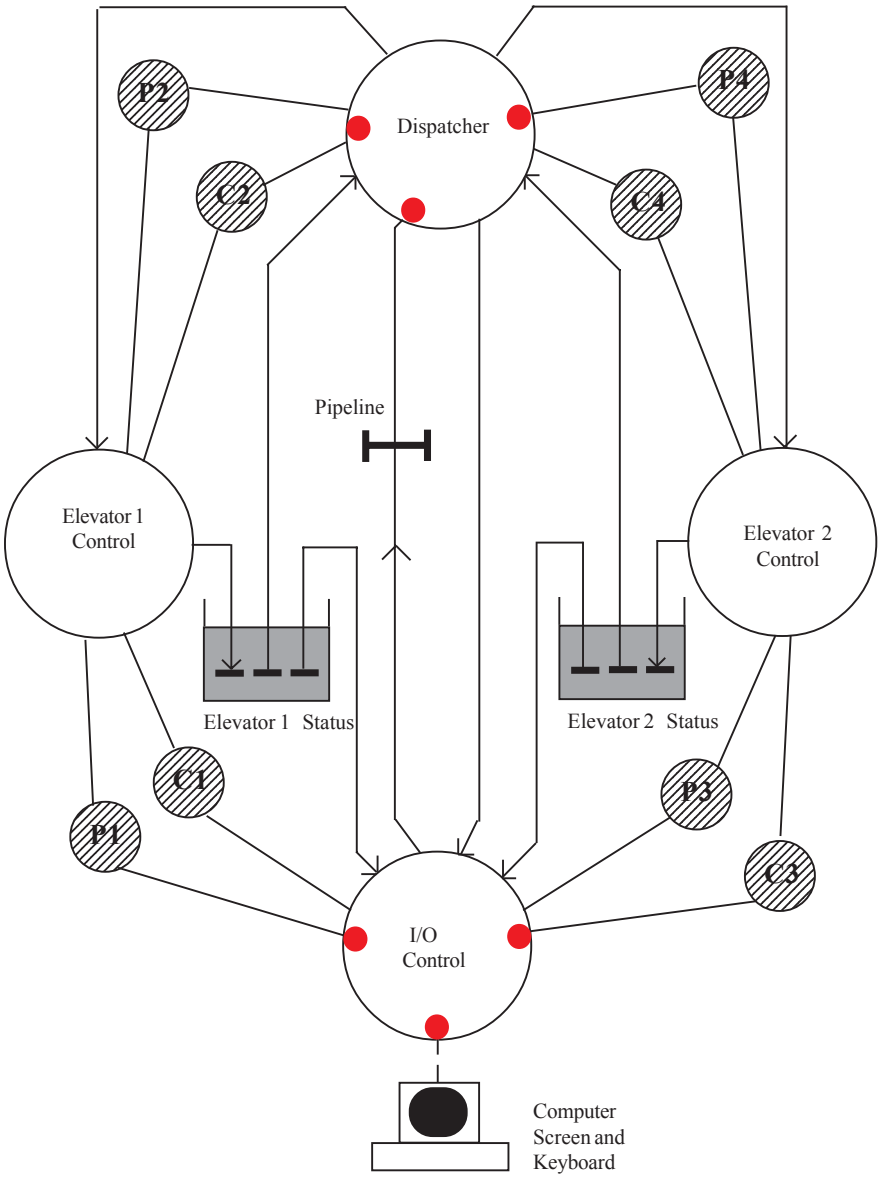
Duplex Elevator System Architecture



Dispatcher, IO and the dual elevators are implemented as 4 ACTIVE CLASSES within a single process/project.

Disptcher can send floor requests to an elevator directly by invoking a member function in the elevator objects depending upon the command entered into the simulation

Connection between Dispatcher and IO can be used to send messages to the IO, such as terminate yourself when the end of the simulation is required

IO Class should have three threads in total, one to deal with input from the keyboard and the other two to deal with updating the display in response to updates from one or other of the two elevators, that is, there are two threads dealing with updating the display and each deals with one elevator. Each should redraw the screen in response to its corresponding elevator updating the datapool and signalling its corresponding producer semaphore (Note you will probably need to implement a mutual exclusion mechanism to ensure the both threads do not try to move the cursor and print to the screen at the same time, i.e. a mutex will be needed to protect the screen from being manipulated concurrenly by two unsynchronised threads)

Dispatcher should also have three child threads, one each for the producer consumer arrangement inherent in the elevators and one to deal with reading commands from the pipeline

Elevators only have one thread each

Pipeline between IO and Dispatcher allows incomming commands that have been validated by the IO process to be sent to the Dispatcher, where upon the dispatcher decides which elevator to delegate the floor request to

Rendezvous objects, conditions, events etc can be used at your discretion to synchronise processes/threads, as can the use of timers and other messages, plus of course the use of threads/active objects (do not feel you have to include any or all these features, but this architecture does not rule them out)

Data pools are updated by each elevator when their status changes, e.g. direction, door status (open/closed), status (in/out of service etc) or floor number changes.

This information is available to the dispatcher so that it can decide which elevator is in the best position to deal with an incoming request. It's also available to the IO so that it can update the window to show the elevator status to the user running the simulation.