 23.7 (Advanced) Producer/Consumer Relationship with `synchronized`, `wait`, `notify` and `notifyAll`
8. 23.8 (Advanced) Producer/Consumer Relationship: Bounded Buffers
9. 23.9 (Advanced) Producer/Consumer Relationship: The `Lock` and `Condition` Interfaces
10. 23.10 Concurrent Collections
11. 23.11 Multithreading in JavaFX
    1. 23.11.1 Performing Computations in a Worker Thread: Fibonacci Numbers