CS F213 - Object Oriented Programming

J. Jennifer Ranjani
email: jennifer.ranjani@pilani.bits-pilani.ac.in
Chamber: 6121 P, NAB
Consultation: Fridays 4 – 5 p.m.

https://github.com/JenniferRanjani/Object-Oriented-Programming-with-Java

BITS Pilani
Pilani Campus

Queries asked during previous class

- Thread groups
  - Thread can be created by mentioning an explicit group name in the constructor (Beyond our scope)
  - If interested, pls chk
    - » https://www.javatpoint.com/threadgroup-in-java

- Relationship between priority and execution frequency
  - It is not necessary that a high priority executes first and ends first.

Queries asked during previous class

- Thread Priorities
  - By default, any child's priority will be that of the parent.
  - Main Thread (5)
    - – Child T1 (5)
      - » Child T2 (5)

- What will join() do?
  - We will definitely see this in detail.

Creating a Thread (Contd.)

It can be created in two ways

- Extending the **Thread** class
    - By creating a new class that extends the **Thread** and then by creating the instance of the class
    - The extending class must override the **run()** method which is the entry point for the new thread

- Call to **start()** begins the executions of the new thread

Choosing an Approach

- It is best to implement Runnable, if we are not overriding any of the other methods by the Thread class .

- When you inherit Thread class if will not be allowed to extend any other class.

Thread Priority

- Priority is represented by a number between 1 and 10

- 3 Priority constants are defined in Thread class
  - public static int MIN_PRIORITY - 1
  - public static int NORM_PRIORITY   - 5
  - public static int MAX_PRIORITY - 10

- Methods
  - final void setPriority(int level)
  - final int getPriority()

Thread Priorities

- In theory, threads run concurrently.
    - In practice, most computers have a single CPU.
    - Threads run one at a time, giving an illusion of concurrency.

- **Scheduling –** execution of multiple threads in some order.

- Java runtime supports fixed priority scheduling and it is also preemptive.

Thread Priorities…

- Runtime system chooses the highest priority thread for execution.

- The scheduler chooses the highest priority thread among the available threads for running.

- This thread runs until,
    - A thread with higher priority comes
        - it preempts the other threads and become runnable.
    - It yields or its run() exists.
    - OS supports time slicing, its time allotment has expired.

Equal Priority threads

- Multi-tasking will be implemented by each OS differently
  - Time slicing with Round robin
  - First come first serve
- For safety, threads should yield the control once in a while.
  - This ensures that every thread gets a chance in non-preemptive environment
- Practically, threads do get a chance to run because threads encounter blocking due to i/o etc.
- Don’t rely on OS scheduling capabilities.

Using isAlive() & join()

- Mostly we want the main thread to finish last
- It is accomplished by calling sleep() within main() with a long delay to ensure that all the child threads are terminated prior to the main thread
- Question: How will main know when the child terminates?
- isAlive() – determines whether a thread has finished; returns true is the thread is still running
- join() – this method waits until the thread on which it is called terminates.
  - Maximum amount of time we want a thread to wait can also be specified.