컴퓨터 응용및실습 팀프로젝트

멀티 스레드를 이용한 다중 통신 다빈치코드 게임 구현

과목명 : 컴퓨터 응용 및 실습

담당 교수 : 최윤정 교수님

201611299 정희승

201212519 김선우

201611262 박동현

201811296 정현재

201811297 주윤지

1. 주제 및 목표

1.1 프로젝트 주제

멀티 스레드를 이용한 다중 통신 다빈치코드 게임 구현

1.2 프로젝트 목표

- MVC 패턴에 입각한 프로그램 설계 및 구현 (Class Diagram, Sequence Diagram 이용)
- 스레드, 소켓 통신, DB, tts 등 강의 내용을 적극적으로 이용하고 활용능력의 향상
- 분야에 따라 개별 구현 후 병합 과정의 경험
- 최대한 다양한 응용 어플리케이션 사용

1.3 팀 목표

- 협업을 위한 GITHUB 등의 프로젝트 도구 사용
- 학습 분야 또는 개인의 목표에 따라 역할 분배
- 원활한 의사소통 및 팀모임을 통한 한계점 극복 및 학습 내용 공유

2. 요구사항 분석

2.1 기능 요구사항

2.1.1 다빈치코드 보드게임의 설계 및 구현

- 게임 구성(Component)

  흰 블록 12개(0 - 12번), 검은 블록 12개(0 – 12번), 조커 블록 2개("-"으로 표시되고 어느 숫자 든 될 수 있다.)

- 게임 규칙

  a) 플레이어가 3명 이하인 경우 4개씩, 4명일 경우 3개씩 자신의 패를 가져오고 크기 순으로 정렬한다. (같은 숫자의 흰색 블록과 검은색 블록의 위치는 상관없다.)

  b) 자신의 차례마다 소유되지 않은 블록을 하나 가져오고 다른 플레이어의 블록을 지목하여 숫자를 예상한다.

c) 숫자를 맞춘 경우 블록을 공개하고, 틀린 경우 자신의 차례에 가져온 블록을 공개한다. 지목에 성공한 경우 차례를 넘길 수도 있고, 다른 플레이어의 블록을 지목할 수 있다.

d) 모든 블록이 공개되면 해당 플레이어는 게임에서 제외된다. 마지막까지 남은 플레이어가 게임의 승자가 된다.

- 모든 플레이어는 준비되지 않음/준비완료 상태로 구분되고, 모든 플레이어가 자신의 프로필을 클릭하여 준비완료 되어야 게임을 시작할 수 있다.
- 게임이 시작되면 심판은 블록을 sorting하여 플레이어에게 제공한다.
- 플레이어가 지목을 원하는 블록은 클릭을 통해 선택, 예상되는 블록의 종류는 라디오버튼과 콤보박스로 선택할 수 있게 한다.
- 플레이어가 지목에 성공하면 지목을 계속, 턴 종료 중 선택할 수 있게 구현한다.
- 플레이어가 게임에서 이탈하게 되면 무승부 처리된다.

2.1.2 JDBC, MySQL를 이용한 유저정보 관리

- MySQL에 UserList라는 테이블을 생성하여 아이디, 패스워드, 승리, 패배 정보를 저장한다.
- 계정 생성 시, UserList의 ID 열을 검색하여 중복되는 아이디가 존재하는지 검사하고 중복되는 아이디가 없는 경우 새로운 사용자를 추가한다.
- 로그인 시, UserList의 ID 열을 검색하여 입력된 ID와 동일한 값을 가지는 행 전체를 가져와 해당 행의 패스워드와 입력된 패스워드를 비교하여 로그인을 수행한다.
- 전적 조회시, UserList의 ID 열을 검색하여 사용자 ID와 동일한 값을 가지는 행의 win, lose 값을 가져와 state 객체로 반환한다.
- 게임 종료 시, DB에 저장된 승리 플레이어와 패배 플레이어 각각의 유저정보에 접근해 전적 업데이트를 실행한다.

2.1.3 TTS를 이용한 심판의 게임 진행

- tts를 이용해 어느 플레이어의 차례인지 사용자에게 음성으로 알려준다.
- 계정 생성, 로그인 시 완료 메시지를 음성으로 출력한다.
- 게임 종료 시, 종료 선언과 승리자 선언을 음성으로 출력한다.

2.2 GUI 요구사항

2.2.1 메인 프레임의 구성

- 여러 개의 JFrame을 연결시키는 대신 하나의 JFrame 창을 두고 전환될 화면들을 JPanel 클래스화 하여 화면을 변경한다.
- 메인 JFrame 클래스가 JPanel 클래스 들을 has 하는 구조로 구현한다. 그것으로 각 패널의 컴포넌트에 접근이 용이하도록 하고, 필요하다면 그러한 이점을 이용하여 메인 JFrame 클래스에서 각 패널의 컴포넌트에 필요한 actionListener를 간편하게 추가한다.

2.2.2 로그인 패널 & 계정 생성 프레임의 구성

- 로그인 화면을 메인 JFrame에 포함될 JPanel 클래스로 구현한다.
- 로그인 화면에서 계정 생성 기능을 이용할 수 있도록 한다.
- 계정 생성 기능은 계정 생성 JFrame에서 처리하도록 하고, 로그인 화면에 계정 생성 버튼을 구현하여 계정 생성 JFrame에 링크되도록 한다.
- 로그인이 간편해지도록 ID와 비밀번호를 입력 받은 상태에서 Enter를 누르면 로그인 버튼을 누르는 것과 동일한 기능을 하도록 한다.

2.2.3 방 목록 패널의 구성

- 방 입장 버튼 구성 시, 빈 방은 '빈 방'으로, 유저가 참가하고 있는 방은 '입장'버튼으로 구별할 수 있도록 한다.
- 방 입장버튼이 '입장'버튼으로 변경될 시에 방에 소속된 유저의 인원 또한 화면에 송출하도록 한다.
- 방 만들기를 수행하는 버튼을 따로 만들 필요 없이 '빈 방' 상태인 입장버튼을 클릭 시에 게임 방을 생성하도록 한다.

