JAVA 프로젝트

TCP서버와 GUI인터페이스를 이용한 2D 그래픽 RPG 게임
프로젝트 온누리

20200925 이우람

목차

1.프로젝트 목적 및 개발툴

프로젝트의 목적

프로그래밍을 배운 후부터 계속해서 들었던 의문을 해결하고 싶은 마음에 예전부터 계속해서 시도해오며, 이번엔 텍스트기반이 아닌 그래픽기반의 인터페이스로 설계

게임을 실행시키고 난 뒤의 작동원리, 캐릭터의 움직임, 레벨이 오르거나, 공격을 할 때의 단순한 행동에도 하나하나 있을 소스를 직접 설계해보고 싶은 마음에 프로젝트 주제로 선정

개발툴
Java SE-1.8버전 - 툴 Eclipse 사용
Oacle 11g 버전 - 툴 SQLDeveloper 사용
GitHub 연동

30	31	1	2	3	4	5
6	7	8	9	10	11	12
13	**14**	**15**	**16**	**17**	**18**	**19**
20	**21**	**22**	**23**	**24**	**25**	26
27	28	29	30	1	2	3

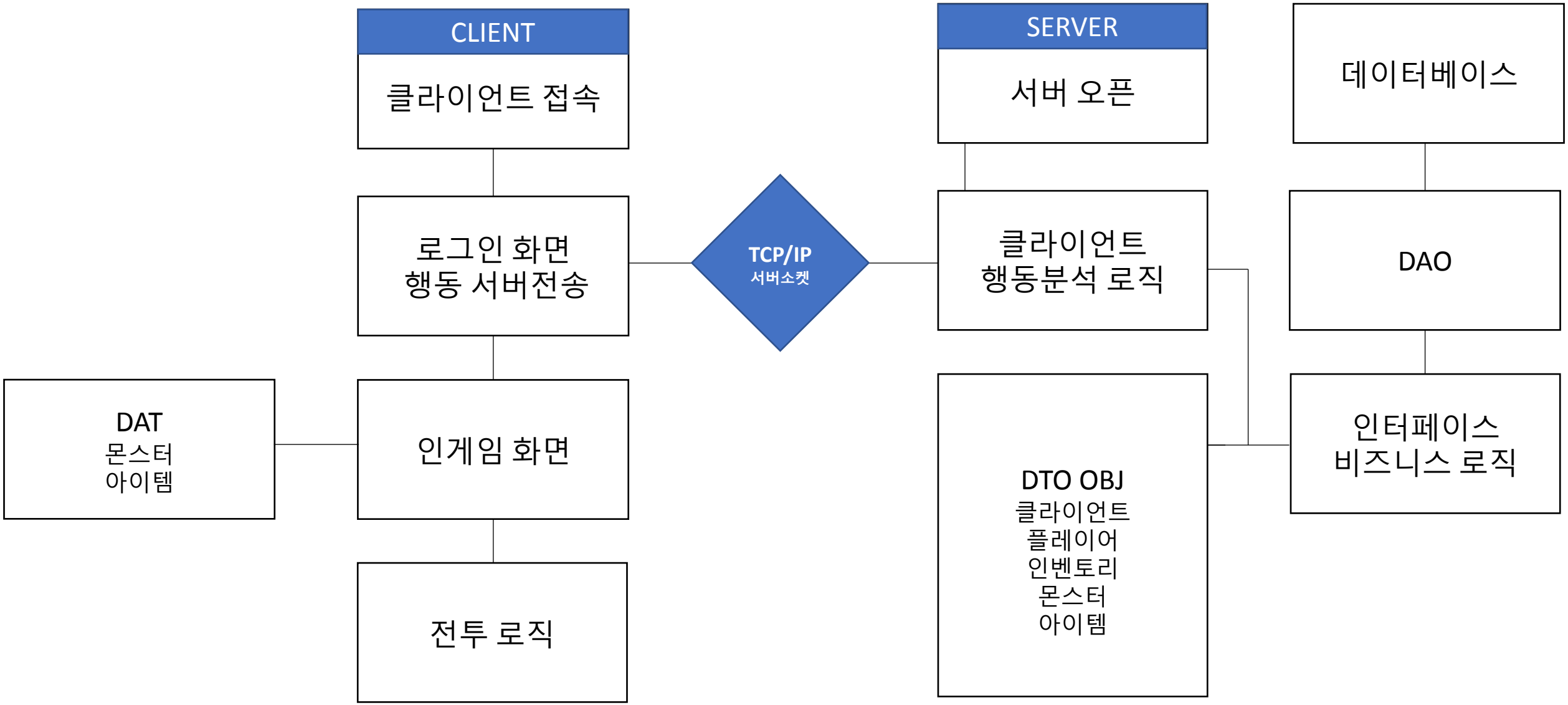
프로젝트 기간 9월 15일 - 9월 25일

14일~15일 프로젝트 구상 및 설계
16일~18일 JAVA DAO DTO 기본 로직설계
19일~22일 JAVA GUI 설계
23일~24일 JAVA TCP 서버구축 및 연동
25일 마무리 작업 및 PPT 제작

