Multithreaded Programming

By,

Hitha Paulson

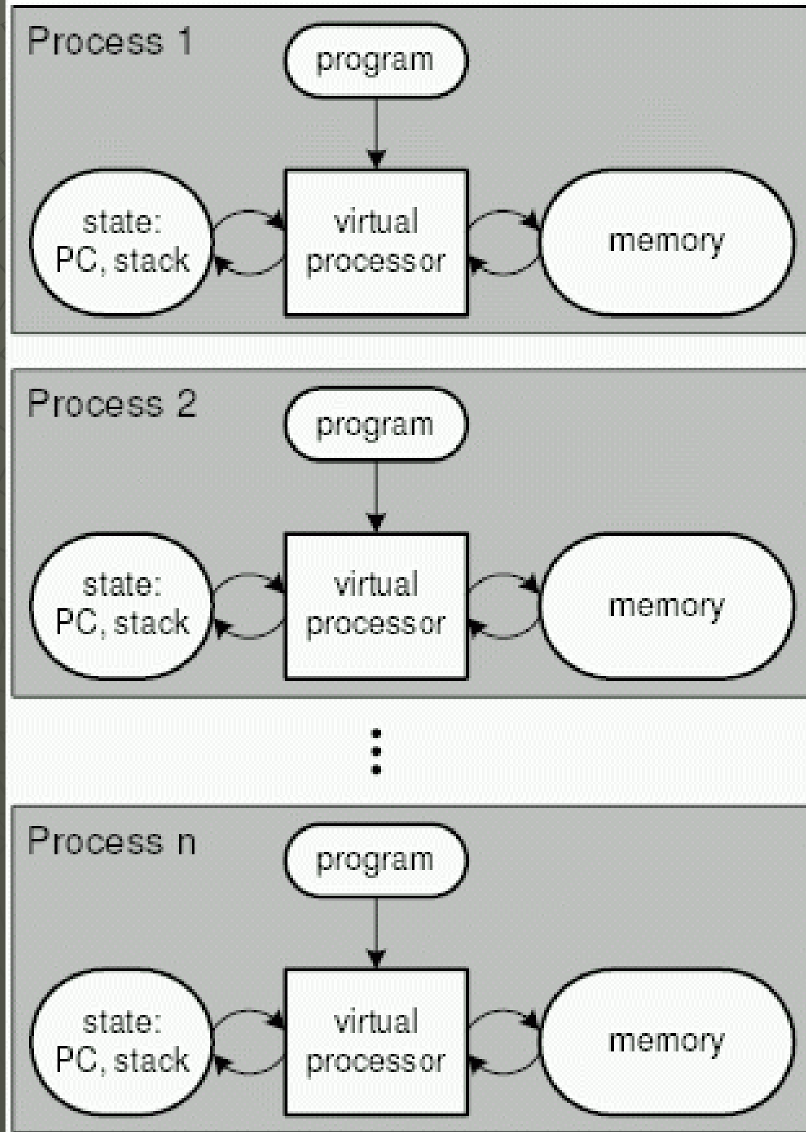
Assistant Professor, Dept. of Computer Science

LF College, Guruvayoor

Multithreading

- Thread
  - Smallest unit of dispatchable code of a program
  - Light weight compared to a process
- Multi-Thread
  - More than one parts of a program that can execute concurrently
  - Specialized form of multitasking
  - Programmer can write code to generate multiple threads

