Lecture 12

Multi-Threaded Programming

References:

1. Herbert Schildt, Chapter 11, The Complete Reference Java 10 edition, McGraw Hill

Topics covered

- Introduction
- The Java Thread Model
- The Main Thread
- Creating Threads
- Synchronization
- Interthread Communication

Multi-threaded Programming - Introduction (1/2)

- Two types of Concurrent Programming
  - Process-based multi-tasking
    - A program as the smallest unit of code
    - Run two or more programs concurrently
    - E.g., run Java compiler and run a text editor

Multi-threaded Programming - Introduction (2/2)

  - Thread-based multitasking
    - A thread as the smallest unit of code
    - A single program performing two or more threads simultaneously
    - E.g., a text editor formatting text and printing it

The Java Thread Model (1/3)

- Synchronization
  - A way to enforce synchronization between threads/processes
    - E.g., two threads/processes to communicate a data
  - Monitor
    - Age-old model of inter-process synchronization

The Java Thread Model (2/3)

- Synchronization in Java
  - No class “Monitor” in Java
  - Instead, each object with its implicit monitor
    - A thread automatically entering to the monitor
      - When an object’s synchronized method is called
    - Once a thread is inside a synchronized method
      - Other threads not to call the synchronized method on the same object

The Java Thread Model (3/3)

- Message communication among threads
  - Once a thread to enter a synchronized method on an object
  - Other threads can call the synchronized method
  - But, wait there until the notification of the entered thread

The Main Thread

- Main thread
  - Important two reasons
    - Can spawn other child threads
    - Must be the last thread to finish execution
      - Because various shutdown actions
  - E.g., CurrentThreadDemo.java Program
    - Thread [main, 5, main]: thread name, priority, a thread group

Creating Threads

- Threads created using Thread class or Runnable interface
  - Thread class
    - run(), getName(), getPriority(), setName(), isAlive(), join(), sleep(), start()
  - Runnable interface
    - run()

Creating Threads with Thread class

- Extending Thread class
  - The extending class must override the run() method
  - run() method
    - The entry point for a concurrent thread
    - Define the thread logic
    - Executed by start() method
  - E.g., ExtendThread.java Program

Creating Threads with Runnable interface

- Implementing Runnable interface
  - Any object implementing Runnable interface
  - Construct a thread in the object
    - Thread (Runnable threadOb, String threadName)
    - threadOb: an instance of a class implementing the Runnable interface
    - Allocate a new thread object to threadOb
  - start() calling run()
  - E.g., ThreadDemo.java Program

Creating Multiple Threads

- Creating Multiple Threads
  - Can spawn as many threads as it needs
  - E.g., MultiThreadDemo.java Program

Checking Thread status and Joining

- Using isAlive() and join()
  - Two ways to determine whether a thread finish
    - final Boolean isAlive()
    - final void join() throws InterruptedException
  - E.g., DemoJoin.java Program

Synchronization

- Synchronization
  - Critical resource used by only one thread at a time
  - *synchronized* keyword
- Using *synchronized* Methods
  - While a thread inside a synchronized method
    - All other threads calling it on the same instance must wait
  - E.g., Synch.java Program

Synchronization

- The synchronized Statement
  - Used for a class objects not designed for multi-thread
  - Synchronized block

```
synchronized(objRef) {
      // statements to be synchronized
}
```

  - E.g., Synch1.java Program

Message Communication between Threads

- Polling wastes CPU cycles
  - producer (sender) and consumer (receiver)
- To avoid polling
  - wait(), notify(), notifyAll():
    - Give up accessing a resource and go to sleep with wait()
    - Until a thread calls notify() or notifyAll()
  - Defined in Object class
    - final void wait() throws InterruptedException
    - final void notify(), final void notifyAll()
  - E.g., PC.java
  - E.g., PCFixed.java

Deadlock in Inter-thread Communication

- Deadlock
  - Two threads with a circular dependency on a pair of synchronized objects
  - Example:
    - Thread A enters the monitor on object X and Thread B enters the monitor on object Y.
    - Thread A tries to enter the monitor on object Y, and Thread B tries to enter the monitor on object X
  - Difficult error to debug