소프트웨어 포트폴리오

지원자 **이석화**

목차

1. 시저암호 채팅

개요

-명　　칭: 시저암호 채팅

-개발인원: 1명

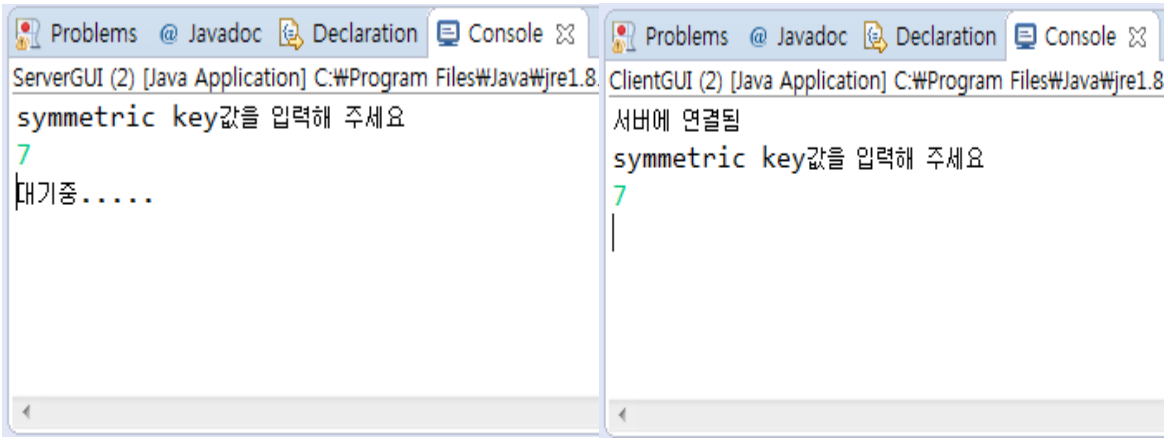
-오픈 소스 사용여부: X

-개발환경: JAVA 8 eclipse 사용

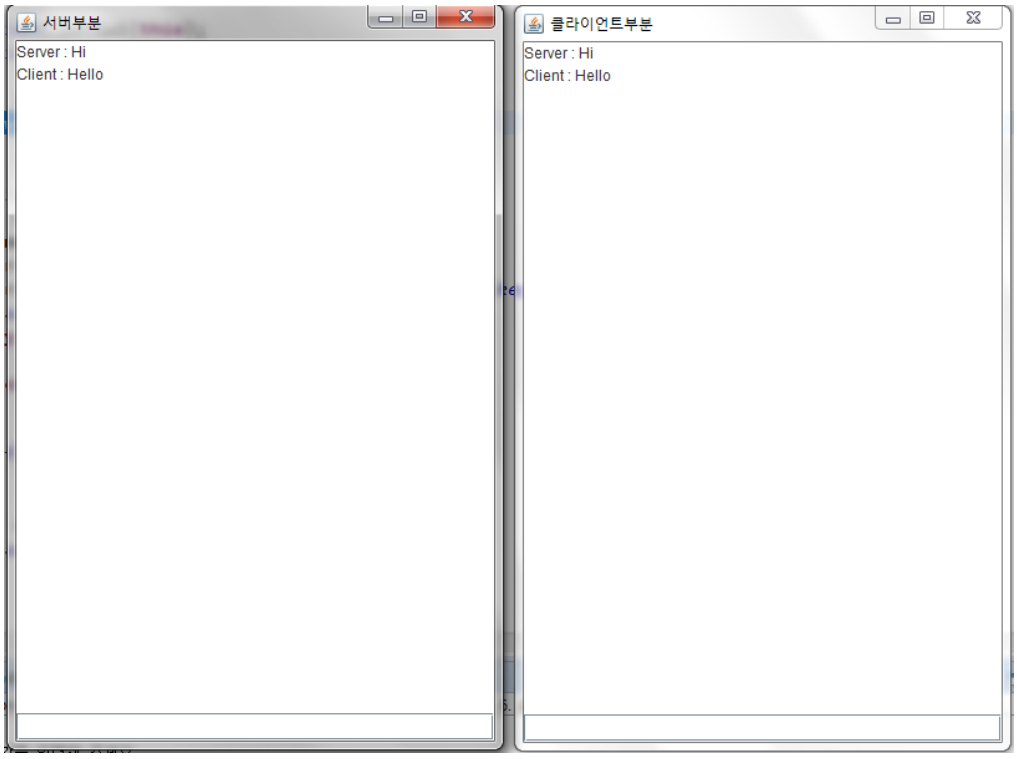
-소스코드: https://github.com/HwaSeok/chatting-program