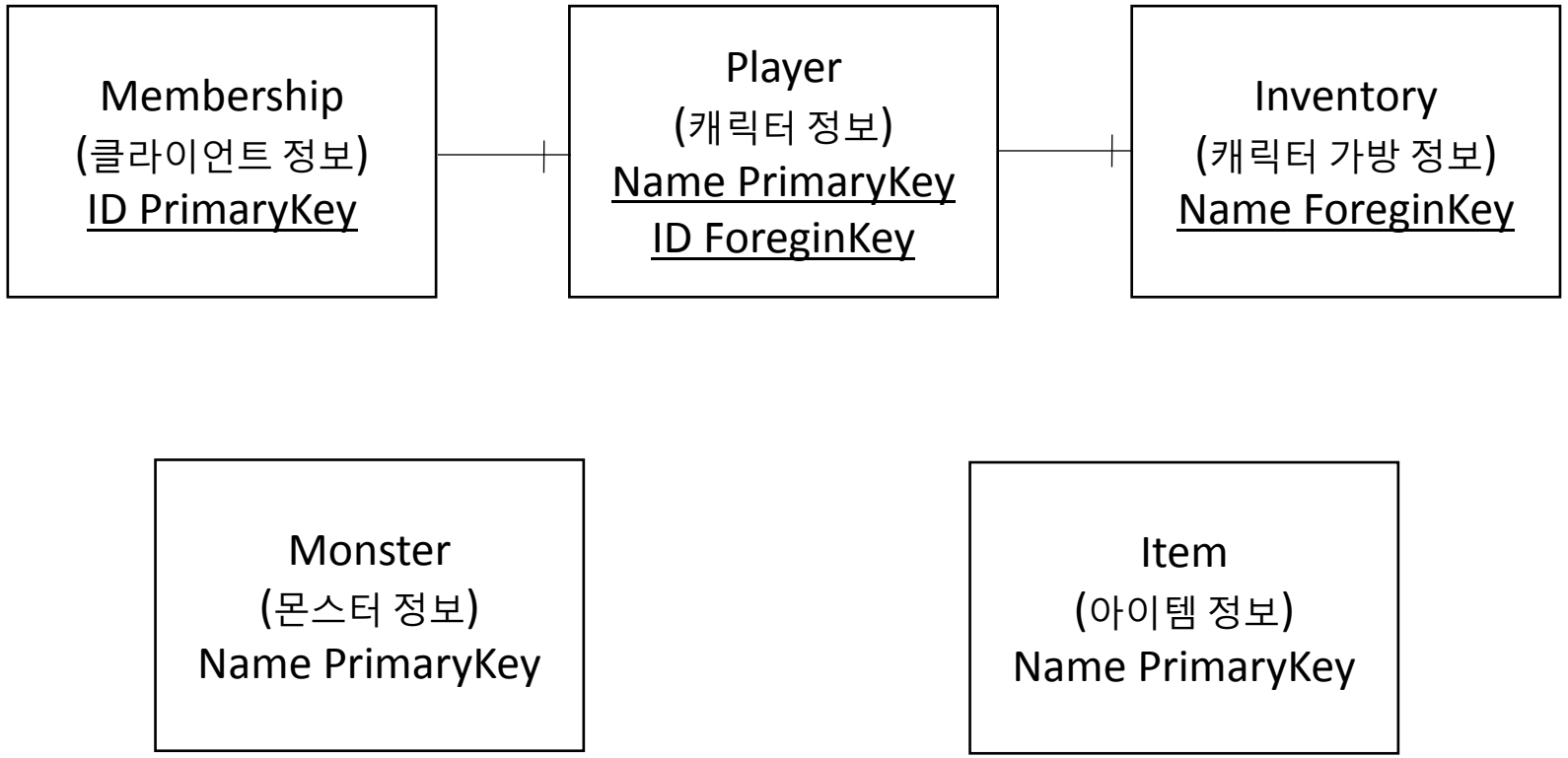
2.주요 사용기술

- 멀티쓰레드를 이용한 TCP 서버 구축 및 GUI 내부적인 동시 실행 설계
- GUI 더블버퍼링을 이용한 이미지 움직임 설계
- 내부적인 객체를 외부 데이터파일로 변환 스트림을 이용하여 불러오기
- TCP 서버와 클라이언트 GUI 사이의 통신을 이용한 유효성 검사
- 클라이언트의 접속 기록인 LOG 서버 내부적인 기록

3.흐름도


CLIENT
클라이언트 접속
로그인 화면
행동 서버전송
TCP/IP
서버소켓
SERVER
서버 오픈
클라이언트
행동분석 로직
데이터베이스
DAO
인터페이스
비즈니스 로직
DAT
몬스터
아이템
인게임 화면
전투 로직
DTO OBJ
클라이언트
플레이어
인벤토리
몬스터
아이템

4.DB 테이블 구조 및 다이어그램



4.DB 테이블 구조 및 다이어그램

Membership

속성이름	속성설명	속성타입	제약조건
NO	고유번호	Number(5)	Not null
ID	아이디	Varchar2(10)	Primary key
PW	비밀번호	Number(4)	Not null

Inventory

속성이름	속성설명	속성타입	제약조건
NAME	이름	Varchar2(10)	Foreign key
WEAPON	장착무기	Varchar2(20)	Not null
ARMOR	장착방어구	Varchar2(20)	Not null
GOLD	보유골드	Number(6)	Not null
LIST	보유아이템	Varchar2(100)	

Player

속성이름	속성설명	속성타입	제약조건
NAME	이름	Varchar2(10)	Primary key
LV	레벨	Number(3)	Default 1
EXP	현재경험치	Number(10)	Default 0
