**Reaching to a Solution oF MESSAGEING\_SERVER**

Here we were using queue for taking message input and bytebuffer for storing the same message. If size of bytebuffer is less than the size of message then we are extending the size of buffer. And when messages got read then we have to delete the messages from the buffer. And in buffer it save in bytes and for reading we have to convert byte into Array of Char.

**Drawback of using this approach**: In my approach I used 2 bytebuffers and for getting offSet I made my own method which is not properly suitable as we want it to be worked. And also with this approach we are not re-sizing array here because we have worked bytebuffer only.

**Architecture:**

**Messaging Server:**

**Class MessageGreater**

This will have a 2 byte buffer both are for storing messages if small buffer is become uncapable to store the message then big buffer take all the messages from small buffer and save it inside him and small buffer get clear so that again small buffer can have remaining capacity of 4096 bytes.

**Class MessageQueue**

This class have the Queue as attribute and also we use composition in this class by using MessageGreater class object reference as it’s attribute .

This is the main class where messages are coming adding into queue and waiting for there turn to get send and read.

1. We have to make 2 different classes named as MessageQueue and MessageGreater
2. We will use MessageQueue as glue which will connect the MessageGreater and MessageQueue.
3. Class MessageQueue and MessageGreater will be connected together and here has a relationship takes place.

**Let’s peep inside each class one by one**

**Class MessageQueue**

This class have following attributes:-

* MessageGreater M -> Object reference and it used to fullfill Composition
* Queue<String> MessageQ -> For sending messages from Queue to bytebuffer

This class have following methods :-

* **putMessage();** -> This method is used for putting messages in queue
* **sendMessage();** -> If putMessage() worked successfully only then sendMessage() start running and here we were sending messages from queue to bytebuffer.
* **showMessage();** -> It used to retrieve the message

**Class MessageGreater**

This class have following attributes:-

* **int** smallBlockSize -> It is used for small buffers capacity allocation
* **int** mediumBlockSize -> It is used for medium buffer capacity allocation
* ByteBuffer Message -> It is the small Message byteBuffer with capacity of smallBlockSize.
* ByteBuffer MessageLarger -> It is the medium Message byteBuffer with capacity of mediumBlockSize.

This class have following methods :-

* StringToByte(String text)**;** -> This method is used for putting messages in bytebuffer.
* ByteTOString()**;** -> This method is used for getting String value from bytebuffer.
* position(ByteBuffer b)**;** -> It used to retrieve the offSet location in buffer.

(Note-> for running above classes we have to make our own class named Test, Launcher or as per your wish this class have a object reference of Class MessageQueue and we have to use methods of MessageQueue in main method of class created by you.)

**UML Diagram for this approach :**

MessageQueue :

Attributes:-

- MessageGreater M

- Queue<String> MessageQ

(using PriorityQueue)

Methods:-

+ putMessage()

+ sendMessage ()

+ showMessage()

+ queueSize()

MessageGreater

Attributes:-

- smallBlockSize

-mediumBlockSize

-ByteBuffer Message

-ByteBuffer MessageLarge

Methods:-

+ StringtoByte();

+ BytetoString();

+ position();