Multiprocessing



Multithreading

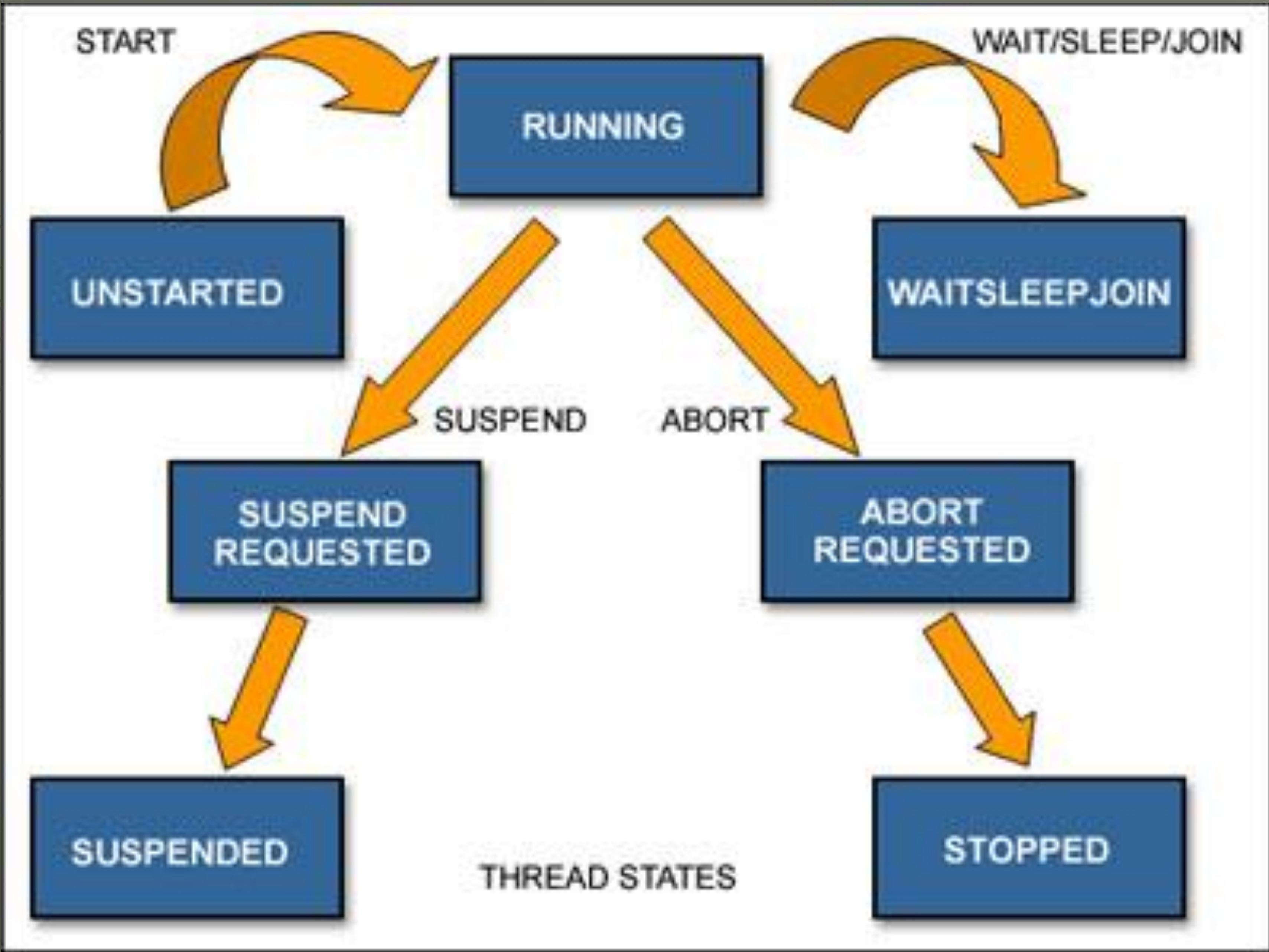
Process 1
program
Thread 1
state:
PC, stack
virtual
processor
Thread 2
state:
PC, stack
virtual
processor
Thread m
state:
PC, stack
virtual
processor
memory
Process n

Multithreading: pros and cons

- Single Threaded Program
  - allows one part of the program can execute at a time
  - It waste CPU time by keeping CPU idle. Eg: Data Read operation
  - When a thread blocks, entire program stops running
- Multithreaded Program
  - Multiple part of the program in execution, thus keeps CPU busy all time
  - Share same address space
  - Inter-Thread communication is inexpensive
  - Low cost context switch

Java Thread Model

- States of Thread
  - Running
  - Ready to run
  - Suspended
  - Resumed
  - Blocked
  - Terminated


