НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ПОЛТАВСЬКА ПОЛІТЕХНІКА
ІМЕНІ ЮРІЯ КОНДРАТЮКА

ЗБІРНИК МАТЕРІАЛІВ

77-Ї НАУКОВОЇ КОНФЕРЕНЦІЇ ПРОФЕСОРІВ, ВИКЛАДАЧІВ, НАУКОВИХ ПРАЦІВНИКІВ, АСПІРАНТІВ ТА СТУДЕНТІВ УНІВЕРСИТЕТУ

16 травня – 22 травня 2025 р.

УДК 004.415

В. В. Загнойко, асистент
Я. А. Мурас, студентка групи 103-ТН
Національний університет
«Полтавська політехніка імені Юрія Кондратюка»

ВИКОРИСТАННЯ JAVA У СТВОРЕННІ БАГАТОПОТОКОВИХ ЗАСТОСУНКІВ

Багатопоточність є надзвичайно важливою при створенні високопродуктивних та масштабних застосунків. Java — одна з найпопулярніших мов для створення багатопотокових програм: має вбудовану модель потоків, забезпечує високий рівень абстракції для роботи з конкурентністю, постійно вдосконалюється. Вона з самого початку мала підтримку багатопоточності. Через свої потужні засоби для паралелізму та асинхронності, вона надала широкий інструментарій для масштабних та складних проєктів.

Базові механізми багатопоточності з'явилися ще у JDK 1.0. Спершу з'явилося створення потоків за допомогою інтерфейсу Runnable. Сам інтерфейс дуже простий, описує лише один метод (run), який автоматично викликається Java під час запуску потоку. Зазвичай використовується разом із класом Thread, за допомогою наслідування якого можна теж створювати потоки. З'являються різні модифікатори, ключові слова. Одним з таких є synchronized, який використовується для забезпечення контролю доступу до методу чи блоку коду. Він гарантує те, що тільки один потік може виконувати код одночасно, що є важливим для збереження цілісності даних. Чимось схожим до попереднього є volatile, який використовується для позначення того, що значення змінної буде змінене різними потоками і ця змінна буде завжди видима для інших потоків, запобігаючи проблем з кешуванням потоків. Для керування потоками були створені методи wait та notify. Попередньо наведений функціонал, який був впроваджений ще на початку, є базовим, однак, може бути джерелом багатьох помилок, якщо їх використовувати без глибокого розуміння.

З виходом Java 5 з'явився пакет java.utill.concurrent який серйозно полегшив роботу розробникам. Тепер у доступі був наявний інтерфейс ExecutorService, який керує пулами потоків. Його методи керували завершенням роботи та повертали результат. Інтерфейси Callable та Future слугували засобами для отримання результату з асинхронних задач. Вони стали альтернативою для Runnable та Thread. Для синхронізації було введено Semaphore, який ставав необхідним коли мова йшла про обмеження доступу до деякого загального ресурсу. Синхронізатор CountDownLatch надавав можливість потокам очікувати поки не виконаються операції в інших потоках, перед тим як вони були пропущені.

Починаючи з Java 8, акцент зроблено на декларативному та асинхронному програмуванні. CompletableFuture був створений як гнучкий інструмент для побудови асинхронних обчислень. Parallel Streams – паралельна обробка колекцій. Механізм для скорочення потоків при роботі з великою кількістю даних, який активно використовує ядра процесору, покращує продуктивність вводу/виводу. Функціональні інтерфейси Function, Supplier та Consumer сприяли зменшенню кількості шаблонного коду.

Реактивне програмування — це парадигма програмування, побудована на потоках даних і розповсюдженні змін. Це означає, що має бути можливість легко виразити статичні чи динамічні потоки даних, а реалізована модель виконання буде автоматично розсилати зміни через потік даних. Реактивні підходи дозволяють створювати масштабовані, подійно-орієнтовані системи. RxJava — бібліотека для створення реактивних стрімів з підтримкою backpressure(регулює швидкість роботи виробника даних). Project Reactor, Akka, Vert.x — альтернативні фреймворки для реактивного стилю.

Що з’явилося нового у Java 21? Virtual Threads - легковагові потоки , що масштабуються до мільйонів одиниць. Вони мінімізуються потребу в складних пулах, дозволяють писати синхронний код з продуктивністю асинхронного, що спрощує застосунок та робить його більш продуктивним. Новий підхід до керування потоками Structured Concurrency (preview - режим) вводить ієрархічну структуру виконання задач. Основною ідеєю є те, що всі потоки, створені під час виконання задачі, мають існувати рівно стільки, скільки і сама задача. Він спрощує читабельність, обробку помилок та очищення ресурсів.

Підсумовуючи, можемо сказати, що Java надає потужний набір засобів для створення багатопотокових та конкурентних застосунків: від базових потоків до віртуальних потоків нового покоління. Знання класичних і сучасних підходів дозволяє розробникам будувати безпечні, масштабовані та високопродуктивні застосунки.

Література:

1.Java: The Complete Reference – H. Schildt. (розділ 11 «Multithreaded Programming»).

2. Java Concurrency in Practice – B. Goetz. (розділи 5–6).

3. Effective Java – J. Bloch (розділ 11 «Concurrency», пункти 78–85).

4. Reactive Programming with RxJava – T. Nurkiewicz (розділ 2 «Reactive Programming with RxJava»).