2.2.4 게임 진행 패널 구성

- 방 목록 패널로 돌아갈 수 있는 버튼을 패널에 추가한다.
- 빈 유저, 접속한 유저, 준비 완료된 유저, 죽은 유저의 이미지를 서로 다르게 구성하여 사용자로 하여금 구별할 수 있도록 한다.
- 게임 진행시에 받을 블록을 저장할 JLabel을 추가하고, 상대방 블록Label에 저장된 블록의 숫자는 볼 수 없도록 한다.
- 유저가 이용할 수 있고 게임 진행 상황을 알려주는 채팅 창을 구현한다.

- 채팅 창에 자동 스크롤과 자동 줄 바꿈 기능이 추가되도록 한다.
- 최대한 간결하고 깔끔하게 보일 수 있게 컴포넌트 투명화 메소드를 활용하도록 한다.

2.3 서버-클라이언트 요구사항

2.3.1 소켓 통신을 이용한 서버-클라이언트 구조 구축

- TCP/IP 프로토콜을 통해 Error control을 수행한다.
- 이외에 통일된 통신 프로토콜을 활용해 로그인, 회원 가입, 게임 진행 등을 수행한다.
- 예외 처리문을 통해 통신상 발생 가능한 예외 상황 처리한다.

2.3.2 멀티 스레드를 이용한 다중 게임플레이 지원

- 서버에서 각 유저의 고유 쓰레드를 관리하며 통신한다.
- 멀티 쓰레드에서 같은 자원에 접근할 때 동기화 처리

2.3.3 각 방을 관리할 서브 서버

- 방에 입장중인 클라이언트를 따로 관리할 수 있어야 한다.
- DB와 연동하여 전적을 관리할 수 있어야 한다.
- GUI에게 필요한 정보를 제공해 주어야 한다.

2.3.4 채팅, 게임 기능 제공

- 클라이언트에게 받은 정보를 서브 서버의 다른 클라이언트들에게도 전달해준다.
- 받은 정보를 기반으로 게임 모델을 활용해 게임 진행을 해야 한다.
- MVC 구조 유지를 위해 필요이상으로 게임에 관여하지 않는다.

2.4 데이터베이스

2.4.1 MySQL과 Java의 연동

- 서버 환경에 MySQL을 구축해준다.
- DB 서버에 미리 정해진 데이터베이스, 테이블의 폼을 제작해둔다.
- JDBC를 통해 Java와 연동

2.4.2 서버에서 관리될 유저 정보들

- ID를 identifier로 같은 id가 존재하지 않도록 관리한다.
- ID를 통해 비밀번호, 전적을 찾을 수 있다.
- 로그인, 회원 가입에 이용 가능해야 한다.
- 전적 관리로 승패에 따라 전적이 업데이트 된다.

2.4.3 동기화

- 멀티 쓰레드에서 동시에 DB에 쿼리를 던지지 않도록 한다.
- 모두 같은 DB에서 사용되도록 한다.

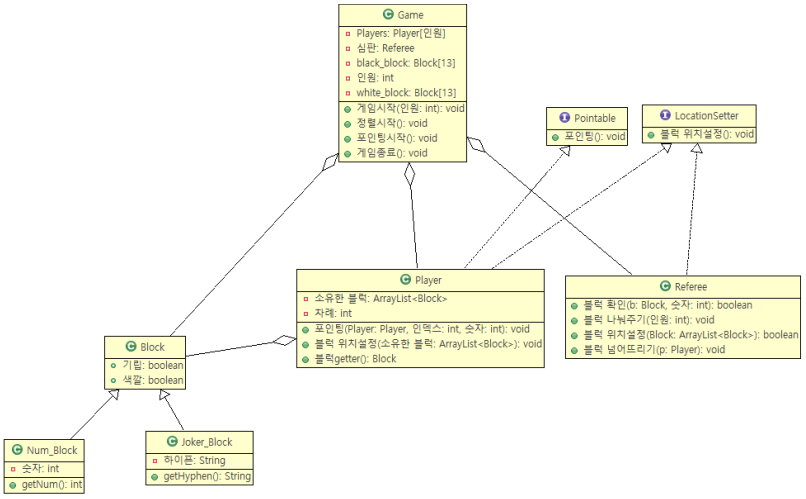
2.5 제약사항

1) 서버-클라이언트 통신 상에서 에러 및 딜레이 발생
   - 메소드 Synchronize 동기화
   - Try-catch-finally 를 통한 소켓 통신 예외처리
2) 게임 진행, 심판 멘트, 채팅 관련 데이터의 분리 필요성 제기
   - 새로운 프로토콜 규정함으로써 진행, 심판, 채팅 데이터에 각각 다른 헤더를 부착해서 구분한다.
3) 한국어 지원 TTS 오픈소스 라이브러리의 부재
   - 네이버 N클라우드의 Clova CSS 사용 (유료)

