Term Project Report

0110830 孫汶琳 電機系 105 級

題目

Let's Paint & Chat! –互動式聊天室

目的

希望能善用目前所學，完成一個可以邊聊天邊畫圖的網頁聊天室。

系統功能

1. 多人進行連線聊天
2. 多人進行畫布編輯

IDE

Notepad++

使用語言

Java、Html、CSS

程式說明

這份作品包含五個主要的 class，分別是 ChatServer、DrawServer、ChatClient、panel、Data。以下將對每個 class 做個別的說明與介紹。

◆ **ChatServer**

ChatServer 的主要工作就是在接收到 ChatClient 端的 Message 後，轉送給所有在線上的 Client。重要程式碼與介紹如下:

1. connect()

connect()主要的功能就是建立 ServerSocket 在指定的 port，並做 Listen，監聽 port 等帶新的 Client 連線後，丟到 Process 這個 thread 去做處理。

```
17      /* Server connect create & listen */
18      public void connect(){
19          try{
20              ServerSocket serversocket = new ServerSocket(8888);
21
22              while(true){
23                  Socket inCon = serversocket.accept();
24                  CON.add(inCon);
25                  System.out.println("Create Server Socket Success!");
26                  Process p = new Process(inCon);
27                  p.start();
28              }
29          }
30          catch(Exception e){
31              System.out.println("Create Server Socket "+e);
32          }
33      }
```

2. toAll()

負責傳送訊息給所有的 Client 端。

```
35      /* Send the message to all the clients */
36      protected void toAll(String message){
37          Iterator it = CON.iterator();
38          while(it.hasNext()){
39              try{
40                  Socket s = (Socket)it.next();
41                  PrintStream writer = new PrintStream(s.getOutputStream());
42                  writer.println(message);
43                  writer.flush();
44              }
45              catch(Exception e){
46                  System.out.println("toAll "+e);
47              }
48          }
49      }
```

3. Process

這是一個 inner class，繼承 Thread，負責接收 Client 端傳送之訊息(message)，並將所接收到之 message 送到 toAll()去處理。

```
51      /* Wait for clients send message */
52      class Process extends Thread{
53          Socket inCon;
54
55          Process(Socket in){
56              inCon = in;
57          }
58
59          public void run(){
60              try{
61                  InputStreamReader in = new InputStreamReader(inCon.getInputStream());
62                  //PrintStream out = new PrintStream(inCon.getOutputStream());
63                  BufferedReader inreader = new BufferedReader(in);
64                  while(true){
65                      try{
66                          String message;
67                          while((message=inreader.readLine())!=null){
68                              toAll(message);
69                          }
70                      }
71                      catch(Exception e){
72                          System.out.println("run-while"+e);
73                          break;
74                      }
75                  }
76
77              }
78              catch(Exception e){
79                  System.out.println("run"+e);
80              }
81          }
82      }
```

◆ DrawServer

基本上功能和架構都和 ChatServer 差不多，比較不一樣的事 ChatServer 傳遞的是字串(String)，而 DrawServer 傳送的是我自己寫的 class-Data，以下就針對這個自定義之 class-Data 來做介紹。

1. Data

負責做為筆跡的傳送媒介的 class，其中 x1,y1,x2,y2 分別代表左上和右下的座標，而 R, G, B 則是 Color 的三個參數，由於不能傳送 Color 這個 class(Java 內建的)，因此改為傳送建構 Color 所需要的這三個參數，分別代表了 Red, Green, Blue。

```
4  public class Data implements Serializable{
5
6      /* axis */
7      public int x1,x2,y1,y2;
8
9      /* color */
10     public int R,G,B;
11
12     /* constructor */
13     public Data(int x, int y, int xx, int yy, int r, int g, int b){
14         x1 = x;
15         x2 = xx;
16         y1 = y;
17         y2 = yy;
18         R = r;
19         G = g;
20         B = b;
21     }
22
23 }
```

