네트워크프로젝트

파싱을 통한 만화보기 프로그램 만들기

20125345 조준형

-같이한 팀원-

20125303 금경원

Intro

개요

GUI 구성

적용된 기술

소스 구성

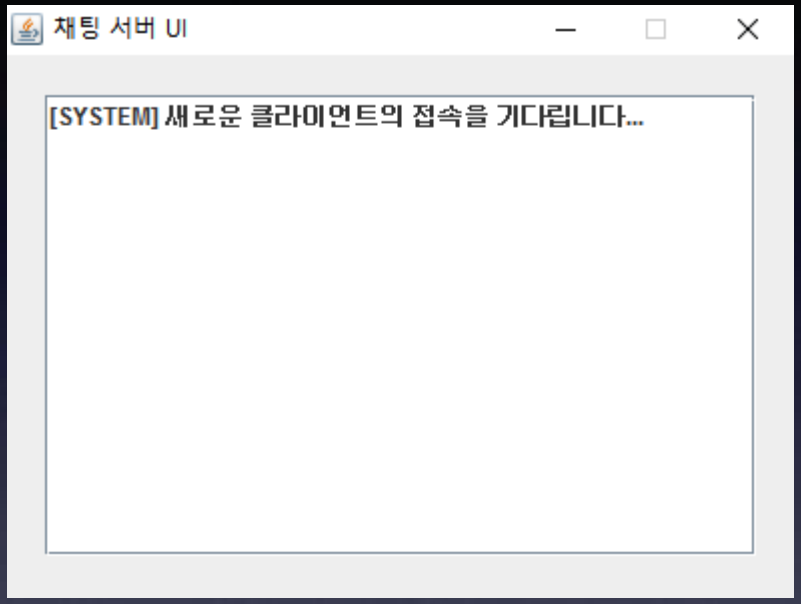
테스트 영상

기대 효과

개요

광고 없이 편안하게 만화를 볼 수 있으며, 다른 사람 독자와 소통을 하면서 편하게 의견을 나눌 수 있는 프로그램을 만들었습니다.

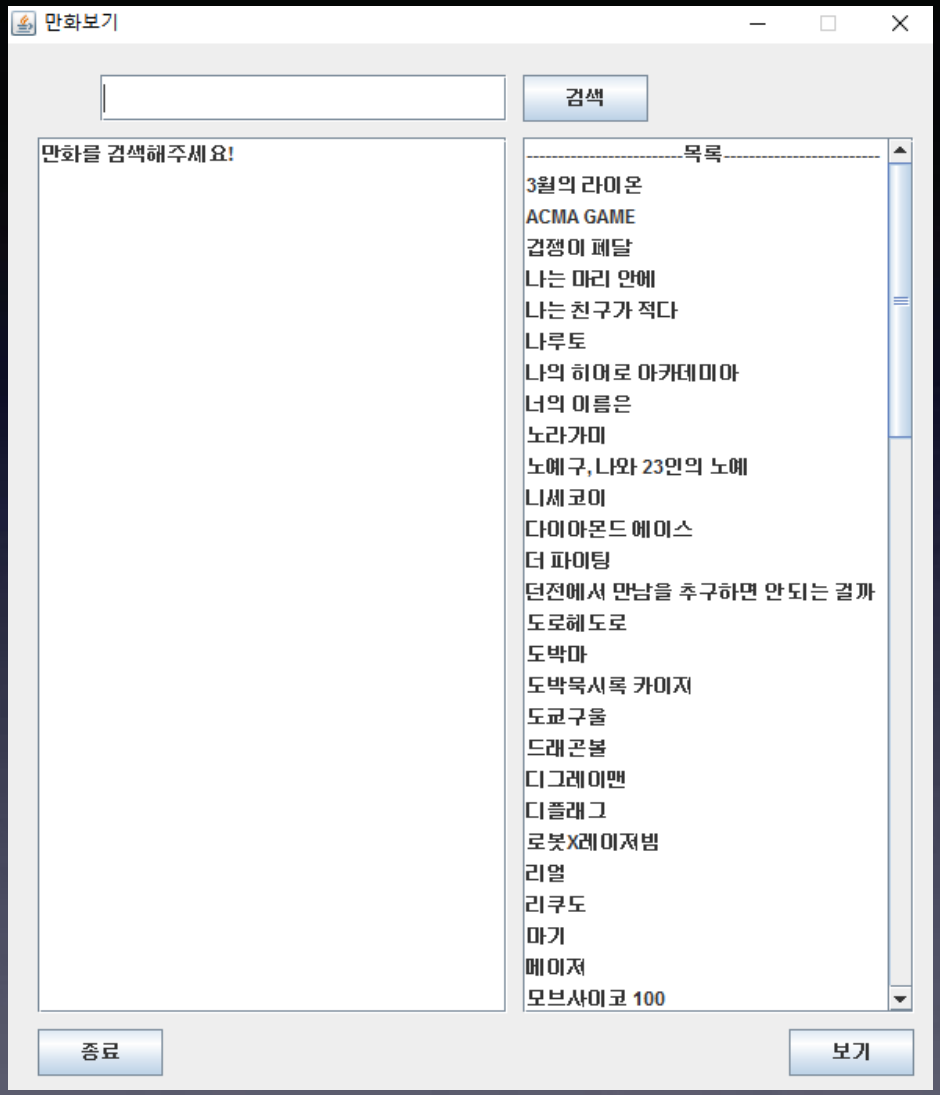
GUI 구성



서버UI

서버 UI를 통해서 사용자의 접속 상태 여부, 채팅이 정상적으로 작동이 되는지를 판단하고 확인 할 수 있게 구성을 했습니다.