3. 설계

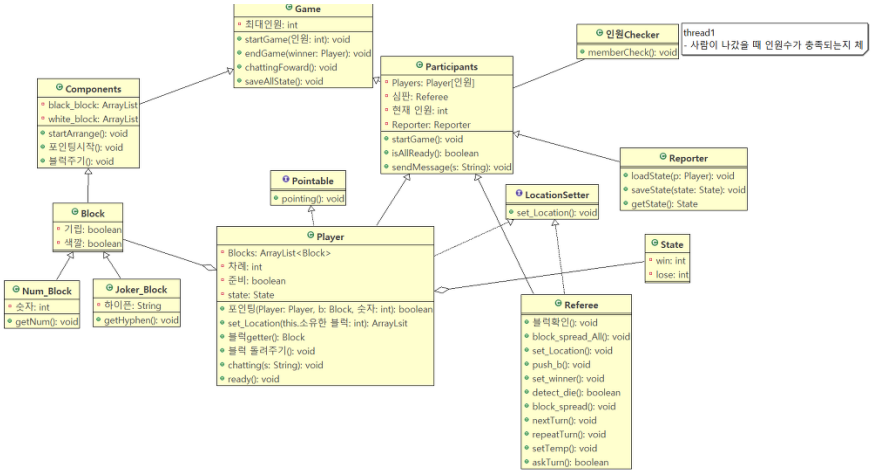
3.1 클래스 다이어그램

3.1.1.1 게임 : 초기 설계



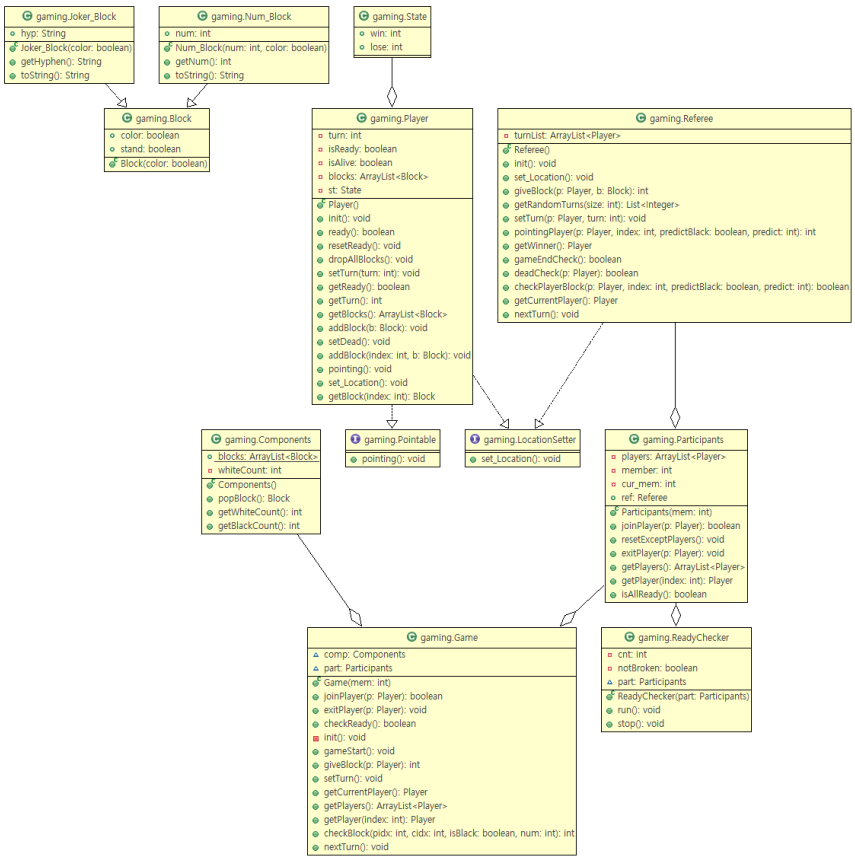
게임에 필요할 필수적인 클래스들을 구성하고, 게임을 진행시킬 클래스인 Referee 클래스를 추가하였다.

3.1.1.2 게임 구현 직전 설계



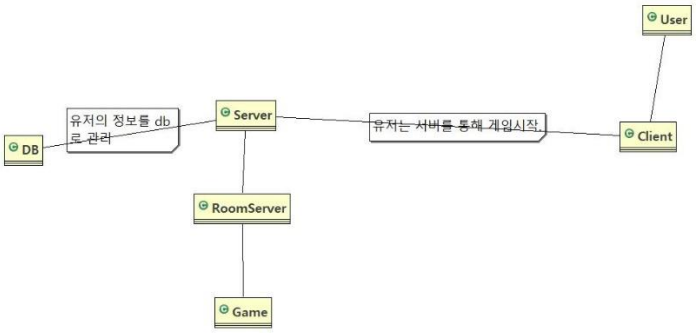
여러 번의 팀 미팅과 교수님의 조언을 바탕으로 클래스를 세분화 하고자 하였고, Block을 가지는 Components 클래스와 Player/Referee를 가지는 Participants 클래스를 추가하였다. 초기 설계보다 모듈화가 된 모습이다.

3.1.1.3 게임 : 구현 후 UML 모습



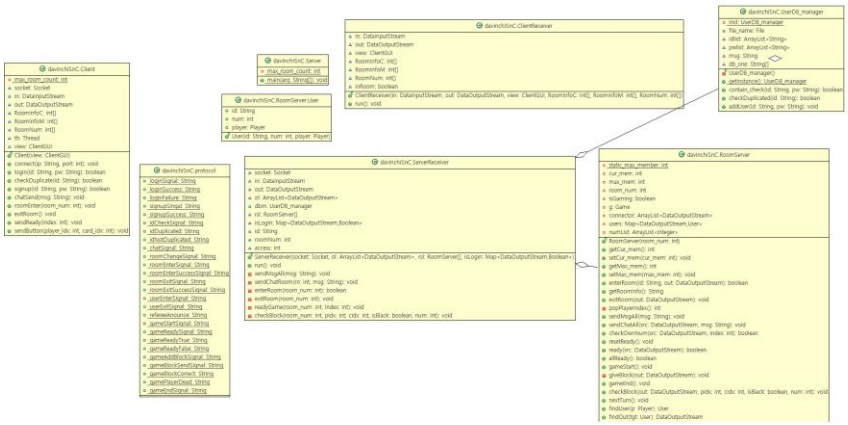
몇 메소드가 추가되었으나 큰 틀은 유지되었다. 기존의 설계를 거의 이어받아 구현되었다고 볼 수 있다. MVC 구조에서 Model의 역할을 하며 Controller에 의해 수행되도록 수동적인 메소드 위주로 구성되었다. Model 역할을 Game 클래스가 수행하므로 Game클래스에서 다른 class들을 호출하는 방식 위주로 설계되었다.

3.1.1.4 서버 : 초기 설계



추상적으로 분석된 구조. 서버는 메인 서버와 방을 관리해줄 서브 스레드들, 방 서브 서버가 갖는 Game 모델들로 구성. 데이터베이스 서버는 따로 독립되어 존재하며 메인 서버와 통신. 유저 GUI와 인터페이스를 갖는 클라이언트는 서버와 양방향 통신.