소개

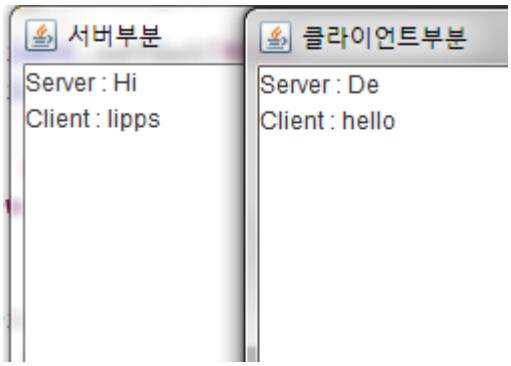
시저암호 채팅은 대화를 하고 있는 상대가 정말로 지정한 사람인지 아니면 그 사람을 사칭하고 있는 다른 사람인지 확인하기 위하여 대칭 키 암호화 방식을 이용하여 개발한 프로그램입니다. 자신과 상대가 같은 암호만 알고 있다면 일반적으로 사용되는 로그인 방식의 프로그램보다도 편리하게 채팅을 사용할 수 있습니다.



먼저 서버에서 프로그램을 실행하고 콘솔 창에 암호를 입력합니다. 그 뒤 클라이언트도 실행한 뒤 콘솔 창에 같은 암호를 입력합니다.



같은 암호를 사용하였으므로 채팅이 정상적으로 이루어졌습니다. 하지만 서로의 암호가 다를 경우 시저 암호방식에 따라 메시지의 내용이 알파벳 별로 일정한 거리만큼 밀어서 다른 알파벳으로 치환되어 아래와 같은 모습이 됩니다.



소스코드 일부

Symmetric_Key.java
서로 암호가 다를 때 메시지를 시저암호에 따라 치환하도록 하는 클래스입니다. String을 Char타입으로 변환하고 키 값 (암호)만큼 알파벳 다음순서로 밀어냅니다. 알파벳 마지막까지 밀어내면 다시 처음부터 시작합니다.

```
public String Encrypt(String text,int key){
	char[] temp = text.toCharArray();
	String result="";

	for(int i=0; i<text.length(); i++){
		if(temp[i] == ' ')
			temp[i] = ' ';
		else{
			for(int j=0; j<key; j++){
				temp[i]++;
				if(temp[i]=='{')
					temp[i]='a';
				if(temp[i]=='[')
					temp[i]='A';
			}
		}
		result = result + temp[i];
	}

	return result;
}

public String Decrypt(String text, int key){
	char[] temp = text.toCharArray();
	String result="";

	for(int i=0; i<text.length(); i++){
		if(temp[i] == ' ')
			temp[i] = ' ';
		else{
			for(int j=0; j<key; j++){
				temp[i]--;
				if(temp[i]=='`')
					temp[i]='z';
				if(temp[i]=='@')
					temp[i]='Z';
			}
		}
		result = result + temp[i];
	}
	return result;
}
```