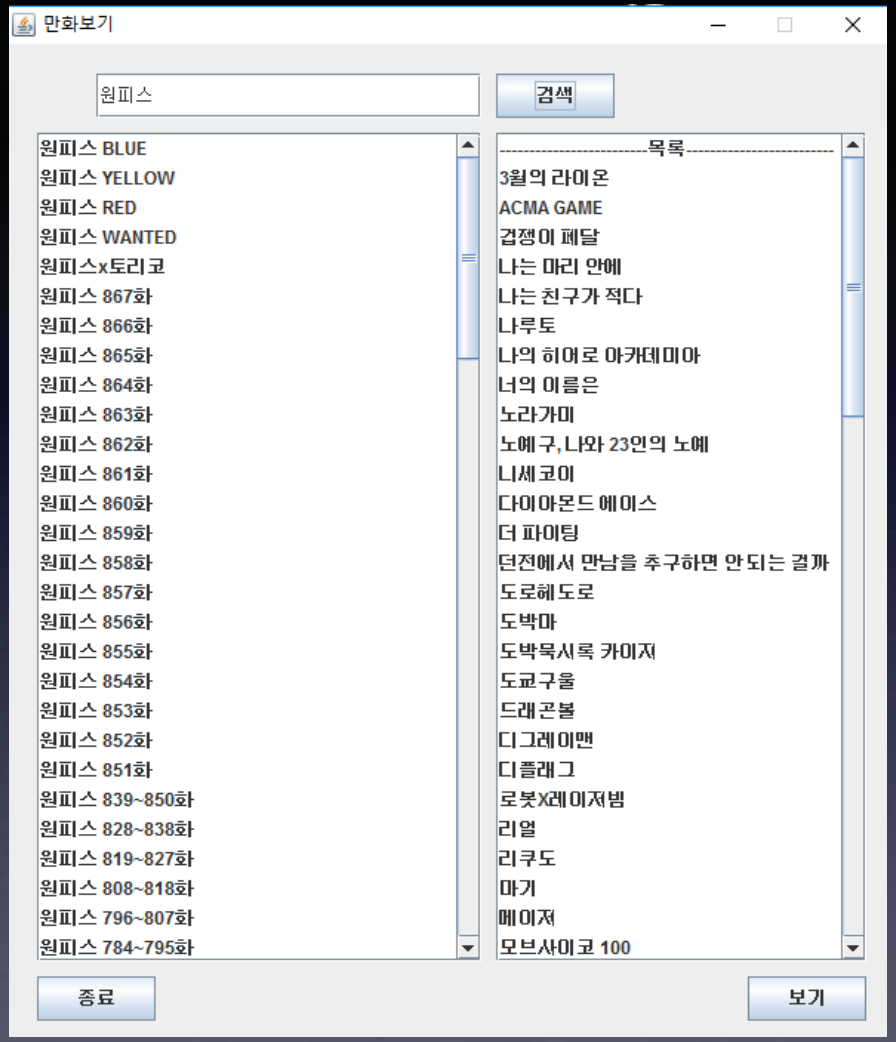
GUI 구성



메인 UI

메인UI는 파싱을 통해서 검색 할 수 목록을 불러올 수 있습니다. 그리고 상단에 있는 텍스트 필드에 보고 싶은 만화 제목을 치게 되면 해당 만화의 목록을 볼 수 있게 구성했습니다.

GUI 구성



메인 UI 검색 후 모습

검색을 하게 되면 해당 만화 회차목록이 나오게
되고 보고 싶은 편을 클릭 후 보기 버튼을 클릭하게 되면 해당 만화를 볼 수 있는 화면으로 넘어가게 됩니다.

GUI 구성

메인 UI 검색 후 모습

만화 화면으로 들어오게 되면 좌측에 만화를 볼 수 있게 되어있고 다음 페이지로 넘어가기 위해서는 하단에 있는 다음페이지 버튼과 이전 페이지 버튼으로 이동 할 수 있으며 전단계로 버튼은 메인 화면으로 돌아갈 수 있게 해줍니다. 우측에는 프로그램을 사용하는 사용자끼리 서로 채팅을 주고 받을 수 있도록 구현을 했습니다.

소스 구성

```java
public class Start {

    public static void main(String[] args) {
        // TODO Auto-generated method stub

        MainUI ui = new MainUI();

    }

}
```

메인 UI를 실행 할 수 있는 메인 함수가 있다.

서버UI에서는 서버GUI를 구성해주고 AcceepServer클래스함수를 선언하여 서버를 실행시켜줄 수 있게 구성했습니다. 하단에 있는 Receive는 클라이언트에서부터 온 메시지를 띄워주기 위해서 인터페이스를 통해 받을 수 있게 구성했습니다.

```java
public class ServerUI extends JFrame implements ReceiveListener{
    private JTextField text1 = null;
    private JList list = null;
    private JScrollPane list_scroll = null;
    private DefaultListModel model = null;
    private AcceepServer acceepserver = null;
    private Client client = null;
    private Server server = null;
    private Receiver receiver = null;

    public ServerUI()
    {
        super("채팅 서버 UI");

        setLayout(null);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setBounds(10,10, 400, 300);
        setResizable(false);

        list = new JList(new DefaultListModel());
        model = (DefaultListModel) list.getModel();

        list_scroll = new JScrollPane(list);
        list_scroll.setBounds(20, 20, 355, 230);

        add(list_scroll);

        setVisible(true);

        acceepserver = new AcceepServer();
        acceepserver.addReceiveListener(this);
        acceepserver.startServer();
    }

    @Override //리스트한테 메세지를 띄워줌.
    public void Receive(String msg) {
        model.addElement(msg);
    }

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        ServerUI ui1 = new ServerUI();


    }
```