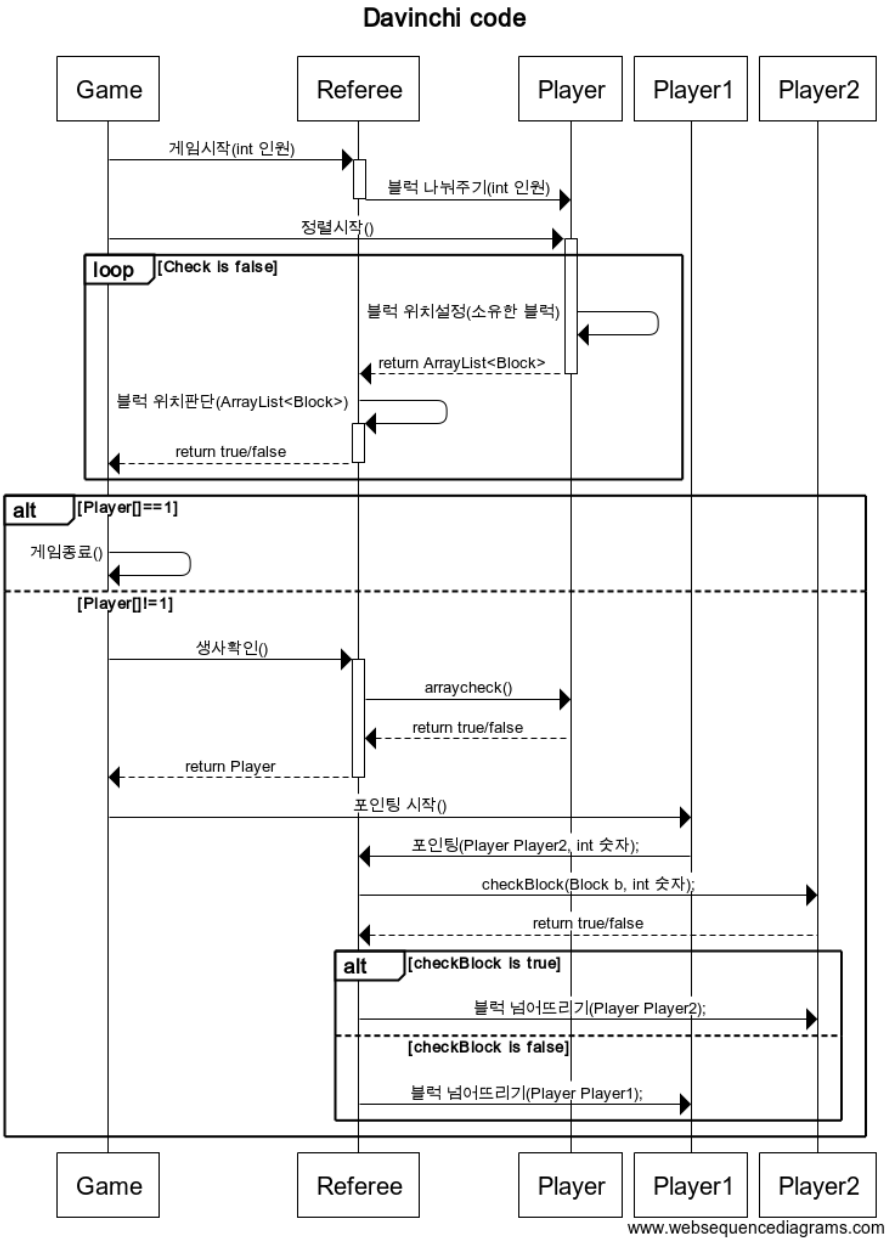
3.1.1.5 서버 : 구현 후



초기에 분석한 구조에 살만 붙혀진 형태. 메인 서버가 방 서브 서버 뿐만 아니라 각 유저들을 위한 수신 서브 쓰레드를 가져 처리. 서버와 클라이언트 사이에서 사용되는 프로토콜을 클래스화 시켜 통일.

3.2 시퀀스 다이어그램

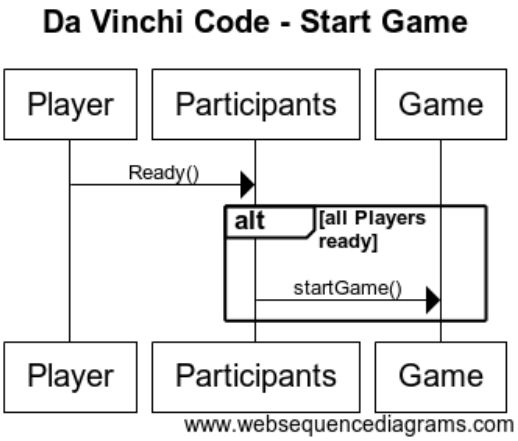
3.2.1 초기 설계



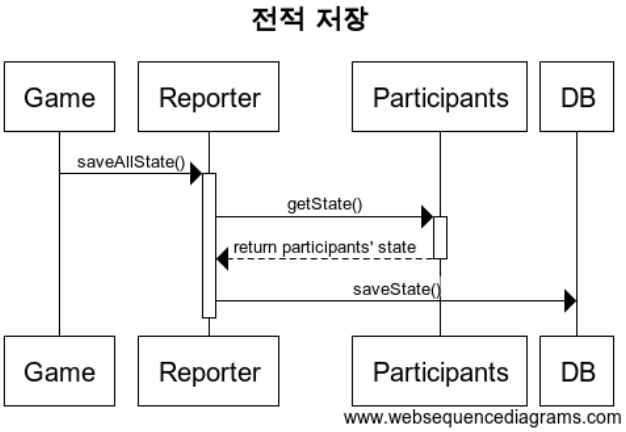
하나의 시퀀스 다이어그램에 모든 기능을 넣으려 하다 보니 세분화가 되지 못하고 뭉뚱그려진 다이어그램이 생성되었다.

3.2.2 중간 점검

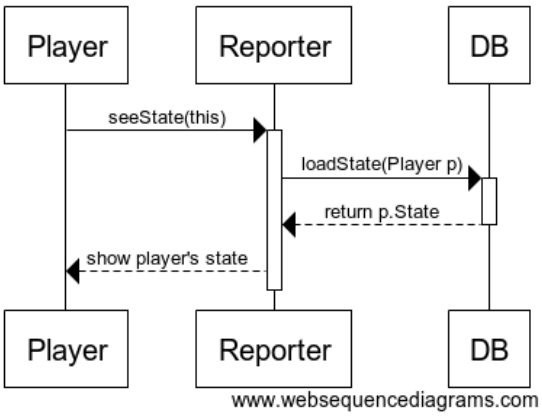
[게임시작]