MAXEXP	최대경험치	Number(10)	Default 10
HP	현재체력	Number(5)	Default 100
MAXHP	최대체력	Number(5)	Default 100
ATTACK	공격력	Number(5)	Default 50
DEFENCE	방어력	Number(5)	Default 5
STRENGTH	힘	Number(3)	Default 5
AGILITY	민첩	Number(3)	Default 5
POINT	포인트	Number(3)	Default 0
GENDER	성별	Varchar2(4)	Not null
ID	아이디	Varchar2(10)	Foreign key

4.DB 테이블 구조 및 다이어그램

Item

속성이름	속성설명	속성타입	제약조건
GRADE	아이템등급	Number(1)	Not null
TYPE	아이템타입	Varchar2(10)	Not null
NAME	아이템이름	Varchar2(20)	Primary key
STAT	증가수치	Number(5)	Not null
PRICE	아이템가치	Number(5)	Not null

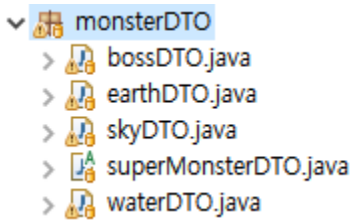
Monster

속성이름	속성설명	속성타입	제약조건
NAME	이름	Varchar2(10)	Primary key
LV	레벨	Number(3)	Not null
ATTACK	공격력	Number(5)	Not null
DEFENCE	방어력	Number(5)	Not null
LIST	보유아이템	Varchar2(60)	Not null
MAP	등장지역	Varchar2(20)	Not null
COLSIZE	세로좌표	Number(4)	Default 0
ROWSIZE	가로좌표	Number(4)	Default 0