소스 구성

```java
public class AcceepServer extends Thread {

    private ReceiveListener listener;
    private ServerSocket serverSocket = null;
    private static final int SERVER_PORT = 6637;
    private boolean serverFlag = true;
    private String users = "broad";

    private String user;


    public void closeServer()
    {
        serverFlag = false;

        try
        {
            serverSocket.close();
        }catch(IOException e)
        {
            e.printStackTrace();
        }

        System.out.println("[SYSTEM] 서버가 종료 되었습니다.");
        System.exit(0);
    }

    public boolean startServer()
    {
        try{
            serverSocket = new ServerSocket(SERVER_PORT);
        }catch(IOException e)
        {
            return false;
        }
        super.start();
        return true;
    }

    public void run()
    {
        while(serverFlag)
        {
            try {
                System.out.println("[SYSTEM] 새로운 클라이언트의 접속을 기다립니다...");
                listener.Receive("[SYSTEM] 새로운 클라이언트의 접속을 기다립니다...");
                Socket socket = serverSocket.accept();

                System.out.println(socket.getInetAddress().getHostAddress());
                int ran = -1;
                do
                {
                    ran = (int)(Math.random()*ServerList.SERVER_MAX);
                }while(!ServerList.serverList[ran].isIdle());

                System.out.println("[SYSTEM] 클라이언트(" + socket.getInetAddress().getHostAddress() +")가 접속하였습니다.");
                listener.Receive("[SYSTEM] 클라이언트(" + socket.getInetAddress().getHostAddress() +")가 접속하였습니다.");


                ServerList.serverList[ran].addReceiveListener(listener);
                ServerList.serverList[ran].setServer(ran,socket);
                ServerList.serverList[ran].startServer();


//                if(ServerList.serverList[ran].getclient_id() == -1)
//                    users += "";
//                else
//                    users += "/" + ServerList.serverList[ran].getclient_id();

                ServerList.serverList[ran].sendMessage(users);



            } catch (IOException e) {
                // TODO Auto-generated catch block
                break;
            }
        }
    }

    public void start()
    {
        throw new RuntimeException("Don't call the start()");
    }
    public void addReceiveListener(ReceiveListener listener) {
        // TODO Auto-generated method stub
        this.listener = listener;
    }


}
```

AcceepServer 클래스에서는 서버를 열어주는 역할을 하는 클래스이다. 스레드를 통해서 소켓통신을 할 수 있게 열어주고 addReceiveListener은 클라이언트에서 만들어진 객체를 받아와 사용하기 위해서 썼습니다.

소스 구성

```
public class ServerList {

    public static final int SERVER_MAX = 50;
    public static Server[] serverList = new Server[SERVER_MAX];

    static
    {
        for(int i=0; i<serverList.length;i++)
            serverList[i] = new Server();
    }
}
```

서버에 접속한 사람들의 아이디 값을 부여해주고 50명까지 접속을 제한하게 하기 위해서 만들어 줬습니다.

```
public interface ReceiveListener {

    public void Receive(String msg);

}
```

클라이언트에서 보낸 메시지를 받아서 전달하기 위해서 인터페이스를 이용해서 Receive함수를 선언했습니다.

```
public class Transmitter {

    private PrintWriter out = null;
    public Transmitter(OutputStream os)
    {
        out = new PrintWriter(new OutputStreamWriter(os));
    }

    public void close()
    {
        if(out != null)
            out.close();
    }
    public void sendMessage(String msg)
    {
        if(out != null)
        {
            out.println(msg);
            out.flush();
        }
    }
}
```

일종의 송신기 역할을 하는 클래스로서 PrintWriter를 이용해서 데이터를 문자 출력 스트림에 출력해준다. sendMessage 메소드를 통해서 각 클라이언트에서 온 메시지들을 출력해준다.

소스 구성

```
public class Receiver extends Thread{
    private ReceiveListener listener;
    private boolean threadFlag = true;
    private int client_id = -1;
    private BufferedReader in = null;
    private AcceepServer acceepserver;
    String users;

    public Receiver(InputStream is, int client_id)
    {
        in = new BufferedReader(new InputStreamReader(is));
        this.client_id = client_id;
    }

    public void close()
    {
        try{
            if(in != null)
                in.close();
            in = null;
        }catch(IOException e)
        {

        }
        threadFlag = false;
        client_id = -1;
    }

    public void run()
    {
        acceepserver = new AcceepServer();
        while(threadFlag)
        {
            try
            {

                String msg = in.readLine();
                if(msg == null)
                    break;
                else
                {
                    System.out.println("[SYSTEM] " + client_id + "님으로 부터 온 메시지 : " + msg);
                    listener.Receive("[SYSTEM] " + client_id + "님으로 부터 온 메시지 : " + msg);

                    for(int i=0;i<ServerList.serverList.length;i++)
                        ServerList.serverList[i].sendMessage(client_id + "님의 말 : " + msg);
                }
            }catch(IOException e)
            {
                break;
            }

        }
        System.out.println("[SYSTEM] " + client_id + "님이 종료하였습니다.");
        listener.Receive("[SYSTEM] " + client_id + "님이 종료하였습니다.");

        ServerList.serverList[client_id].closeServer();
        close();
    }

    public void addReceiveListener(ReceiveListener listener)
    {
        this.listener = listener;
    }


}
```

Receiver 클래스는 여러 클라이언트들의 메시지들을 중계하고 각 클라이언트에게 전송한다. 또 클라이언트들이 접속을 종료했을 때 다른 클라이언트들에게 종료했다라는 문구를 보낼 수 있다.