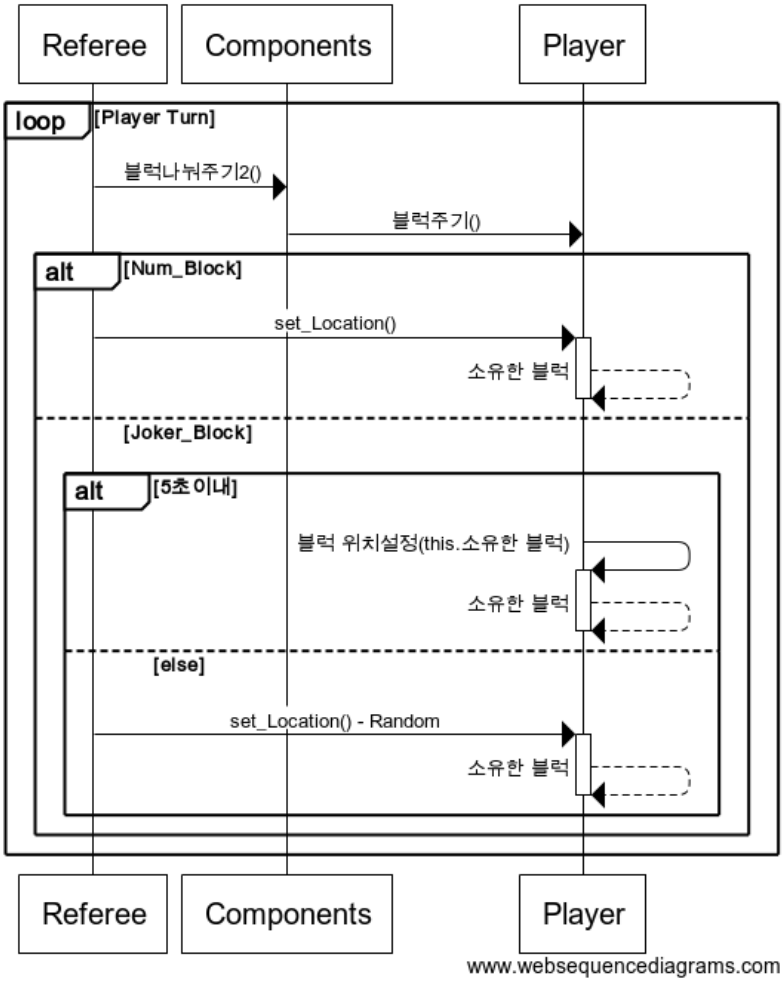
[전적 저장]



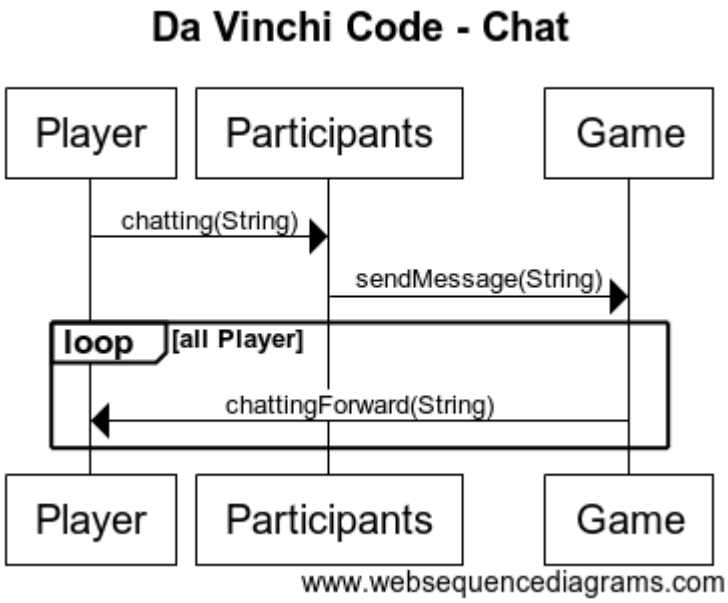
[전적 조회]



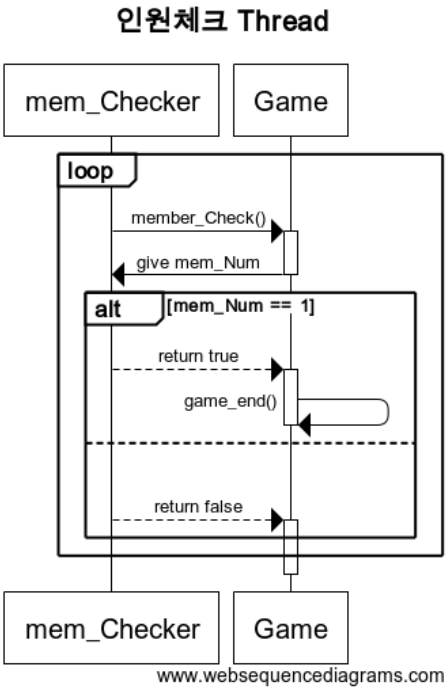
[블록 배분]


Referee
Components
Player
loop
[Player Turn]
블럭나눠주기2()
블럭주기()
alt
[Num_Block]
set_Location()
소유한 블럭
[Joker_Block]
alt
[5초이내]
블럭 위치설정(this.소유한 블럭)
소유한 블럭
[else]
set_Location() - Random
소유한 블럭
Referee
Components
Player
www.websequencediagrams.com

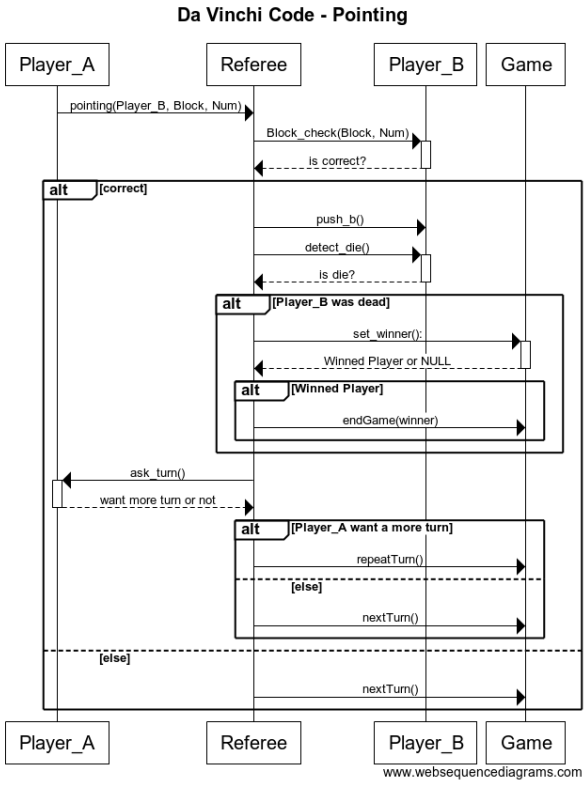
[채팅]


Da Vinchi Code - Chat
Player
Participants
Game
chatting(String)
sendMessage(String)
loop
[all Player]
chattingForward(String)
Player
Participants
Game
www.websequencediagrams.com

[인원 체크]



[블록 지목]



팀 미팅과 교수님의 조언을 바탕으로 시퀀스 다이어그램 또한 최대한 세분화 하였다.