5.주요소스 설명<몬스터클래스>

```
create or replace view earth_monster as select * from monster where map_name = '땅';
create or replace view water_monster as select * from monster where map_name = '물';
create or replace view sky_monster as select * from monster where map_name = '하늘';
create or replace view boss_monster as select * from monster where map_name = '미구현';
```

뷰를 이용하여 몬스터 등장지역 별로 구분



등장지역마다의 몬스터를 구분짓기 위해 상속받을 수 있는 슈퍼클래스 생성
해당 **슈퍼클래스**를 이용하여 모든 몬스터의 정보를 ArrayList 상에 등록

```
public abstract class superMonsterDTO implements Serializable {
    private String name;// 몬스터이름
    private int level;// 몬스터레벨
    private int hp;// 몬스터체력
    private int attack;// 몬스터공격력
    private int defence;// 몬스터방어력
    private String[] list = new String[3];// 보유 아이템 리스트
    private String map_name;// 출연 맵
    private int colsize;// 출연위치 세로
    private int rowsize;// 출연위치 가로

    private static final long serialVersionUID = 1L;
```

5.주요소스 설명<유효성 검사>

```
public void msgAnalysis(String id, String msg) {
    ClientConnect chk = clientList.get(id);
    String[] analysis = msg.split(" ");
    if (msg.indexOf("@login") >= 0) {
        chk.export("@login " + String.valueOf(player.login(a
                + String.valueOf(player.load(analysis[1])) +
    } else if (msg.indexOf("@join") >= 0) {
        chk.export("@join " + String.valueOf(player.join(ana
    } else if (msg.indexOf("@drop") >= 0) {
        chk.export("@drop " + String.valueOf(player.drop(ana
    } else if (msg.indexOf("@create") >= 0) {
        player.create(analysis[1], analysis[2], analysis[3])
    } else if (msg.indexOf("@load") >= 0) {
        chk.export("@load " + startSet(analysis[1]));
    } else if (msg.indexOf("@save") >= 0) {
        startGet(msg);
    } else if (msg.indexOf("@exit") >= 0) {
        clientList.remove(id);
        System.out.println(clientList.size()+"명의 접속자");
    } else if (msg.indexOf("@all") >= 0) {
        allClient(msg);
    }
}
```

```
public interface allINF {
    public boolean memberLogin(String id, int pw);//로그인

    public boolean memberJoin(String id, int pw);// 회원가입

    public boolean memberDrop(String id, int pw);// 회원탈퇴

    public boolean playerChk(String id);// 캐릭터생성여부

    public void createPlayer(String id, String name, String gender);// 캐릭터생성

    public startLogic startInfo(String id);// 시작 로딩

    public ArrayList<superMonsterDTO> inGameInfo(String name);// 게임 로딩

    public void update(playerDTO p, inventoryDTO i);// 유저정보갱신

}
```

메인서버와 클라이언트화면의 소통을 구분짓기 위해 특정한 문자를 지정
각각의 행동에 맞는 로직을 통하여 인터페이스 호출
인터페이스에선 boolean을 통하여 **유효성 검사** 후에 DB 상의 데이터를 불러오기

