Resource Scheduler Supporting Document

1. A text file with a running commentary of what you are doing at each step and your thinking behind each key decision (like the conversation you'd have while writing it with someone else present or pair-programming)
2. A solution that you would consider "production quality"
3. All source and project files required to build and run the task (an eclipse project would ideal)
4. All the tests written to support the work (we would love to see evidence of Test Driven Development!)
5. Any other design work you undertook

# Running Commentary

1. First opinions of the problem were that certain required functionality would cause trouble since designing them in after the fact would be harder. These areas are main:
   1. Dynamic Resourcing
   2. Selectable Prioritisation Strategy
2. How to do dynamic resourcing?
   1. Threads are my first thought.
      1. A thread per resource so the work would only be done as stated if there was a resource that was not busy.
3. How to manage the threads?
   1. I thought that the java.util.concurrent library has a lot of useful elements in there for managing pools of jobs with variable number of threads so I looked there.
   2. I found the ThreadPoolExecutor which sounded ideal for my purposes since it had a simple interface allowing for a configurable size thread pool which would each then run an instance from a blocking queue of runnables.
   3. I created a small test which trailed the executor out.
4. How to allow for different forwarding schemes?
   1. I decided to split my design into two main parts.
      1. Resource Scheduler
         1. The resource scheduler part was the idea that this would contain any state that necessary.
         2. Not currently required but most definitely would be if the resources were real.
      2. Forwarding Strategy
         1. Contains an interface that all strategies would obey.
            1. Accepting messages
            2. Cancel
            3. Shutdown
            4. Wait on shutdown
         2. The forwarding