4. 기능 (메소드 분석)

4.1 기능 분리

4.1.1 Gaming

4.1.1.1 Game

게임 프로세스의 모델이 되는 클래스.

Participant 와 Component를 가지고 있으며, 각 클래스들에게 명령을 내리거나 게임의 상태를 관리하는 역할.

4.1.1.2 Player

클라이언트가 게임에 입장했을 때 갖게 되는 게임 플레이어 클래스.

Ready, 카드 블록, 자신의 차례 등 여러 정보를 갖는다.

자신의 블록에 대한 몇 행동을 위한 메소드들이 있다.

4.1.1.3 Referee

게임의 전반적인 컨트롤을 담당하는 심판 클래스.

게임 모델당 하나의 심판이 존재하며, 카드 분배, 위치 설정, 게임 승패 판별 등의 역할을 한다.

4.1.1.4 Block

플레이어들이 게임에 사용하는 카드 블록 클래스.

자신의 색, 번호, 조커 유무, 카드 상태 등 정보를 담고 있다.

4.1.2 Server & Client

4.1.2.1 Server

메인 서버. 이 서버가 열려 있어야 모든 프로세스가 가능하다.

방 서브 서버들을 열어주며, 접속하는 모든 클라이언트들을 받아 각 유저들의 메시지를 수신한다.

수신된 메시지를 자신이 처리하거나 서브 서버에게 명령한다.

4.1.2.2 RoomServer

게임 방의 처리를 담당하는 서브 서버.

유저의 방 출입, 게임 과정 등 과정을 관리한다.

4.1.2.3 Client

서버와 연결하기위한 소켓이 존재하는 클래스.

GUI로부터 받은 정보들로 서버에게 메시지를 송신하고, 늘 서버의 메시지를 받는 역할을 한다.

4.1.2.4 Protocol

서버와 클라이언트 간의 모든 프로토콜을 담아 놓은 클래스.

4.1.3 Database

4.1.3.1 fileDB

MySQL 데이터베이스가 구축되기 전 게임을 위해 사용되던 파일 형식의 데이터 베이스.

4.1.3.2 DAO

MySQL 데이터베이스 클래스.

모든 데이터베이스 서버와의 쿼리 송수신을 담당.

로그인, 회원가입을 위한 유저 정보 입력 또는 조회.

전적 갱신을 위한 데이터 업데이트.

4.1.4 GUI

4.1.4.1 ClientGUI

유저가 게임을 위해 실행하는 클라이언트 GUI 메인 클래스.

하나의 프레임으로 나머지 화면들을 패널로 갖고 있어 상태에 따라 화면을 바꾸어 준다.

4.1.4.2 LoginGUI_p

로그인 화면을 위한 패널 클래스.

아이디와 비밀번호를 받아 로그인이나 회원가입을 진행한다. (Client에게 명령한다.)

4.1.4.3 RoomListGUI_p

입장하기위한 방 정보를 갖고 있는 패널 클래스.

현재 방들의 상태를 보여주며, 클릭하면 방에 입장할 수 있다.

4.1.4.4 GameGUI_p

게임 방에 들어갔을 때, 내부 정보를 표시하는 패널 클래스.

게임을 위한 정보들, 채팅 내용을 전송한다.

서버로부터 받은 게임 정보, 채팅 내용을 표시한다.

4.1.5 TTS

4.1.5.1 Clovatts

Naver API를 이용하여 한글 TTS를 제공해주는 클래스.

기존 API를 변형하여 파일로 저장하지 않고 Data stream을 그대로 읽어주는 기능.

4.2 구현 과정

4.2.1 게임 구조 분석 및 설계

4.2.1.1 요구사항 정밀 분석

4.2.1.2 Class Diagram 설계

4.2.1.3 Sequence Diagram 설계

4.2.1.4 Test code 제작

4.2.1.5 Test

4.2.1.6 완전한 설계까지 이 과정을 반복(Version 4 까지)

4.2.2 게임 및 GUI 상태 설계

4.2.2.1 게임 진행에 있어 필요한 게임 상태 분석.

4.2.2.2 게임 진행에 있어 필요한 화면 상태 분석.

4.2.2.3 목업을 활용한 GUI 초안 제작.

4.2.3 Game, GUI, Server & Client (MVC) 각자 분담 구현

4.2.3.1 Console 진행이 가능한 gaming 모델 구현

4.2.3.2 각 상태를 위한 GUI 제작.

4.2.3.3 탄탄한 기반 Server & Client 제작

4.2.4 Server & Client 를 이용한 Game 과 GUI의 결합

4.2.4.1 모든 네트워크 진행 상황 분석.

4.2.4.2 필요한 신호 분석 및 프로토콜화.

4.2.4.3 Client내부에 Game을 주고, GUI와 Client를 연동하여 모든 과정 연결.

4.2.5 DB 구축

4.2.5.1 MySQL과 JDBC를 세팅

4.2.5.2 위 환경을 활용해 기존 파일형태 DB를 MySQL로 대체

4.2.6 TTS 적용

4.2.6.1 Naver API 구매

4.2.6.2 API 코드를 분석, 변환해 mp3 파일을 java내에서 stream으로 변환하여 바로 출력.

4.2.7 Brutal test 및 Debug

4.2.7.1 기존 분석 단계에서 찾지 못한 예외상황이 있는지 계속해서 Brutal test

4.2.7.2 발견된 사항이 있다면 확인 후 수정.

4.2.7.3 위 반복을 반복.

4.3 제한 점

4.3.1 Gaming

Model 역할을 하며 명령을 받기에 수동적이어야 하나 초기에 이러한 분석이 부족하여 초기 설계와 달라진 부분이 존재.

GUI와의 한계로 유저가 가질 수 있는 카드 수를 12개로 제한

조커 카드의 위치를 정해주는 것이 아닌 랜덤으로 제한.