START
RUNNING
WAIT/SLEEP/JOIN
UNSTARTED
WAITSLEEPJOIN
SUSPEND
ABORT
SUSPEND REQUESTED
ABORT REQUESTED
SUSPENDED
THREAD STATES
STOPPED

Thread Priorities

- Integers to specify the relative priority of one thread compared to another thread
- It is used to decide when to switch from one thread to next thread (Context switch)
- Rules for context switch
  - A thread can voluntarily relinquish control
  - A Thread can be preempted by a higher priority Thread

Synchronization

- The mechanism prevents the execution of one thread, which affects the execution of another thread
- Protects shared asset being manipulated by more than one Thread at a time
- Statements in different threads executes synchronously.
- Implementation of critical region
- Java uses Monitors to handle Synchronization
- Once a Thread enters a monitor, all other threads must wait until that Thread exits the Monitor

The Main Thread

- Every Java Application program is executing with Thread called MAIN THREAD
- This Thread will automatically create when a program starts executing
- All child threads are created from this Main Thread
- This Thread terminates only after rest of child Threads are terminated
- The default name of this Thread is “main”
- When a thread object is printed, displays: [Thread Name, Priority, Group]
- Thread.currentThread(), getName(), setName(String)

Creating Thread

- Two Methods
  - Use **Thread** Class
  - Use **Runnable** interface
- Using Thread Class
  - Create a class that extended from Thread class
  - Define a method called run() I the format
    public void run()
  - Run() method contain the code that constitutes the thread
  - Run() is like any other method can do anything
  - Run() will act as the entry point for the thread and it will end when run() terminates
  - Thread is invoked by using start() method with object of class extended from Thread class

- Using Runnable Interface
  - Create a class that implements Runnable Interface
  - Redefine the abstract method run(). Eg: *public void run()*
  - Run() method contain the code that constitutes the thread
  - Run() is like any other method can do anything
  - Run() will act as the entry point for the thread and it will end when run() terminates
  - Thread is invoked by using start() method with object of class implemented Runnable interface

isAlive() and join()

- isAlive()
  - used to determine whether a thread has finished or not
  - Returns boolean value to indicate the status
- join()
  - Used to wait for a thread to finish
  - void join() throws InterruptedException
  - Allows to specify maximum amount of time that a Thread should wait for termination

Thread Priority

- Used by the thread scheduler to decide when each thread should be allowed to run
- Higher priority thread will get more CPU time than lower priority threads
- A priority thread can preempt by low priority thread
- Threads having equal priority should get equal access to the CPU

Implementing Thread Priority

- void setPriority(int level)
- Value of **Level** must be within the range MIN_PRIORITY and MAX_PRIORITY
- NORM_PRIORITY used to set/get default priority
- All these are final variables in Thread class
- int getPriority() used to get the priority

Synchronization

- The mechanism used to ensure that the shared resources are accessed by more than one threads in a sequential manner
- No two threads are simultaneously operating upon a shared resource
- Synchronization is implemented by using monitor (also called Semaphore)

Monitor

- A monitor is an object that is used as a mutually exclusive lock or mutex
- Only one thread can own a monitor at a given time
- When a thread acquires a lock,
  - Thread entered the monitor
  - All other threads will suspend as long as the first thread exits the monitor
  - ie) all other threads are waiting for the monitor
  - A locked thread can reenter the same monitor

Implementing Synchronization

- Java uses Synchronized methods, methods declared with synchronized keyword
- All objects have their own implicit monitor associated with them
- Call to a synchronized method invokes the monitor in that object
- While a thread is inside the synchronized method, all other threads that try to call it on the same instance must wait
- Exiting from synchronized method releases the monitor
- Unsynchronized methods causes race condition.

Interthread Communication

- Synchronized methods can communicate each other without using polling
- Methods to support interthreaded communication: wait(), notify(), notifyAll()
- All are final methods defined in Object class
- All methods can call from synchronized context
- wait(): tells the calling thread to give up the monitor and go to sleep until some other thread enters the same thread and calls notify()
- notify(): wakes up a thread that called wait() on the same object
- notifyAll(): wakes up all the threads that called wait() on the same object. One of the thread will granted access
- All the above methods will throw InterruptedException