5.주요소스 설명<DAT 로드>

```
public ArrayList<superMonsterDTO> monster(String map) throws
    String mon = null;
    switch (map) {
    case "땅":
        mon = "dat/earthMonster.dat";
        break;
    case "물":
        mon = "dat/waterMonster.dat";
        break;
    case "하늘":
        mon = "dat/skyMonster.dat";
        break;
    }

    FileInputStream fis = new FileInputStream(mon);
    BufferedInputStream bis = new BufferedInputStream(fis);
    ObjectInputStream ois = new ObjectInputStream(bis);

    ArrayList<superMonsterDTO> list = new ArrayList<>();
    superMonsterDTO m = null;
    String msg = (String) ois.readUTF();

    for (int i = 0; i < 6; i++) {
        m = (superMonsterDTO) ois.readObject();
        list.add(m);
    }

    ois.close();
    bis.close();
    fis.close();
    return list;
```

```
public ArrayList<itemDTO> item() throws IOException, ClassNo
    String item = "dat/item.dat";

    FileInputStream fis = new FileInputStream(item);
    BufferedInputStream bis = new BufferedInputStream(fis);
    ObjectInputStream ois = new ObjectInputStream(bis);

    ArrayList<itemDTO> list = new ArrayList<>();
    itemDTO m = null;
    String msg = (String) ois.readUTF();

    for (int i = 0; i < 25; i++) {
        m = (itemDTO) ois.readObject();
        list.add(m);
    }

    ois.close();
    bis.close();
    fis.close();
    return list;
}
```

```
<terminated> fileStreamTest (1) [Java Application] C:₩Program Files₩Java₩
onnuri earthMonster ver 1
onnuri skyMonster ver 1
onnuri waterMonster ver 1
onnuri item ver 1
```

아직은 완벽하게 구현하지 못한 소스
DB 내부의 정보를 받아와 DAT 파일을 만들고 해당 **파일을 스트림을 이용하여 클라이언트 폴더에 저장하는 기술**을 구현하고 싶었으나, 구현하지 못하였음
각각의 데이터파일 내부에 객체와 함께 스트링UTF 형식으로 버전을 저장하여 해당 기록과의 비교로 업데이트하려는 내용을 구상
현재의 소스는 만들어진 DAT 파일을 불러와 해당 정보를 인게임 상에서 사용가능한 소스