일부 경우, 이중 참조가 존재.

4.3.2 Server & Client

미리 정해 둔 프로토콜 이외의 정보는 처리가 불가.

Byte 통신이 아닌 String(UTF)로 통신하여 불필요하게 데이터가 큼.

많은 테스트를 통해 고쳤으나, 모든 예외상황을 처리했다고 보장되지 않음.

4.3.3 DB

DB 정보 중 패스워드를 SHA256 과 같은 방식으로 보호하려 했으나 시간 부족으로 하지 않는 것으로 제한.

웹이 아닌 서버와 같은 로컬환경으로 제한.

4.3.4 GUI

최대한 많이 카드를 나열하기 위해 Grid layout을 활용하여 무조건 두줄로 나열.

13장 이상의 카드를 표시하는데 어려움, 따라서 12개를 상한으로 제한.

버튼 신호를 줄 때 팝업이 아닌 하단 콤보 박스와 라디오 박스로 제한.

조커 카드의 랜덤 인덱스 값을 GUI로 받는 점이 애매하여 랜덤으로 처리.

4.4 구현 결과

게임 특성상 GUI 화면 위주로 담았습니다.

4.4.1 Gaming

Class Diagram, Sequence Diagram을 따르며 문제없이 구현되었습니다.

4.4.2 Server & Client

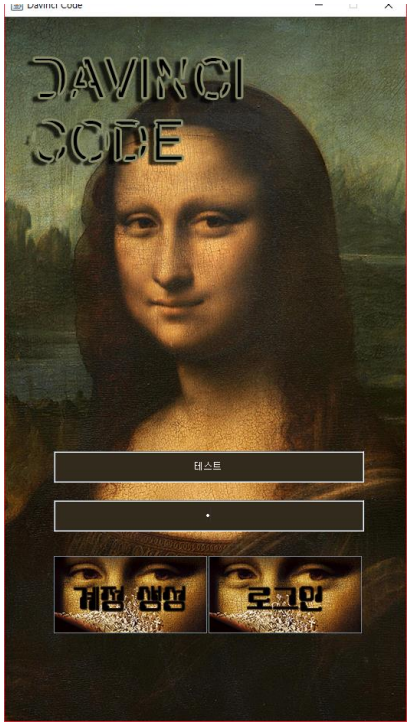
하나의 소켓으로 통신하는 구조로 원활하게 통신이 이루어졌습니다.

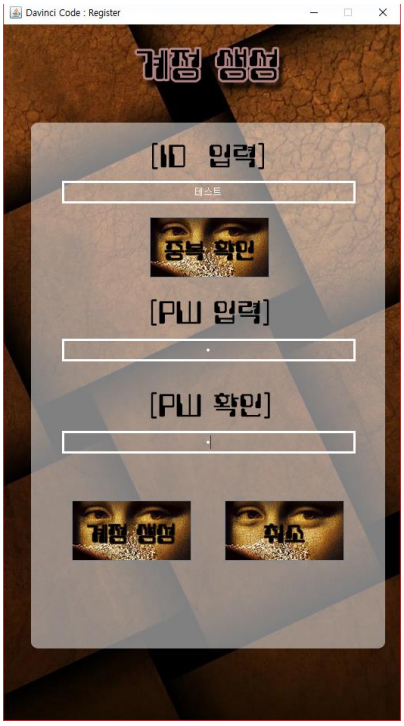
4.4.3 DB

초기 개발에서 쭉 파일 구조를 이용한 후, DB 강의 이후 JDBC를 이용해 적용하였습니다. 적용 결과 잘 나오는 것을 확인 할 수 있었습니다.

