Thread Priorities

CSCI 201L

Jeffrey Miller, Ph.D.

jeffrey.miller@usc.edu

HTTP://WWW-SCF.USC.EDU/~CSCI201
Outline

- Thread Priorities
Thread Priorities

- Threads have priorities and can get set using `setPriority(int)` and can be retrieved with `getPriority()`
- Priorities are given values from 1 to 10
  - `Thread.MIN_PRIORITY` is 1
  - `Thread.NORM_PRIORITY` is 5
    - Default priority for main thread
  - `Thread.MAX_PRIORITY` is 10
- The JVM tries to execute the highest priority thread that is in the ready state
  - Higher priority threads will be chosen based on a probabilistic algorithm
    - A thread with a priority of 10 is twice as likely to be chosen as a thread with a priority of 5
- There is still a chance that lower priority threads will execute even when a higher priority thread is ready because the higher priority thread does get time-sliced out of the CPU while it is running
  - Another thread then can be switched into the CPU
public class Test {
    public static void main(String[] args) {
        System.out.println("First line");
        TestThread ta = new TestThread('a');
        TestThread tb = new TestThread('b');
        TestThread tc = new TestThread('c');
        tb.setPriority(Thread.MAX_PRIORITY);
        ta.start();
        tb.start();
        tc.start();
        System.out.println("Last line");
    }
}

class TestThread extends Thread {
    private char c;
    public TestThread(char c) {
        this.c = c;
    }
    public void run() {
        for (int i = 0; i < 20; i++) {
            System.out.print(i + " + c + " );
        }
        System.out.println(" ");
    }
}
Starvation

- If there is a higher priority thread that never yields or sleeps, the lower priority thread *may* never execute
  - This is called *starvation* or *contention*
- This can be seen in the infamous Dining Philosopher problem
Dining Philosopher Problem

- Originally proposed by Dijkstra in 1965
- There are five philosophers sitting around a table, each with a plate in front of them
- There is one chopstick placed between each of the plates, so there are only five chopsticks on the table
- Philosophers only do two things – think and eat
  - When they aren’t thinking, they are eating
  - When they aren’t eating, they are thinking
- To eat, they need both of the chopsticks surrounding their plate
- With each philosopher operating independently and not communicating, how can you make sure the philosophers don’t starve?
Dining Philosopher Problem
Outline

- Threads
- Program
Program

- Create an instance of the Dining Philosopher problem with five threads all sharing five static variables. Implement one of the solutions discussed in class.