◆ **ChatClient**

負責建立 Client 部份之連線與 GNU 介面。

1. Init()

JApplet 中類似 Constructer 的函式，前半部主要是一些物件以及排版的設定，建構完 GNU 後，就是利用 Socket 連線到 ChatServer，以下只秀出連線部分的程式碼。

```
73          try{
74              socket = new Socket("140.113.254.77",8888);
75              Process p = new Process(socket);
76              p.start();
77              show.append("connect!\n");
78          }
79          catch(Exception e){
80              show.append("Socket "+e);
81              System.out.println("Socket "+e);
82          }
```

2. mouseClicked()

當滑鼠按下 Send 這個 Button 後會觸發的 Event，負責將使用者名稱和所輸入的訊息合併後，送到 ChatServer 端。

```
85      /* Action */
86      public void mousePressed(MouseEvent event){}
87      public void mouseReleased(MouseEvent event){}
88      public void mouseClicked(MouseEvent event){
89          try{
90              String message = name.getText()+" : "+input.getText();
91              PrintStream writer = new PrintStream(socket.getOutputStream());
92              writer.println(message);
93              writer.flush();
94              input.setText("");
95          }
96          catch(Exception ex){
97              show.append("mouseClicked"+ex);
98              System.out.println("mouseClicked"+ex);
99          }
100     }
101     public void mouseEntered(MouseEvent event){}
102     public void mouseExited(MouseEvent event){}
```

3. Process

這是一個 inner class，繼承 Thread，負責接收 Server 端傳送之訊息(message)，並將所收到之訊息顯示在對話框內。

```
107     /* get message from server */
108     class Process extends Thread{
109         Socket inCon;
110 
111         Process(Socket in){
112             inCon = in;
113         }
114 
115         public void run(){
116             try{
117                 InputStreamReader in = new InputStreamReader(inCon.getInputStream());
118                 //PrintStream out = new PrintStream(inCon.getOutputStream());
119                 BufferedReader inreader = new BufferedReader(in);
120                 while(true){
121                     try{
122                         String message;
123                         while((message=inreader.readLine())!=null){
124                             show.append(message+"\n");
125                             System.out.println(message);
126                         }
127                     }
128                     catch(Exception e){
129                         show.append("run-while"+e);
130                         System.out.println("run-while"+e);
131                         break;
132                     }
133                 }
134             }
135             catch(Exception e){
136                 show.append("run "+e);
137                 System.out.println("run"+e);
138             }
139         }
140     }
```

◆ **panel**

繼承 JPanel。雖然為了程式上管理的方便而當做是 ChatClient 的 component，不過實際上可獨立出來做為一 Applet 使用，因為其連線部分是獨立與 DrawClient 連線，而且 GNU、Listener 和 Thread 的部份也都是和 ChatClient 獨立的。

1. **panel()**

此 JPanel 的 Constructor，主要負責建立 Listener 以及連線，並觸發一個 Timer 來不斷更新畫布。

```
19      /* Constructor */
20      panel(){
21          /* Set panel size */
22          setSize(500,500);
23          addMouseListener(this);
24          addMouseMotionListener(this);
25
26          /* store the drawing info.*/
27          position = new Vector();
28
29          /* socket to DrawServer */
30          try{
31              socket = new Socket("140.113.254.77",8887);
32              writer = new ObjectOutputStream(socket.getOutputStream());
33              Process p = new Process(socket);
34              p.start();
35          }
36          catch(Exception e){
37              System.out.println("Socket "+e);
38          }
39
40          /* Timer for repaint() */
41          sleepTime = 50;
42          time = new Timer();
43             time.schedule( new TimerTask() {
44                                 public void run()
45                                 {
46                                     repaint();
47                                 }
48                             },0,sleepTime);
49      }
```

2. mouseClicked()

