Distributed Coursework 2 Report

100136054

1 Introduction

In my coursework, I have implemented every requirement, albeit sometimes in an easy way.

2 Features

2.1 Text, Lines and Images

I implemented three types of image items, text, lines and images (a JPG or PNG file). Colour, font size and style can be customised.

2.2 Image Redraw

When the image is being redrawn, the items are shown one by one according to their creation time.

2.3 Connection

Images can be shared with multiple, synchronised whiteboard nodes. When a new node joins the group, every member will be notified. A node can connect or disconnect to others. The disconnected nodes will not receive image updates from other nodes. When the disconnected node reconnects to the network, it will receive the image items it missed and give out image items it has added.

2.4 Synchronisation

Every node has a synchronised clock. The whiteboard application uses it to synchronise with each other.

The two keys for an image item (text, lines and images) the author and the order. If a reader node has received the second item of an author node without getting the first one, the reader node will request the image from the author node. If the reader node cannot retrieve the first item from the author node, the second item will be regarded as invalid and will not be drawn. This makes sure that no graphical item is rendered before all items it may reference (e.g. point to) are shown.

2.5 Image Sharing

The image, instead of every item, is transferred in the network. The first reason for that is because of the data is just several kilobytes and the application is used in the LAN (local area network). It is not needed to control the data size. Furthermore, some items might be lost. Some nodes might suffer connection issues. The original author node might not be available. Thus, retrieving the whole image is much safer and easier.

2.6 Robot

The robot has three functions.

1) Type – types letters one by one.
2) Move – creates series of images with an offset. When they are being transferred to the end node. It looks like a moving banner of words.
3) File – send the image stored in a file, adding one item at a time.

2.7 Error Handling

Malformed data is caught by ClassNotFoundException. Networking transmission loss is caught by the TimeOutException and other net exceptions. Error messages and logs are centralised by the LogObservable class, then are distributed to the GUI, the console, etc.

3 Protocols

3.1 Communication Protocol

The primary communication protocol is UDP multicasting. Whenever a UPD message is received, a separate thread will deal with it. Without the separate thread, the message receiving thread might be stuck and miss messages. Most of the messages are transferred through the multicasting protocol.

3.2 Image Transfer Protocol

The image transfer protocol is TCP. Every node starts a TCP server. When requests come, the TCP channel will be connected and the image will be transferred to the requester.

3.3 Node Identification Protocol

The solution to let everyone know everyone is by broadcasting messages. The one joins the group will send a message containing its IP address. Everyone receives the message will reply by their IP addresses. Thus, everyone knows everyone. When a node leaves, it sends the leave message to everyone so everyone knows it leaves and will remove it from the IP list.

4.0 Design Patterns

I use the singleton design pattern for the SimpleWhiteboard class because there is only one instance at runtime. I use observer design pattern for the log. Log and error messages are centralised to the LogObservable class. The LogObservable class will add two observers. One is the GUI observer. The other is the console observer. Other observers such as a file observer can be added easily if needed.

5.0 Third Party Code

The third party code is in a separate folder. I use two third party classes. One is the font chooser. The other is the synchronised clock.