
Q&A
You have
Questions
We have
Answers

What is concurrency ?

- Run or appear to run several programs or several parts of a program simultaneously.
- Threads → Operate in same address space.
- Java is a multi-threaded language → Thread support for sequential programs.
- Advantages of using Threads
  - Light-weight
  - Flexibility
  - Ease of resource sharing

The Need for Concurrency within a program

- Multi-processor system
  - Faster execution.
- Single processor system
  - Faster execution/performance
    - Sounds counter intuitive. What about the overheads of context switches ?
    - Waiting
      - For Memory, Disk, Network or something else.
      - For User Interaction (Event driven programming).
      - For distributed systems.
  - Improving code design
    - Simulations (games)
    - Actor model

Creating Threads

- Method 1: Implement the Runnable interface

```
public class HelloWorldRunnable implements Runnable {

    public void run() {
        System.out.println("Hello World");
    }

    public static void main(String[] args) {
        new Thread(new HelloWorldRunnable()).start();
    }

}
```

Creating Threads

- Method 2: Extend the Thread class

```
public class HelloWorldThread extends Thread {

    public void run() {
        System.out.println("Hello World");
    }

    public static void main(String[] args) {
        new HelloWorldThread().start();
    }

}
```

Creating Threads

- A thread performs the task defined in the “run” method.
- Implementing Runnable vs Extending Thread
  - Runnable provides purer composition.
  - Extending Thread is useful for simple applications but limited owing to sub-classing.
- User Threads and Daemon Threads.
- JVM runs a process until at least one user thread is running.
- SetDaemon(boolean) function can turn on/off the daemon status of a thread.

Manipulating Threads

- Manipulate execution of threads.
- Thread.yield() signals the scheduler to schedule another thread for execution. Just a hint.
- Thread.sleep() pauses execution of the calling thread.
- t.join() makes the calling thread wait for the thread “t” to exit. Timed versions available.
- setPriority() can set the scheduling priority of a thread.
- MAX_PRIORITY, MIN_PRIORITY, NORM_PRIORITY.

Manipulating Threads

- Sending and handling Interrupts.
- t.interrupt() sends an interrupt to thread “t” for handling. Thread “t” can detect the interrupt by
  - catching “InterruptedException”
    - If you do nothing, set the status back.
    - Thread.currentThread.interrupt().
  - checking status using Thread.interrupted(). Do not check after catching an “InterruptedException”.
- Interrupts are used to cancel a thread. Good programs support thread cancellation.

What can we do

- Parallel version of quicksort.
- Parallel linear search.
- Parallel binary search.
- Situations where threads can work independently.

Using Executors and Threadpools

- Thread creation and destruction are expensive.
- Large scale applications require separate thread management and creation.
- Java.util.concurrent package provides thread management via Executors.
- Executors are a layer of indirection between client and execution of a task.
- Multiple types of threadpools like CachedThreadPool, FixedThreadPool.
- Frees the application from thread-management and focus on resource sharing and co-ordination among threads

Callable and Future

- Callable is an “extension” of the Runnable interface → Represents a task that returns a result or throws Exception.
- Must implement call() method instead of run() which returns a result.
- Is only supported for execution through Executor interface.
- Submitting a task obtains a Future object.
- Calling get() on the object blocks if the thread has not completed.
    - Timed version of get() is available.
    - isDone() is a non blocking version which can check to see if the result is available.

Thread Sharing and Synchronization

- Threads do not always operate on independent resources or wait for other threads to finish.
- With concurrency resource contention and sharing is a problem that needs to be tackled.
- Proper access to resources must be ensured.
- Mechanisms that can be used
  - Monitors
  - Using explicit Lock objects
  - Atomic classes and volatility

Lab sessions

- Assignment 1 will be released.
- Lab sessions (work on assignment) will take place in Lille UP from 13:15 – 17:00.
- Work in groups of 3.
- No need for a report, comment your code :-).
- Deadline 23:59 hrs today.