2. 다중 채팅

개요

-명　　칭: 다중 채팅

-개발인원: 1명

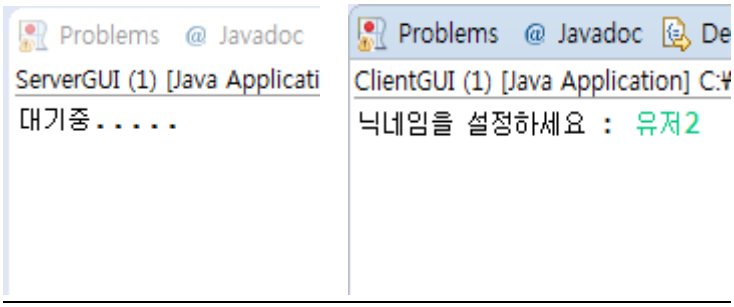
-오픈 소스 사용여부: O

-개발환경: JAVA 8 eclipse 사용

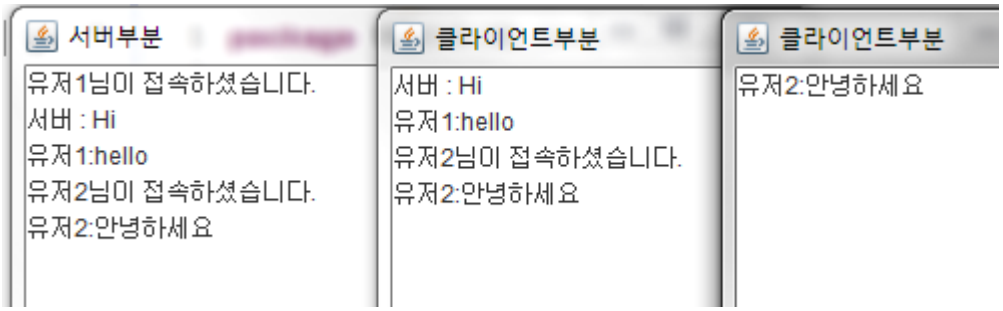
-소스코드: https://github.com/HwaSeok/multi-chatting

소개

1:1이 아니라 여러 사람이 동시에 채팅을 하기 위해 개발한 프로그램입니다. 쓰레드를 구현하여 동시에 채팅을 할 수 있게 하고 닉네임을 정하여 서로를 구별합니다. 1:1채팅과는 다르게 서버가 계속 클라이언트를 받아줘야 하기 때문에 listen상태를 유지합니다.



위와 같이 서버는 계속 대기하고 클라이언트는 실행할 때 콘솔 창에 자신의 닉네임을 입력하고 접속합니다.



클라이언트가 접속할 때 마다 서버는 쓰레드를 생성하고 실행시켜 주며 매번 메시지를 전송하여 클라이언트가 접속되었음을 알립니다.

소스코드 일부

ServerBackground.java

서버 백그라운드의 setting메소드와 내부클래스인 Reciver입니다.
Setting은 클라이언트의 요청을 수락하고 쓰레드를 생성, 실행하는 동작을 반복합니다.
Receiver는 쓰레드를 상속하여 네트워크의 처리를 계속 듣고 처리합니다.

```
public void setting() {

        try {

            Collections.synchronizedMap(clientMap);
            serverSocket = new ServerSocket(7777);

            while (true) {
                System.out.println("대기중.....");
                socket = serverSocket.accept();
                System.out.println(socket.getInetAddress() + "에서
접속했습니다.");

                Receiver receiver = new Receiver(socket);
                receiver.start();
            }

        } catch (IOException e) {
            e.printStackTrace();
        }
    }

class Receiver extends Thread {
        private DataInputStream in;
        private DataOutputStream out;
        private String nick;

        public Receiver(Socket socket) {
            try {
                out = new DataOutputStream(socket.getOutputStream());
                in = new DataInputStream(socket.getInputStream());
                nick = in.readUTF();
                addClient(nick,out);
            } catch (IOException e) {
                e.printStackTrace();
            }
        }

        @Override
        public void run() {

            try {
                while (in != null) {
```

```
                msg = in.readUTF();
                sendMessage(msg);
                gui.appendMsg(msg);
            }
        } catch (Exception e) {
            removeClient(nick);
        }
    }
}
```