소스 구성

```java
public class Client extends Thread
{
    private static final int ACCEPT_PORT = 6637;
    private static final String ACCEPT_IP = "127.0.0.1"; //203.253.255.99

    private BufferedReader receive = null;
    private PrintWriter trans = null;
    private BufferedReader in = null;
    private ReceiveListener listener;
    private ReceiveListener UserList;
    private AcceepServer acceepserver;
    private String msg;


    public void run()
    {
        Socket socket = null;


        try
        {
            System.out.println("[IP:" + ACCEPT_IP + ",PORT:" + ACCEPT_PORT + "]로 접속중..." );
            listener.Receive("[IP:" + ACCEPT_IP + ",PORT:" + ACCEPT_PORT + "]로 접속중...");
            socket = new Socket(ACCEPT_IP, ACCEPT_PORT);
            System.out.println("서버에 접속하였습니다.");
            listener.Receive("서버에 접속하였습니다.");
            receive = new BufferedReader(new InputStreamReader(socket.getInputStream()));
            trans = new PrintWriter(new OutputStreamWriter(socket.getOutputStream()));

        }catch(IOException e){
            // 서버 접속 실패 - ip나 port가 잘못되었을 경우
            System.err.println("[ERR]서버 접속 실패");
            return;
        }
         //서버로 보낼 메시지를 입력받는 동시에 서버에서 오는 메시지를 받음
        try
        {
            while(true)
            {
//              // 서버로 부터 메시지를 받는 것을 기다린다.
                msg = receive.readLine();
//              // 받은 메시지를 출력한다.
                System.out.println(msg);
                listener.Receive(msg);
            }
        }catch(IOException e){
```

```java
        } catch (IOException e) {

        }
    }
    public void sendMessage(String msg)
    {
                trans.println(msg);
                trans.flush();
                this.msg = msg;
    }

    public void addReceiveListener(ReceiveListener listener) {
        this.listener = listener;
    }


}
```

클라이언트 클래스는 클라이언트들이 서버에 접속할 수 있게 구현을 했고, 메시지를 적으면 Receive를 통해서 서버를 통해서 다른 클라이언트들에게 전송됩니다. 이 클래스는 cartoon클래스에서 생성됩니다.

소스 구성

```java
public class MainUI extends JFrame {

    private JTextField text1 = null;
    private JList Cartoon_list = null, index_list = null;
    private JButton button_search = null, button_next = null, button_exit = null;
    private JScrollPane Cartoon_list_scroll  = null, index_list_scroll  = null;
    private DefaultListModel Cartoon_list_model = null, index_list_model = null;
    private Cartoon cartoon = null;
    private Cartoon2 cartoon2 = null;
    private Cartoon3 cartoon3 = null;
    private String name; // 검색키워드
    private Parser p;
    private ArrayList<Data> list = new ArrayList<>();

    public MainUI() {
        super("만화보기");

        setLayout(null);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        //setBounds(290,160, 345, 700);
        setBounds(450,200, 600, 700);
        setResizable(false);

        Cartoon_list = new JList(new DefaultListModel());
        Cartoon_list_model = (DefaultListModel) Cartoon_list.getModel();

        Cartoon_list_scroll = new JScrollPane(Cartoon_list);
        Cartoon_list_scroll.setBounds(20, 60, 300, 560);
        Cartoon_list_model.addElement("만화를 검색해주세요!");
        MouseListener mouseListener = new MouseAdapter() {  //리스트 클릭했을 때
             public void mouseClicked(MouseEvent mouseEvent) {
                if (mouseEvent.getClickCount() == 2) {
                   int index = Cartoon_list.locationToIndex(mouseEvent.getPoint());
                   if (index >= 0) {
                      Object o = Cartoon_list_model.getElementAt(index);

                }

                   }
                }
            };
```

```java
        index_list = new JList(new DefaultListModel());
        index_list_model = (DefaultListModel) index_list.getModel();

        index_list_scroll = new JScrollPane(index_list);
        index_list_scroll.setBounds(330, 60, 250, 560);
        index_list_model.addElement("----------------------목록----------------------");

        list = p.list();
        for(int i = 0; i < 82 ; i++) {
            index_list_model.addElement(list.get(i).toString());
        }



        text1 = new JTextField();
        text1.setBounds(60, 20, 260, 30);


        button_search = new JButton("검색");
        button_search.setBounds(330,20,80,30);
        button_search.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {    // 검색 버튼

            ArrayList<Data> da = new ArrayList<>();
            int i = 0;
            int blank = 0;
            name = text1.getText();
            da = p.SearchTitle(name);
            Cartoon_list_model.removeAllElements();
            try {
                if ((da.get(i).toString()) == null) {
                    ++i;
                    blank++;
                    if(blank > 3) {
                        while((da.get(i).toString()) != null) {
                            Cartoon_list_model.addElement(da.get(i).toString());
                            i++;
                        }
                    }
                    else {
                        while((da.get(i).toString()) != null) {
                            Cartoon_list_model.addElement(da.get(i).toString());
```

MainUI클래스는 프로그램을 실행했을 때 나오는 첫 화면이다. 2개의 Jlist를 선언하여 하나는 만화목록을, 하나는 만화를 검색 후 나오는 회차목록을 보여줍니다. 그리고 검색버튼을 눌렀을 때 JTextField에서 문자값을 받아와서 Parser클래스의 SearchTitle메소드로 인해서 검색한 만화의 회차목록을 찾아서 Jlist안에 출력해줍니다.

