Resource Scheduler

Current System

You are working on a system which uses a single, very expensive, external/3rd party resource to perform some, potentially very time consuming, operations on messages that you send to it. You are supplied with the Gateway and Message interfaces describing how to interact with the external resource:

- send messages to be processed by calling the Gateway's send(Message msg) method:
    - public interface Gateway
        - public void send(Message msg)
- when a Message has completed processing, its completed() method will be called:
    - public interface Message
        - public void completed()

Task

The number of these external resources has just been increased to allow more messages to be processed. However, as these resources are very expensive, we want to make sure that they are not idle when messages are waiting to be processed. You should implement a class or classes that:

- can be configured with the number of resources available
- receives Messages (and queues them up if they cannot be processed yet)
- as available resources permit (or as they become available), sends the 'correct' message to the Gateway

Selecting the right message

- Messages to the Gateway have a logical grouping and several Messages form a "group" (messages have a group ID).
- Messages are not guaranteed to be delivered in their groups. I.E. you might get messages from group2 before you are finished with group1
- Where possible, the message groups should not be interleaved...except where resources are idle and other work can be done.
- The priority in which to process groups is defined by the order in which you receive the first message from the group

This is captured as a set of ***behaviors*** below:

Forwarding

The class you write must forward Messages via the Gateway interface when resources are available:

- For a single resource, when one message is received, that message is sent to the gateway
- For two resources, when two messages are received, both messages are sent to the gateway

Queuing

When no resources are available, messages should not be sent to the Gateway

- For a single resource, when two messages are received, only the first message is sent to the gateway

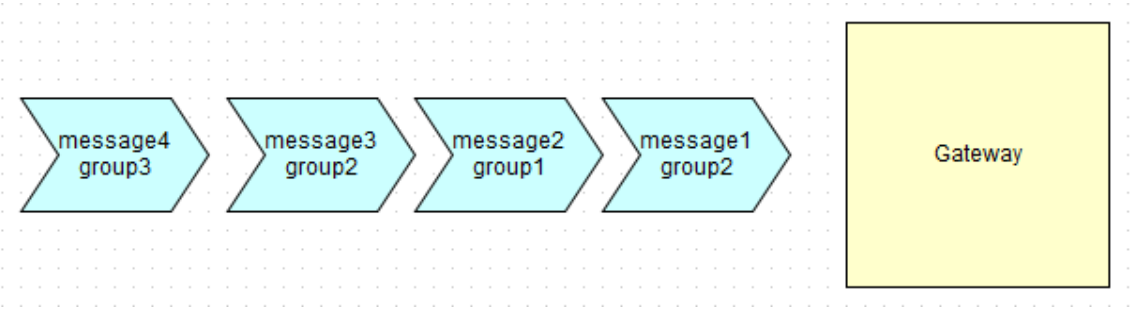
Responding

As messages are completed, if there are queued messages, they should be processed

- Same as the queuing above, but after the first message is completed, the second message is sent to the gateway

Prioritising

If there are messages belonging to multiple groups in the queue, as resources become available, we want to prioritise messages from groups already started.



- For a single resource, messages received:
  - message1 (group2)
  - message2 (group1)
  - message3 (group2)
  - message4 (group3)
- message1 (group2) was received first so will be processed first
- as messages complete, the order they are sent to the gateway should be:
  - message1
  - message3 (it's part of group2, which is already "in-progress")
  - message2
  - message4

Extra credit

Please extend your solution to include at least one of the following features:

Cancellation

It should be possible to tell the scheduler that a group of messages has now been cancelled. Once cancelled, no further messages from that group should sent to the Gateway.

Alternative Message Prioritisation

It should be possible to use different Message prioritisation algorithms to select the next Message from the queue. Invent a new strategy and allow the resource scheduler to be run with this or the original algorithm easily.

Termination Messages

When a Termination Message is received, that means that it is the last Message in that group (not all groups have the same number of messages). If further Messages belonging to that group are received, an error should be raised.

Submission

Please submit the following artifacts for review:

- 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)
- a solution that you would consider "production quality"
- all source and project files required to build and run the task (an eclipse project would ideal)
- all the tests written to support the work (we would love to see evidence of Test Driven Development!)
- any other design work you undertook

You should be able to reach a sensible conclusion to the exercise (you do not have to implement every suggestion in the linked article) in between 1-4 hours. Please submit it along with any instructions on building it when you are ready. Please try to minimise use of 3rd party libs to: JUnit, Log4J, perhaps a mocking framework. The aim is to see you test and build the classes, not just call libraries.