當滑鼠點擊左上方隻變色方框就會變更畫筆顏色，若是點擊其他地方，也會觸發此函式，不過不會有影響。

```
61      /* Set Color */
62      public void mouseClicked(MouseEvent event){
63
64
65          if((event.getX()>=10) && (event.getX()<=30) && (event.getY()>=10) && (event.getY()<=30) )
66              h=1;
67          if((event.getX()>=40) && (event.getX()<=60) && (event.getY()>=10) && (event.getY()<=30) )
68              h=2;
69          if((event.getX()>=70) && (event.getX()<=90) && (event.getY()>=10) && (event.getY()<=30) )
70              h=3;
71          if((event.getX()>=100) && (event.getX()<=120) && (event.getY()>=10) && (event.getY()<=30) )
72              h=4;
73
74      }
```

3. mouseDragged()

當滑鼠在做拖曳的動作時(也就是在繪圖時)，所會觸發的函式。觸發後會傳送所抓到的滑鼠位置，並將會圖所需之資訊組裝成一個 Data object 後，送出到 DrawServer 做處理。

```
80      public void mouseDragged( MouseEvent event ){
81
82          Color color = new Color(0,0,0);
83          switch(h){
84              case 1:
85                  color = new Color(255,0,0);
86                  break;
87              case 2:
88                  color = new Color(0,255,0);
89                  break;
90              case 3:
91                  color = new Color(0,0,255);
92                  break;
93              case 4:
94                  color = getBackground();
95                  break;
96          }
97          x2 = event.getX();
98          y2 = event.getY();
99
100         Data data = new Data(x1,y1,x2,y2,color.getRed(),color.getGreen(),color.getBlue());
101
102         /* output */
103         try{
104             synchronized(writer){
105                 writer.writeObject(data);
106                 writer.flush();
107             }
108         }
109         catch(Exception e){}
110
111         /* update */
112         x1 = x2;
113         y1 = y2;
114
115         repaint();
```

4. paintComponent()

繪製 JPanel 的函式，當 call repaint()時，也會呼叫這個函式。負責會至左上角的變色方框和 position 這個 Vector 中所儲存之 Data(儲存 Server 端目前為止傳過來之筆跡資料)。