소스 구성

```
                    i++;
                }
            }
        }catch (IndexOutOfBoundsException e3) {
            // TODO: handle exception

        }
      }
    });

    button_next = new JButton("보기");
    button_next.setBounds(500,630,80,30);
    button_next.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {     // 보기 버튼
       int index = Cartoon_list.getSelectedIndex();

       if( index != -1 && Cartoon_list_model.size() != 0 ) {
          Object o = Cartoon_list_model.getElementAt(index);
          String title = (String)o;
          if(name.equals("아카메가 벤다") ||name.equals("ACMA GAME") || name.equals("도박마") || name.equals("오늘부터 신령님")
                  || name.equals("장난을 잘치는 타카기 양") || name.equals("테라포마스") || name.equals("피아노의 숲"))
          {
              String link = p.SearchLink(name,title);
              cartoon3 = new Cartoon3(title,link);
          }
          else if(name.equals("사남") ||name.equals("블랙 클로버") ||name.equals("리쿠도") ||name.equals("리얼") ||name.equals("로봇x레이저빔")
                  ||name.equals("아오오니") ||name.equals("원피스") ||name.equals("헌터x헌터") || name.equals("하이큐") || name.equals("플래티넘 엔드")
                  || name.equals("파이어 펀치") || name.equals("토리코") || name.equals("테니스의 왕자") || name.equals("킹덤") || name.equals("악속의 네버랜드")
                  || name.equals("원펀맨 히어로대전"))
          {
              String link = p.SearchLink(name,title);
              cartoon2 = new Cartoon2(title,link);
          }
          else {
              String link = p.SearchLink(name,title);
              cartoon = new Cartoon(title, link);
          }

       }
          setVisible(false);
       }
    });
    add(index_list_scroll);
    add(Cartoon_list_scroll);
    add(text1);
    add(button_search);
    add(button_next);
    add(button_exit);
    setVisible(true);


}

}
```

보기 버튼을 눌렀을 때는 Jlist에 있는 회차목록의 url과 이름을 Parser클래스에 있는 SearchLink메소드를 통해서 가지고와서 Cartoon클래스로 넘겨줍니다. 여기서 만화마다 태그값이 다르기때문에 여러 클래스를 만들어서 if문으로 문자열 비교를 한 후 다음 UI를 띄워주게 됩니다.

소스 구성

```java
public class Cartoon extends JFrame implements ActionListener, ReceiveListener{

    private JTextField text1 = null;
    private JList chatlist= null;
    private JButton button_back = null, button_next = null, button_before = null, button_send = null ;
    private JScrollPane chatlist_scroll = null;
    private DefaultListModel model = null;
    private ImageIcon image = null;
    private JPanel panel = null;
    private JLabel cartonView = null;
    private ScrollPane scroll = null;

    private Client client = null;

    private int page = 0;
    private String Slink = null;
    private byte[] carton = null;

    public Cartoon(String title, String link) { //매개변수로 클릭한 제목을 넘길 생각.
        super(title); // 상단에 제목을 표시하기 위해서 매개변수를 넣음

        InputStream in = URLManager.getURLInputStream(link, URLManager.USER_AGENT_PC);

        try{
            Document doc = Jsoup.parse(in, URLManager.ENCODING_UTF8,"");
            Elements root = doc.select("div[class=contents]");
            Elements rankList = root.select("p");
            Elements rankList2 = rankList.select("a");

            Slink = rankList2.get(page).attr("href");
            carton = URLManager.getImage(Slink ,link);

            setLayout(null);
            setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            setBounds(290,160, 900, 700);
            setResizable(false);
```

```java
System.out.println("Cartoon_link : " + link);
panel = new JPanel();
scroll = new ScrollPane();
panel.setBounds(10,10,573, 625);
panel.setLayout(null);


button_back = new JButton("전 단계로");
button_back.setBounds(50,630,110,30);
button_back.addActionListener(new ActionListener() {
// @Override
public void actionPerformed(ActionEvent e) {

    MainUI ui = new MainUI();
    setVisible(false);
}
});

button_next = new JButton("다음페이지");
button_next.setBounds(450,630,110,30);
button_next.addActionListener(new ActionListener() {
// @Override
public void actionPerformed(ActionEvent e) {
    page = page + 1;

    if(root.size() < page) {
        JOptionPane.showMessageDialog(null, "마지막 페이지 입니다.");
        page = page - 1;
    }

    Slink = rankList2.get(page).attr("href");
    carton = URLManager.getImage(
            Slink ,link);


    // System.out.println(carton);
     cartonView.setIcon(new ImageIcon(carton));
     cartonView.revalidate();
     cartonView.repaint();
     cartonView.update(cartonView.getGraphics());
}

});
```

Cartoon클래스에서는 MainUI클래스에서 매개변수를 통해서 해당만화의 url을 받아와서 imageicon에 이미지를 파싱하게 됩니다. 이때 URLManager의 getImage메소드를 통해서 값을 받아오게 됩니다. 전 단계로 버튼을 클릭했을 때는 전화면인 MainUI클래스를 불러오게 됩니다. 다음페이지로 버튼은 현재 나와있는 이미지의 다음 이미지를 불러옵니다. 이때 setIcon메소드로 새로운 이미지를 받고, revalidate메소드와 repaint메소드로 이미지를 새로고침해주고, update메소드를 통해서 다시 호출해줍니다.

