1) General Project Description

My project fills a good portion of the requirements given in the description, leaving out only 5 of them that I did not have time for.

Program Files - I have 5 Program templates that each use a different ratio of Calculate and I/O operations as well as having random memory and time requirements for each instruction, within a given threshold. Once given a number of processes to produce, the code randomly decides which template to use for each process

Scheduling - I chose 2 scheduling algorithms that the user can choose between, Shortest job first and Priority Scheduling. The user decides which to use after deciding how many processes to create.

Process States - The code uses all of the 5 states throughout the creating, scheduling, and running of each process.

Memory - The program uses both virtual and physical memory to allocate and run the processes, only processes that can fit in memory can be executed. There is also a LRU algorithm implemented to keep old pages from clogging up memory, this also includes a second chance algorithm to make sure new pages don't get switched off before they get to be used.

I/O - The program has a certain percent chance to create an I/O interruption between starting processes and allocating the memory for new ones.

Multi-threading - each process and each of their children are a seperate thread.

2) When you start the program, it will ask how many processes you would like to create. Once you give it a number, it will ask which scheduling algorithm you would like to use, and then create and schedule the processes. Finally it will ask you if you want to go ahead and run the processes. If you enter Y, it will run all of the processes in order of the scheduling algorithm, only running the ones that there are space for, but no more than 50 at a time. It will print out when each process is complete as well as tell you if it was a child or not. Then it will ask if you would like to quit, N lets you create more processes and Y ends the program. None of the memory information prints out, but if you use breakpoints in the dispatcher class, you can see that it is allocating and deallocating memory to specific processes based on their pID.
3) **- Requirement #1** Process implementation and PCB, the Process.java and ProcessControlBlock.java classes
   - Requirement #2 Critical Section within each process, the ProcessGenerator.java class lines 84-92, 131-139, and 159-164
   - Requirement #3 Critical Section resolving Scheme, the Dispatcher.java class lines 157-172 and 187-200
   - Requirement #4 Single inter-process communication method, The Dispatcher.java class lines 114-126, the ProcessControlBlock.java class lines 52-61, and the PipeReaderProcess.java class
   - Requirement #5 Single level parent-child relationship, The Dispatcher.java class lines 104-112, 135-182, and the ChildProcess.java class

- Requirement #6 Two inter-process communication methods, The Dispatcher.java class lines 128-133, The ProcessControlBlock.java class lines 63-73, the SocketServerProcess.java class, and the lines from Requirement #4
- Requirement #7 Multi-level parent-child relationship, The Dispatcher.java class, lines 104-107, and lines from Requirement #5
- Requirement #8 Scheduler, The Scheduler.java class, and the ProcessGenerator.java class lines 101-106
- Requirement #9 Two schedulers and comparison, the lines from Requirement #8 and the OperatingSystemRunner.java class lines 15-18
- Requirement #10 Process Priorities, ProcessGenerator.java class line 95, and the Scheduler.java class lines 35-55
- Requirement #14 Basic memory and operations on it, The Page.java class, the Frame.java class, the VirtualMemory.java class, the PhysicalMemory.java class, the memoryManagementUnit.java class, and the Dispatcher.java class lines 222-243
- Requirement #15 Memory divided into hierarchy the lines from Requirement #14
- Requirement #16 Paging, the lines from Requirement #14
- Requirement #17 Virtual memory with paging The page.java class, the frame.java class, the VirtualMemory.java class, and the Dispatcher.java class lines 222-234
- Requirement #18 I/O interrupts and handlers, the Dispatcher.java class lines 75-77, and the InterruptHandler.java class
- Requirement #19 Multithreading via hardware, most of the code deals with this, but mainly the Dispatcher.java class lines 41-88
- Requirement #23 Loading external processes and generating new ones on user request, the OperatingSystemRunner.java class calls all of the methods to create and run processes and lets you keep going as long as you wish

4) **- Extra Requirement #1** Second Chance Algorithm, MemoryManagementUnit.java class lines 22-63, and the Frame.java class lines 5 and 9