5.주요소스 설명<더블버퍼링>

```
@Override
public void paint(Graphics g) {
    img = createImage(800, 450);
    imgG = img.getGraphics();
    if (gender != null) {
        imgG.drawImage(Toolkit.getDefaultToolkit().getImage("ima
        imgG.drawImage(pla, x, y, this);
    } else {
        imgG.drawImage(image, 0, 0, this);
    }
    g.drawImage(img, 0, 0, this);
}

@Override
public void update(Graphics g) {
    paint(g);
}
```

```
public void sound(String file, boolean loop) {
    try {
        AudioInputStream ais = AudioSystem.getAudioInputStream(new BufferedInputStream(
        clip = AudioSystem.getClip();
        clip.open(ais);
        clip.start();
        if (loop)
            clip.loop(-1);
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

기존의 페인트는 프레임 상에 바로 이미지를 입는 방식이지만 이렇게 하면 한번의 이미지가 움직일 때마다 딜레이에서 생기는 끊김현상이 생긴다
그걸 방지하기 위해 이미지를 하나 더만들고 해당 이미지에 미리 다음에 오게 될 이미지를 그리고 반복적으로 이미 그려진 그림을 불러오는 것으로 이걸 방지할 수 있으며 이 기술을 **더블버퍼링**이라 부른다
새로운 이미지를 계속 그리는 것이 아닌 이미 그려진 이미지를 불러와 딜레이없이 깔끔한 움직임을 볼 수 있다
위와는 다르지만 이미지처럼 오디오를 불러올 수 있는 오디오스트림 역시 존재하며 루프를 이용하여 계속 오디오가 진행될 지 한 순간만 진행될 지 지정가능하다
GUI는 소스의 복잡함보다는 검색의 기술이 중요하다 위의 두 기술 역시 찾아보면 금방 인터넷 상에 존재하는 기술이다

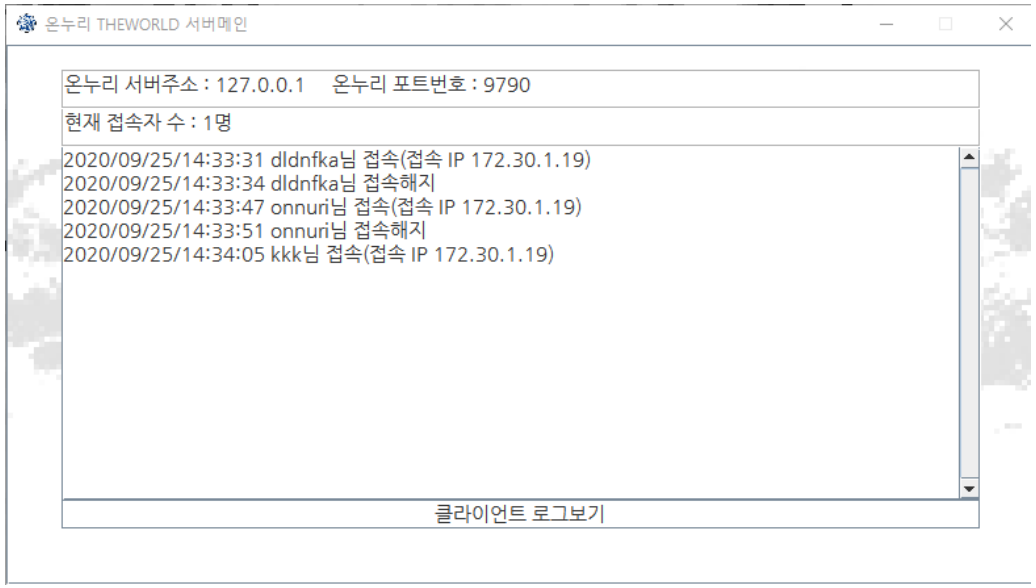
5.주요소스 설명<LOG 기록>

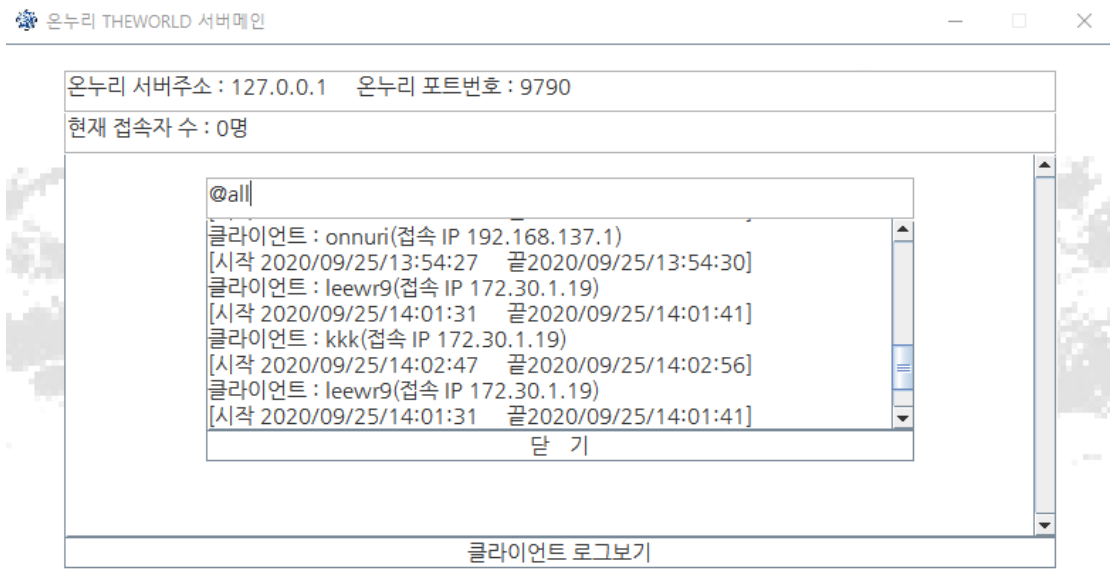
```java
public class logDTO {
    private String id;// id
    private String address;// ip
    private String startTime;// 접속시작 시간
    private String endTime;// 접속종료 시간
    private SimpleDateFormat sdf = null;//시간 받아오기
    private Calendar cal = null;//현재시간으로 받기

    public void set(String id, String address) {
        this.id = id;
        this.address = address;
        sdf = new SimpleDateFormat("yyyy/MM/dd/HH:mm:ss");
        cal = Calendar.getInstance();
        this.startTime = sdf.format(cal.getTime());
    }

    public void setEnd() {
        sdf = new SimpleDateFormat("yyyy/MM/dd/HH:mm:ss");
        cal = Calendar.getInstance();
        this.endTime = sdf.format(cal.getTime());
    }

    public void load(String id, String address, String start, String end) {
        this.id = id;
        this.address = address;
        this.startTime = start;
        this.endTime = end;
    }
```