소스 구성

```
button_before = new JButton("이전페이지");
button_before.setBounds(250,630,110,30);
button_before.addActionListener(new ActionListener() {
// @Override
public void actionPerformed(ActionEvent e) {
    page = page-1;

    Slink = rankList2.get(page).attr("href");
    carton = URLManager.getImage(
            Slink ,link);

      //System.out.println(carton);
      cartonView.setIcon(new ImageIcon(carton));
      cartonView.revalidate();
      cartonView.repaint();
      cartonView.update(cartonView.getGraphics());
    }
});

image = new ImageIcon(carton);
cartonView = new JLabel(image);

chatlist = new JList(new DefaultListModel());
model = (DefaultListModel) chatlist.getModel();
chatlist_scroll = new JScrollPane(chatlist);
chatlist_scroll.setBounds(583, 20, 300, 602);

text1 = new JTextField();
text1.setBounds(583, 630, 215, 30);

button_send = new JButton("전송");
button_send.setBounds(803,630,80,30);
button_send.addActionListener(this);

scroll.setSize(570,625);
scroll.add(cartonView);
add(scroll);
add(button_back);
add(button_next);
add(button_before);
add(chatlist_scroll);
add(text1);
add(button_send);
setVisible(true);
```

```
            client = new Client();
            client.addReceiveListener(this);
            client.start();

        }catch(java.lang.IndexOutOfBoundsException e1){
            // TODO: handle exception
            JOptionPane.showMessageDialog(null, "존재하지 않는 페이지입니다.");

        }catch (Exception e) {
            // TODO: handle exception
        }
    }

    public void actionPerformed(ActionEvent e) {
        // TODO Auto-generated method stub
        Object o = e.getSource();
        //boolean clear = false;
        if( o == button_send)   // 통신
        {
            String str = text1.getText();
            client.sendMessage(str);
            text1.setText("");
        }
    }

    @Override
    public void Receive(String msg) {   //소켓통신 부분
        // TODO Auto-generated method stub
        boolean a = msg.matches(".*broad/.*");
        if(a){
            for(int i = 1; i<msg.split("/").length;i++)
            {

                System.out.println(msg.split("/")[i]);
            }
        }
        else{
                model.addElement(msg);
                System.out.println(msg);
        }
    }


}
```

클라이언트를 실행시켜서 서버와 연결 하게 해줍니다 여기서 만들어준 객체를 서버클래스들에게 보내주기위해서 addReceiveListner함수를 사용했습니다. Receive함수를 통해서 메시지를 주고 받을 수 있게 구현했습니다.

소스 구성

```
            Document doc = Jsoup.parse(in, URLManager.ENCODING_UTF8,"");
            Elements root = doc.select("div[class=contents]");
            Elements rankList = doc.select("div[class=separator]");
            Elements rankList2 = rankList.select("a");


            Slink = rankList2.get(page).attr("href");
            carton = URLManager.getImage(
                    Slink ,link);
```

```
InputStream in = URLManager.getURLInputStream(link, URLManager.USER_AGENT_PC);

try{
    Document doc = Jsoup.parse(in, URLManager.ENCODING_UTF8,"");
    Elements root = doc.select("div[class=contents]");
    Elements rankList = root.select("p");
    Elements rankList2 = rankList.select("img");
    Slink = rankList2.get(page).attr("src");
    carton = URLManager.getImage(Slink ,link);
```

사이트에서 만화마다 태그값이 다르기 때문에 클래스를 각자 만들었습니다.

```
public class Data {
    String keyword, link;

    public Data(String keyword, String link)
    {
        this.keyword = keyword;
        this.link = link;
    }

    public Data(String keyword)
    {
        this.keyword = keyword;
    }

    public String getkeyword()
    {
        return keyword;
    }
    public void setkeyword(String keyword)
    {
        this.keyword = keyword;
    }

    public String getLink()
    {
        return link;
    }
    public void setLink(String link)
    {
        this.link = link;
    }

    public boolean equals(Object obj)
    {
        Data d = (Data)obj;
        return d.getkeyword().equals(keyword) &&
                d.getLink().equals(link);
    }

    @Override
    public String toString() {
        // TODO Auto-generated method stub
        return String.format("%s",keyword);
    }

}
```

파싱을 할 때 일 일정한 포맷으로 값을 ArrayList에 저장하기 위해서 만들었습니다.

소스 구성

```java
public class Parser {
    public static ArrayList<Data>list()
    {
        ArrayList<Data> da = new ArrayList<>();
        String url = "http://zangsisi.net/";
        String Slink = null;
        String title = null;
        boolean endflag = false;
        Data d;
        InputStream in = URLManager.getURLInputStream(url, URLManager.USER_AGENT_PC);

        try{
            Document doc = Jsoup.parse(in, URLManager.ENCODING_UTF8,"");
            Elements root = doc.select("div[id=manga-list]");
            Elements rankList = root.select("a");
            for(int j = 0; j<rankList.size();++j){
                title = rankList.get(j).text();
                d = new Data(title);
                da.add(d);
            }
        }catch (IOException e) {
            // TODO: handle exception
            e.printStackTrace();
        }
        return da;
    }
```

