CISC 3320 Homework 2 Description and Assumptions

Michael Rocco

Assignment 2 is to demonstrate a basic understanding of mutithreading. It does this by taking the program from assignment 1 and modifying it to create 100 threads that each allocate a PID using allocate\_pid, waits a random time, and releases that PID with release\_pid. To prevent race conditions allocate\_pid() and release\_pid() are now synchronized because BitSets are not multithreading safe.

To simplify the PID functions an internal protected static class of the program’s main class will now be created to hold the BitSet that PIDs are stored in. This avoids the unnecessary complication of passing the BitSet’s address to each thread.

Java has a built in concurrency library called java.util.concurrent. This library is used to implement an ExecutorService that manages a set number of threads. For the purpose of this assignment 100 threads are created. Multiple threads are created using a Runnable class that follows the functions laid out in the assignment. A PID is assigned in main right after the threads are created. The main program waits until all threads are finished using TimeUnit.MILLISECONDS.sleep(101). The PID made in main is tested and released and a new PID is made to make sure all threads finished. The executor service must be shut down using shutdown() or it will continue running.

The thread prints the PID it is assigned to console. In java sleeping inside a thread is accomplished using the Thread.sleep() method. It accepts a long for the maximum time it sleeps in milliseconds. Randomness is accomplished using Math.Random(), multiplying the value by 100 and converting to a long. The thread then prints when it released the PID it is using. For simplicity’s sake the console output is used for demonstration. This will vary between program executions because all threads have a random wait and are run concurrently.