**Project 4 -**

Extend project 3 to include making jobs wait until people with the resources required by the job are available at the port.

Elaboration:

1. Reading Job specifications from a data file and adding the required resources to each Job instance.
2. Resource pools - SeaPort.ArrayList <Person> list of persons with particular skills at each port, treated as resource pools, along with supporting assignment to ships and jobs.
3. Job threads - using the resource pools and supporting the concept of blocking until required resources are available before proceeding.
4. The Job threads should be efficient:
   1. If the ship is at a dock and all the people with required skills are available, the job should start.
   2. Otherwise, the Job should not hold any resources if it cannot progress.
   3. Use synchronization to avoid race conditions.
   4. Each Job thread should hold any required synchronization locks for a very short period.
   5. When a job is over, all the resources used by the job (the people) should be released back to the port.
   6. When all the jobs of a ship are done, the ship should depart the dock and if there are any ships in the port que, one of then should should be assigned to the free dock, and that ships jobs can now try to progress.
   7. NOTE: If a job can never progress because the port doesn't have enough skills among all the persons at the port, the program should report this and cancel the job.
5. GUI showing:
   1. Resources in pools - how many people with skill are currently available
   2. Thread progress, resources acquired, and resources requests still outstanding