해당 사이트에 어떤 만화를 볼 수 있는지를 보여주기 위해서 목록만 파싱 후 ArrayList에 넣어서 리턴합니다.

```java
public static ArrayList<Data>SearchTitle(String keyword)
{
    ArrayList<Data> da = new ArrayList<>();
    String url = "http://zangsisi.net/";
    String Slink = null;
    String title = null, title2 = null;
    boolean endflag = false;
    Data d;
    InputStream in = URLManager.getURLInputStream(url, URLManager.USER_AGENT_PC);

    try{
        Document doc = Jsoup.parse(in, URLManager.ENCODING_UTF8,"");
        Elements root = doc.select("div[id=manga-list]");
        Elements rankList = root.select("a");
        for(int j = 0; j<rankList.size();++j){
            Slink = rankList.get(j).attr("href");
            title = rankList.get(j).text();

            if(title.equals(keyword)){
                InputStream in2 = URLManager.getURLInputStream(Slink, URLManager.USER_AGENT_PC);
                Document doc2 = Jsoup.parse(in2, URLManager.ENCODING_UTF8,"");
                Elements root2 = doc2.select("div[id=post]");
                Elements subroot2 = root2.select("div[class=contents]");
                Elements rankList2 = subroot2.select("p");

                for(int i = 0; i < rankList2.size(); i++) {
                    title2 = rankList2.get(i).text();

                    d = new Data(title2);
                    da.add(d);
                    endflag = true;
                }
            }
            if(endflag)
                break;
        }
    }catch (IOException e) {
        // TODO: handle exception
        e.printStackTrace();
    }
    return da;
}
```

해당 사이트에 내가 검색한 만화가 있으면 해당 만화 목록을 파싱 후 ArrayList에 입력 후 리턴합니다.

소스 구성

```java
public static String SearchLink(String keyword, String keyword2)
{
    ArrayList<Data> da = new ArrayList<>();
    String url = "http://zangsisi.net/";
    String Slink = null, Slink2 = null;
    String title = null, title2 = null, result = null;
    boolean endflag = false;
    Data d;
    InputStream in = URLManager.getURLInputStream(url, URLManager.USER_AGENT_PC);

    try{
        Document doc = Jsoup.parse(in, URLManager.ENCODING_UTF8,"");
        Elements root = doc.select("div[id=manga-list]");
        Elements rankList = root.select("a");

        for(int j = 0; j<rankList.size();++j){
            Slink = rankList.get(j).attr("href");
            title = rankList.get(j).text();

            if(title.equals(keyword)){
                InputStream in2 = URLManager.getURLInputStream(Slink, URLManager.USER_AGENT_PC);
                Document doc2 = Jsoup.parse(in2, URLManager.ENCODING_UTF8,"");
                Elements root2 = doc2.select("div[id=post]");
                Elements subroot2 = root2.select("div[class=contents]");
                Elements rankList2 = subroot2.select("p");
                Elements rankList_link = subroot2.select("a");
                for(int i = 0; i < rankList2.size(); i++) {
                    title2 = rankList2.get(i).text();
                    if(title2.equals(keyword2)){
                        Slink2 = rankList_link.get(i).attr("href");
                        endflag = true;
                    }
                }
            }
            if(endflag)
                break;
        }
    }catch (IOException e) {
        // TODO: handle exception
        e.printStackTrace();
    }

    return Slink2;
}
```

만화을 검색하게 되면 만화목록들이 나오게 되는데 이때 해당하는 만화목록의 url과 이름을 리턴하기 위해서 함수를 만들었습니다. Url을 통해서 다음 화면의 이미지 아이콘에 만화가 나오게 구성했습니다.

적용된 기술

1. 소켓통신을 이용한 다중 채팅 구현 - 조준형

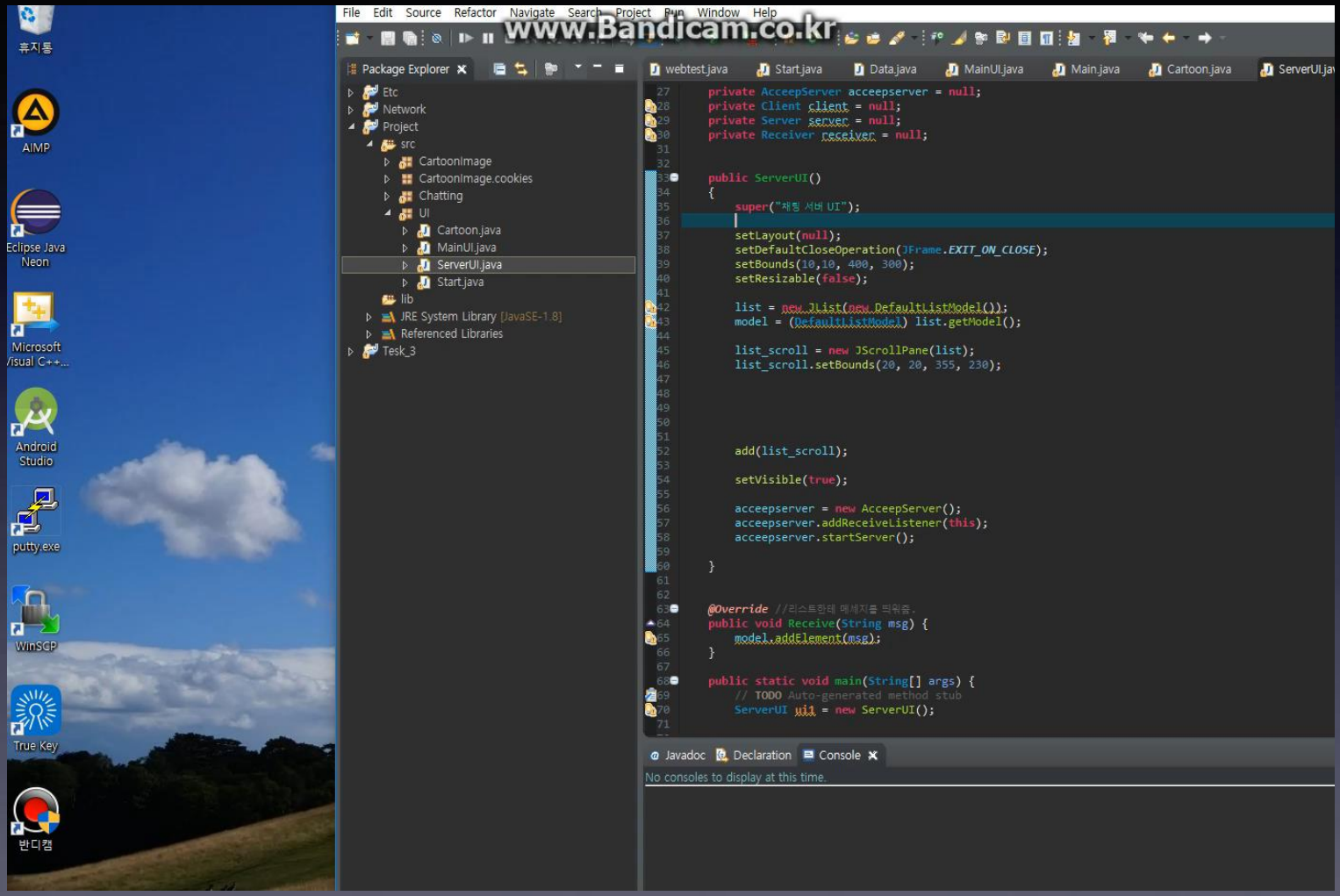
2. 파싱을 이용하여 사이트에 있는 이미지 형태의 만화를 가져옴 – 금경원, 조준형

