CIS 36B : Prof. Abboud

Lab 8

Hamer, Jenny

Lab Assignment 8: Multithreaded Programming

1. How does Java’s multithreading capability enable you to write more efficient programs?

The reason for which multithreading allows for great programming efficiency is that one can have multiple actions be performed by separate threads. This means that idle time – time spent waiting for various I/O, for instance – can be used to perform additional tasks. (Text example: “while one part…is sending a file over the Internet, another part can be reading keyboard input, and still another can be buffering the next block of data to send”(436).)

2. Multithreaded programming is supported by the **Thread** class and the **Runnable** interface. (implement the **Runnable interface** or extend the **Thread class**).

3. When creating a runnable object, why might you want to extend **Thread** rather than implement **Runnable**?

Extending Thread might be used if it’s desired to override or change certain aspects or functionality of the Thread class itself, as opposed to simply implementing the built-in functionality. When extending Thread, one must override the run() method; as well as call start() to begin the execution of a new thread.

4. Show how to use **join()** to wait for a thread object called **MyThrd** to end.

MyThrd.join();

5. Show how to set a thread call **MyThrd** to three levels above normal priority.

(call) setPriority(3);

6. What is the effect of adding the **synchronized** keyword to a method?

Using the synchronized keyword in a method allows coordination of two or more tasks or activities. The most common application is to synchronize two (plus) threads to a jointly shared resource that can only be accessed by a single thread at any given time. This is to hold a thread in a suspended state if need be while waiting for an event to happen.

7. The **wait()** and **notify()** methods are used to perform interthread communication. This allows two objects that depend upon each other for resources or some other access in a multithreaded environment to suspend one thread while waiting for information / execution of the other. If one thread is unavailable and the other requires some data from it, instead of the latter tying up the object and preventing other threads from running, it waits for the first to finish its process and is notified when it can resume running.