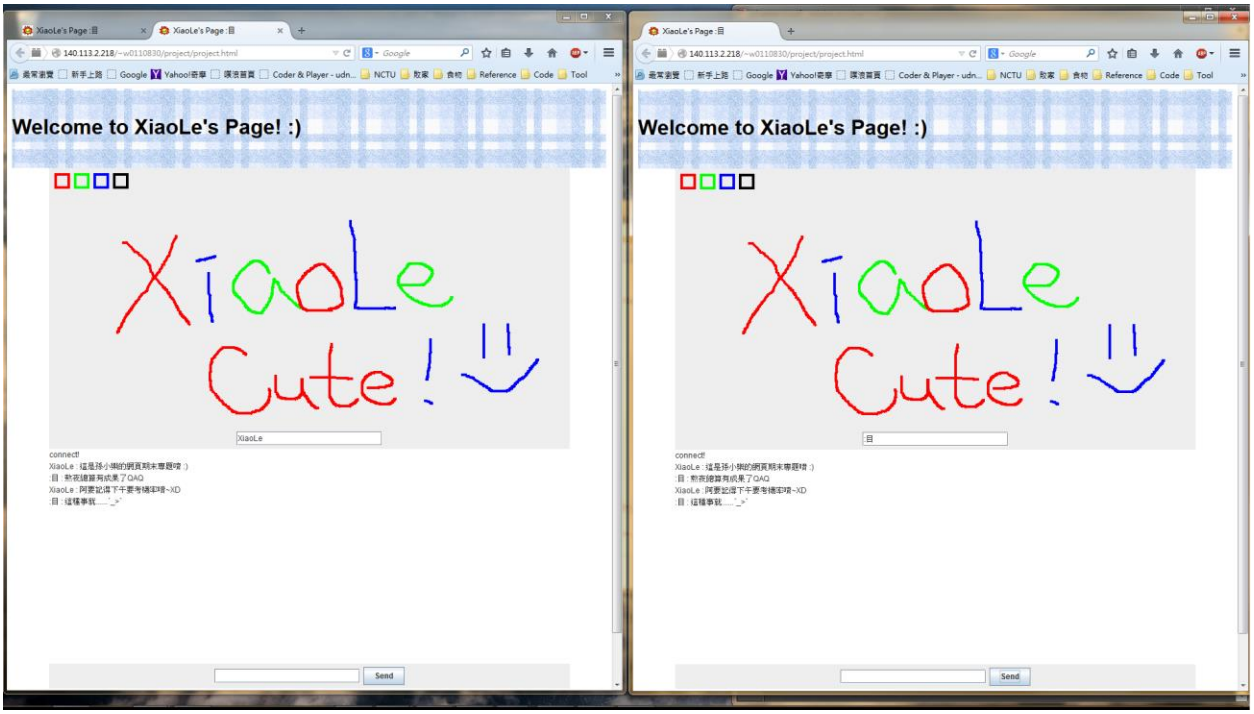
```
119     public void paintComponent(Graphics g){
120
121         super.paintComponent(g);
122         Graphics2D g2 = (Graphics2D) g;
123         g2.setStroke(new BasicStroke(4));
124
125         /* Color Select */
126         g2.setColor(new Color(255,0,0));
127         g2.drawRect(10,10,20,20);
128         g2.setColor(new Color(0,255,0));
129         g2.drawRect(40,10,20,20);
130         g2.setColor(new Color(0,0,255));
131         g2.drawRect(70,10,20,20);
132         g2.setColor(Color.black);
133         g2.drawRect(100,10,20,20);
134
135         /* Draw position */
136         Iterator it = position.iterator();
137         while(it.hasNext()){
138             Data data = (Data)it.next();
139             g2.setColor(new Color(data.R,data.G,data.B));
140             g2.drawLine(data.x1,data.y1,data.x2,data.y2);
141         }
142
143     }
```

5. Process

這是一個 inner class，繼承 Thread，負責接收從 DrawServer 端送來之筆跡資料(Data)後，儲存到 poistion(Vector)中。

```
145     /* get draw information from server */
146     class Process extends Thread{
147         Socket inCon;
148 
149         Process(Socket in){
150             inCon = in;
151         }
152 
153         public void run(){
154             try{
155                 ObjectInputStream in = new ObjectInputStream(inCon.getInputStream());
156                 while(true){
157                     try{
158                         Data data;
159                         while((data=(Data)in.readObject())!=null){
160                             synchronized(position){
161                                 position.add(data);
162                             }
163                         }
164                     }
165                     catch(Exception e){
166                         System.out.println("run-while"+e);
167                         break;
168                     }
169                 }
170             }
171             catch(Exception e){
172                 System.out.println("run"+e);
173             }
174         }
175     }
```

作品截圖



上圖是同時開啟兩個瀏覽器頁面，做為兩個 Client 端的測試結果。

心得

這學期剛好有修 Java 的課程，雖然才剛接觸這個語言，不過我還是很希望藉由這個有相當豐富的 API 的語言，來挑戰看看實現這個作品。這也我第一次接觸 Java 連線的東西，過程中遇到許多的挫折，到現在已經可以獨立完成這個程式了，讓我覺得很有成就感!感謝老師給我了網路的概念，如果沒有老師教導，我在看連線相關東西的時候，一定會吃力許多!

這個作品特別的地方就是它幾乎可以做到同步繪圖的效果，這是我實驗和研究很久之後才是出來的方法，網路上很少這部分的教學或是範例 code。不過基本上，這整份作品都是我一個字一個字打出來的，因此我覺得它的一切都是我值得我驕傲的地方!

很高興這學期能夠完整的修完這堂課，老師教了我們很多概念和新東西，雖然學期中沒有太多的心力去吸收它們，不過它們真的都很實用!希望之後能夠更伸入的學習!