3. 파싱을 통해서 목록을 불러온다 – 금경원, 조준형

4. 해당 만화를 클릭 후 버튼을 누를 시 해당하는 만화화면으로 이동 - 조준형

5. 자바 GUI를 이용해서 화면을 구성 - 금경원

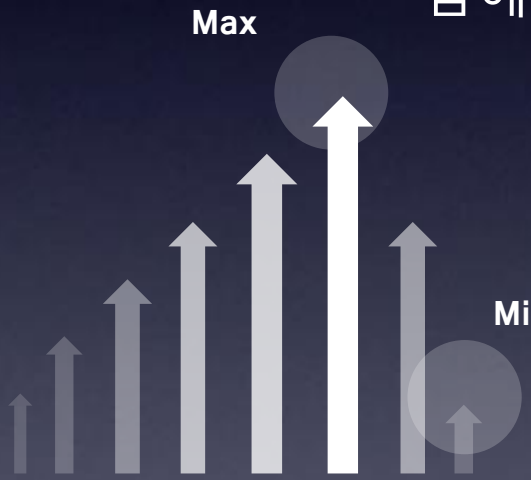
테스트 영상


File Edit Source Refactor Navigate Search Project Run Window Help
www.Bandicam.co.kr
Package Explorer
Etc
Network
Project
src
CartoonImage
CartoonImage.cookies
Chatting
UI
Cartoon.java
MainUI.java
ServerUI.java
Start.java
lib
JRE System Library [JavaSE-1.8]
Referenced Libraries
Tesk_3
webtest.java
Start.java
Data.java
MainUI.java
Main.java
Cartoon.java
ServerUI.ja
private AcceepServer acceepserver = null;
private Client client = null;
private Server server = null;
private Receiver receiver = null;
public ServerUI()
{
super("채팅 서버 UI");
setLayout(null);
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
setBounds(10,10, 400, 300);
setResizable(false);
list = new JList(new DefaultListModel());
model = (DefaultListModel) list.getModel();
list_scroll = new JScrollPane(list);
list_scroll.setBounds(20, 20, 355, 230);
add(list_scroll);
setVisible(true);
acceepserver = new AcceepServer();
acceepserver.addReceiveListener(this);
acceepserver.startServer();
}
@Override //리스트한테 메세지를 띄워줌.
public void Receive(String msg) {
model.addElement(msg);
}
public static void main(String[] args) {
// TODO Auto-generated method stub
ServerUI ui1 = new ServerUI();
Javadoc Declaration Console
No consoles to display at this time.
휴지통
AIMP
Eclipse Java Neon
Microsoft Visual C++...
Android Studio
putty.exe
WinSCP
True Key
반디캠

기대효과

내가 원하는 만화 작품만 골라 볼 수 있는 편의성 상승

다른 독자들과 소켓통신으로 인한 작품에 대한 이해도와 재미 상승



광고로 인한 불편함은 감소

마치며..

이번 프로젝트를 통해서 파싱을 확실히 사용할 수 있게 연습이 된 계기가 되서 좋았습니다. 이번에 프로젝트를 하면서 힘들었던 점은 여러 이미지를 불러와서, UI화면에 띄워주는 것이 가장 힘들었습니다. 하지만 setIcon, revalidate메소드, repaint와 update메소드를 통해서 Jlabel를 업데이트를 해서 보여주는 기법을 배울 수 있었습니다.
이번 프로젝트를 하면서 아쉬웠던 점과 개선할 사항은 첫 번째로 너무 많은 url과 태그값을 거쳐서 그런지 프로그램의 구동이 다소 느린부분을 개선하겠습니다. 두번째는 UI디자인을 좀 더 꾸며서 사용자가 더욱 편리하게 사용 할 수 있도록 개선하겠습니다. 세 번째는 채팅접속시 나오는 아이디가 임의 값으로 나오는데, 사용자가 원하는 아이디와 접속한 사람들의 아이디목록을 보여줄 수 있게 개선하겠습니다. 마지막으로는 만화 종류가 생각보다 많이 없는데 종류를 늘려서 더욱 많은 만화를 볼 수 있게 지원하겠습니다.

Thank you