```java
private String id = null;
private String clientHostAddress;

public ClientSetting() {
    try {
        clientHostAddress = InetAddress.getLocalHost().getHostAddress();
        startin = new startIn(this);
    } catch (Exception e) {
    }

}

public void connect(String i) {
    try {
        i += " " + clientHostAddress;// 아이디와 아이피 주소보내기
        client = new Socket("localhost", 9790);
        out = client.getOutputStream();
        out.write(i.getBytes());
```

게임 내부 서버에 접속한 클라이언트의 기록을 남길 수 있게 클라이언트가 접속할 시에
클라이언트의 ID와 함께 IP를 서버로 넘기고, 그걸 받아 logDTO라는 또 다른 객체에 저장,
벡터를 이용하여 필요할 시에 다시 받아오는 구조로 설계
InetAddress 클래스는 자바에서 IP 주소를 다루기 위한 클래스로써 나의 ip주소를 받아와
String 자료형으로 변환시켜 대입시킨 뒤에 서버로 송신
서버에서는 클라이언트 정보를 받은 뒤에 **SimpleDateFormat 클래스(날짜, 시간을 간단히 받아오는 클래스)**
를 이용하여 시작시간을 함께 입력 종료 시에는 다시 종료시간을 입력

5.주요소스 설명<LOG 기록>





현재 접속해 있는 클라이언트의 로그 확인 및, 특정 아이디나 전체 목록을 확인할 수 있는 걸 알 수 있음
LOG 테이블에서 받아온 내용을 서버 GUI에서 확인 가능

5.구현영상 및 깃허브

프로그래밍은 맛있다. 카페를 이용해주세요
구현영상

서버 깃허브
클라이언트 깃허브

6.후기

아쉬움이 많이 남는다. 시작은 간단 하게 하려 했으나, 하나를 진행하면 할 수록 생각을 하게 되고 생각을 할 수록 필요한 기능은 점점 늘어가, 시작은 작은 개울이었지만, 끝은 바다처럼 크게 되어 버렸다. 그로 인해 몇가지는 포기하게 되고, 상황에 맞게 계획을 바꿔갔지만 끝끝내 구현 못한 기능들이 있어 그것이 가장 아쉽다.

반면에 프로젝트를 진행하며 내가 현재 몰랐던 기술을 알아갈 수 있었고 그 기술을 내 것으로 만드는 과정은 정말 유익한 시간이었으며, 구현해낸 기술들을 하나, 하나 생각하면 못했던 것에 대한 아쉬움도 있지만, 해낸 것에 대한 만족감도 있다. 여러 예외들을 보면서 다음에는 이와 같은 실수를 안 할 수 있게 해준 과정은 만족하고 있다.

게임이란 것이 소스 뿐만이 아닌 그래픽 적인 것 역시 중요하므로, 그와 관련된 전문기술이 전무한 내가 할 수 있는 건 한정적이라 프로젝트란 것을 혼자 하는 것이 얼마나 힘든 지, 왜 팀으로 해야 하며, 한 가지의 프로그램이 나오기까지의 기간이 오래 걸리는지 알게 된 시간이었다.

그리고 역시 프로젝트를 하며 느끼지만, 삽질, 즉 노력을 많이 하면 할수록 머리는 힘들지만 몸이 노력을 알아가고, 전혀 처음 본 문외한 기술이라도 지금에 와서는 익숙하게 다룰 수 있다는 게 기분이 좋다. 주요소스에서는 설명하지 못했지만 대다수의 GUI와 서버 내부적으로 쓰레드가 돌고 있는데, 해당 기술을 처음 배웠을 때만해도 어떻게 쓰는지 감도오지 않았지만, 지금은 어떻게, 언제, 왜 써야 하는지를 알게 된 유익한 과정이었다.