3. 메모리 파워

개요

-명　　칭: 메모리 파워

-개발인원: 1명

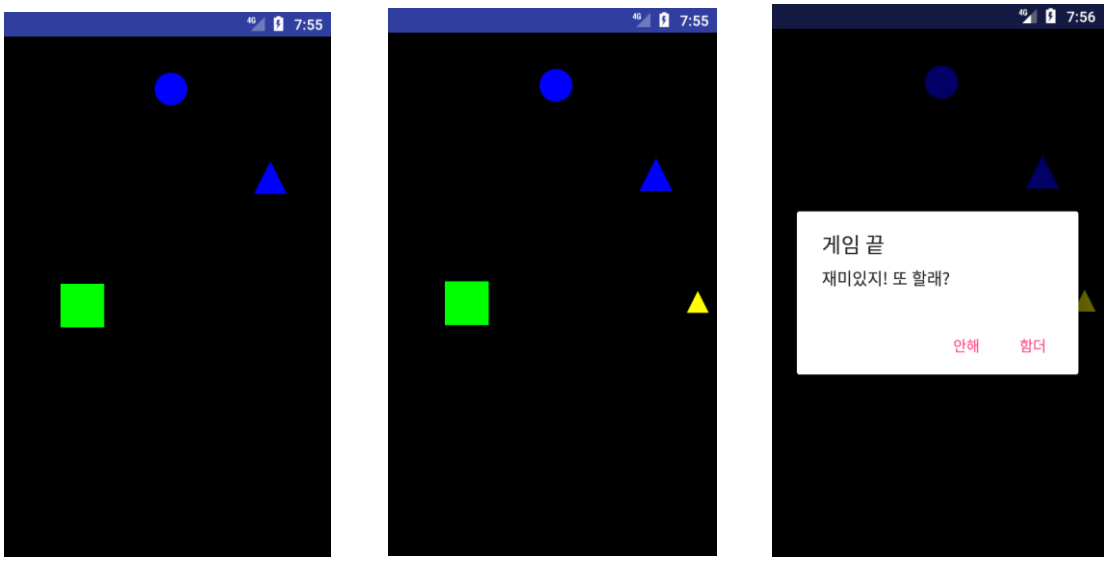
-오픈 소스 사용여부: O

-개발환경: Android Studio 2.3.1 사용

-소스코드: https://github.com/HwaSeok/memory-power

소개

메모리파워는 재미와 기억력향상을 위해 안드로이드 환경에서 개발한 간단한 게임입니다. 게임이 진행될 때마다 화면에 임의의 도형이 하나씩 생성됩니다. 사용자는 이전의 화면을 기억해 두었다가 가장 최근에 생긴 도형을 터치해야 계속 진행할 수 있으며 틀린 경우에 게임을 다시 시작할지, 종료할지 묻는 알림 창이 나타납니다.



나온 도형을 2번까지 맞춘 상황입니다. 세 번째 도형을 맞추는데 성공하면 잠깐 검은 화면이 나온 후 새로운 도형이 추가됩니다. 만약 맞추지 못했다면 마지막 그림 같은 알림 창이 생성됩니다.

소스코드 일부

MemoryPower.java

새로운 도형을 목록에 추가하는 AddNewShape 메소드의 일부입니다.
새로운 도형의 위치와 크기를 랜덤으로 정하고 목록에 있는 도형과 겹치지 않는지 확인합니다. 기존의 도형과 겹치지 않고 화면 밖으로 나가지 않는 도형이 생성 될 때까지 반복합니다.

```
 Shape shape = new Shape();
 int idx;
 boolean bFindIntersect;
 Rect rt = new Rect();

for(;;){
    int Size = 32 + 16 * Rnd.nextInt(3);

    rt.left = Rnd.nextInt(getWidth());
    rt.top = Rnd.nextInt(getHeight());
    rt.right = rt.left + Size;
    rt.bottom = rt.top + Size;

    if(rt.right>getWidth() || rt.bottom>getHeight()){
        continue;
    }

    bFindIntersect = false;
    for(idx=0; idx<arShape.size(); idx ++){
        if (rt.intersect(arShape.get(idx).rt) == true) {
            bFindIntersect = true;
        }
    }

    if(bFindIntersect == false){
        break;
    }
}
```