Before starting the code let’s first think about the main structure of the application.

It should mainly be based on a class that will manage scheduling . According to the behaviour of the scheduler, the management of the queue of messages should be done via an ArrayList because:

* It Is ordered
* We can add and remove messages during runtime

This class should be implemented using a Singleton pattern because it contains the queue of messages, which should be unique and also methods to work on this queue. I call it RessourceScheduler.

Based on this, I choose as an extra credit from the beginning the alternative prioritization feature.

For this feature, choosing a Strategy pattern for switching from a prioritization to another during runtime seems to be relevant. It will be based on interface that I call IPrioritizationStategy.

Having this clearly on mind, let’s create a blank Eclipse project. with the two required interfaces.

I create also the ResourceScheduler class with a singleton structure.

I already add the ArrayList of Message and instanciate it in the constructor.

The priorization strategy will be based on already sent messages. Let’s create this interface with the method:

* Public Message chooseMessageToSend(List<Message> toSend, List<Message> sent)

that returns the message to send and where arguments *toSend* is the queue of messages and *sent* is the list of already sent objects.

Here I figure out that the list of sent messages is only needed because of the prioritization. So it shoud be better to keep the list of sent messages in the prioritization classes. So I will change the interface IPrioritizationStategy to an abstract class AbstractPrioritizationStategy having

* Private ArrayList<Message> sent as an attribute
* Public Message chooseMessageToSend(List<Message> toSend) as a method

Then I add two more method to the ResourceScheduler class in order to increment or decrement the available resources. The increment method should be called by the completed method of Message because when a message is completed, a resource becomes available. The decrement method should be used by the send method of Gateway for the opposite reason.

So in this application I have the described prioritization strategy (PrioritizationStrategyDefault class) and a cutom one I should defined (PrioritizationStrategyCustom class).

I will choose the following strategy:

* A message of a group that has never been treated has absolute priority in treatment
* If all groups have been treated then choose the oldest message in queue

I define both strategies and finalize the application.

For my unit tests, I need to add a method that returns the sent messages in the AbstractPrioritizationStategy class.

The unit test using JUnit will be divided in three scenarios

* The scenario for one resource available defined in the exercice description
* A scenario for two resources available
* A scenario for one resource available including switching prioritization strategies

I commented the ResourceSchedulerTest class for more details.

I hope my explanations where clear and you liked this way of programming.