Course : Java, multithreaded programming

Practical course - 4d - 28h00 - Ref. JQT

Multithreaded programming in Java is becoming increasingly important thanks to the widespread use of multiprocessor architectures: it simplifies the design and development of applications with intrinsic parallelism, and offers effective solutions to performance problems.

Teaching objectives

At the end of the training, the participant will be able to:

- ✔ Master multithreaded programming models and the corresponding standard libraries
- ✔ Know the main data structures suitable for multithreaded programming
- ✔ Know the main bugs and get an overview of solutions
- ✔ Knowledge of test and debugging tools
- ✔ Understanding the links between multithreaded programming and performance

Intended audience

Developers, software architects and project managers.

Prerequisites

Basic knowledge of Java and the use of an IDE (Eclipse, IntelliJ...).

Course schedule

1 Multithreaded programming models, interfaces and classes

- Basic concepts: task, execution resource, activity, execution service, future.
- Various concept implementations (Runnable, Callable<T>, ExecutorService, Future<T>...).
- Uncaught exceptions, thread groups.
- The complete future.

Hands-on work
Programming an application combining the different models.

PARTICIPANTS
Developers, software architects and project managers.

PREREQUISITES
Basic knowledge of Java and the use of an IDE (Eclipse, IntelliJ...).

TRAINER QUALIFICATIONS
The experts leading the training are specialists in the covered subjects. They have been approved by our instructional teams for both their professional knowledge and their teaching ability, for each course they teach. They have at least five to ten years of experience in their field and hold (or have held) decision-making positions in companies.

ASSESSMENT TERMS
The trainer evaluates each participant's academic progress throughout the training using multiple choice, scenarios, hands-on work and more.
Participants also complete a placement test before and after the course to measure the skills they've developed.

2 Constraints on the correct behavior of activities

- Some formal specification methods.
- A semi-formal specification method.
- Implementing specifications.

Hands-on work
Use of formal specification methods.

3 Thread synchronization and communication

- Status "synchronized", "wait", "notify" and monitor programming.
- Synchronization interfaces and classes: locks, semaphores, cyclic barriers.
- Cue.

Hands-on work
Use interfaces and synchronization classes.

4 Parallel task execution

- ExecutorService.
- The fork/join model (RecursiveTask<T>, RecursiveAction, ForkJoinPool).

Hands-on work
Use of runtime services and the fork/join model.

5 Data structures dedicated to multithreaded programming

- Specialized collections.
- Local storage of thread data: ThreadLocal<T>.
- Atomic classes.

Hands-on work
Use of data structures.

6 Threads and performance

- The impact of thread creation.
- The impact of synchronization.
- The impact of memory caches.
- Threads and IOs, DBs and graphics.
- Threads and scheduling.

Hands-on work
Optimisation des programmes.

7 Alternative models

- Asynchronous models: JReact.
- Actor models (Akka Actor4J...).
- Synchronous reactive models.

8 **Tools for developing competitive programs**

- JConsole, jstack.
- The JArmus bookshop.
- Lamport's temporal logic: TLA+.
- Common errors: contention, dormancy, deadlock, premature termination.