4.4.4 GUI

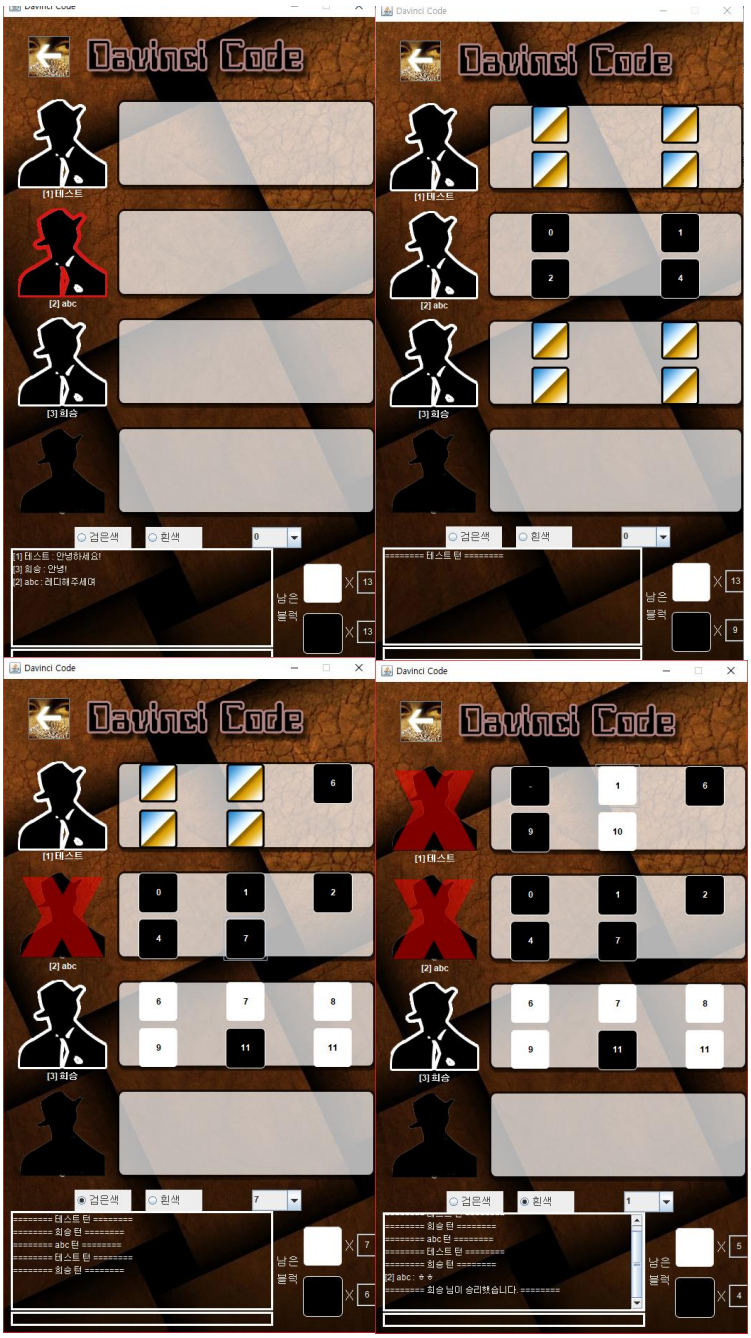
4.4.4.1 로그인 및 회원 가입





4.4.4.2 방 리스트 및 입장

4.4.4.3 방 내부 채팅과 게임 과정



5. 프로젝트 수행평가

5.1 진행 과정

팀플 시작 이후	1주차	2주차	3주차	4주차	5주차
요구사항 분석					
설계 및 UML 작성					
GUI 구성					

서버-클라이언트 구성					
게임 구현					
병합 및 에러 체크					

5.2 역할 분담

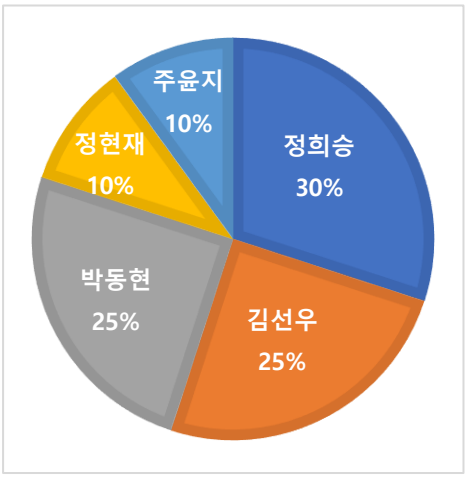
5.2.1 정희승: 게임 구현, 서버-클라이언트 모델 구축, GUI 이벤트 및 프로젝트 병합&총괄

5.2.2 김선우: 서버-클라이언트 모델 구축, TTS(Clova CSS) 적용 및 DB 서버 구축.

5.2.3 박동현: 게임 DFD적 분석 및 요구사항 분석. GUI 구성 및 연결

5.2.4 정현재: 게임 분석, 설계 및 구현

5.2.5 주윤지: 게임 분석, 설계 및 구현



5.3 초기 목표와 결과 비교

- MVC 패턴에 입각한 프로그램 설계 구현

➢ Game, GUI, Server & Client 구조로 MVC 패턴을 매우 잘 따라 구현되었다 생각합니다.

- tts를 이용한 알림 메시지(계정 생성, 로그인, 게임 시작, 게임 종료, 승리선언 등)를 음성으로 출력

➢ 턴 종료 메시지만 음성으로 출력(다른 메시지도 연결하는 것은 비슷하므로 가능했으나 다른 부분을 수정하다 보니 시간 부족)

- 쓰레드, 소켓 통신, DB 등을 강의 내용을 활용하여 적극 활용하고 능력 향상

➢ 각자의 역할의 충실로 각자 맡은 부분을 매우 잘 활용하고 능력이 향상됨. 하지만 정보를 공유하더라도 다른 분야의 경우에는 학습 결과가 부족

5.4 구현 중 가장 힘들었던 기능 & 알고리즘

5.4.1 자꾸만 설계와는 다르게 흘러가는 구현상의 Flow

➔ 설계를 따라가나 필요에 따라 다소 설계를 일부 수정

5.4.2 역할 분담

➔ 협업 툴(Github)을 이용하며, 분담 및 병합이 용이하게 MVC 각 3파트로 분담.

5.4.3 GUI, Server & Client, Game Model, DB 간의 병합

➔ 서버 & 클라이언트 세부 구현 시 프로토콜을 정하여 함께 병합.

5.4.4 예외 처리

➔ 미리 예상한 예외 처리 뿐만 아니라 여러 차례 Brutal test로 발견되는 예외 처리.

5.5 의외로 쉬웠던 기능 & 알고리즘

5.5.1 서버와 클라이언트 간의 통신 프로토콜

➔ 단순 String의 나열로 사전에 정의해두고 사용.

5.5.2 게임내 카드 Sorting

➔ 새로운 카드를 넣는 식이므로, Sorting 알고리즘이 필요하지 않고, 우선순위 큐 형태로 삽입.

5.5.3 게임내 조커 카드에 대한 이벤트

➔ 여러 제한 점 때문에 선회한 내용이지만, 랜덤으로 처리.

5.6 학습 내용

정희승 :

> 처음엔 JVM 구조와 Java 프로그래밍 기초를 복기했습니다. 이후 소프트웨어 설계 방법에서 소프트웨어 공학에서는 C를 가지고 Flow위주 DFD 분석을 수행하며 공부했으나, OOP에서 설계하는 법을 이 강의에서 처음 배워 새롭고 유용했습니다. OOP를 많이 다루지 않았던 입장에서 모두 개념적으로 이해하고 클래스 다이어그램을 짜는 것이 어려워 고생했으나, 점점 해당 개념에 밝아진 것 같습니다. 시퀀스의 경우엔 OOP지만 절차 지향적으로 생각하면서 시퀀스를 짜면 더 잘 구상된다고 느껴 진행했었습니다. 또 데이터베이스 수업을 못 듣고 졸업해서 연이 없다고 생각했으나, 이 수업에서 MySQL을 다루고 Java와 연동까지 하게 되어 익숙하지 않았지만, SQL문법을 공부하고 DB 구조를 익히면서 유용한 정보가 되었습니다.

김선우 :

> 기업에서 제공하는 여러 클라우드 플랫폼에 대해 공부하고, Clova CSS를 프로젝트에 적용했다. 이를 통해서 mary tts 외의 tts 사용법 및 자바 내 음원 파일 기능을 제공하는 Java Zoom Player을 공부했다. 또한 프로젝트에 필요한 오픈소스 라이브러리를 적용할 수 있는 능력이 향상되었다. MySQL에서 서버를 생성하고 DB를 만들어 관리해봄으로써 cmd 뿐만 아니라 autoset, MySQL workbench 활용법을 복습했다. JDBC api를 사용해서 자바 내에서 DB에 접근하여 조작해보고, 삽입, 삭제, 업데이트 등의 기본적인 쿼리문을 익혔다.

박동현 :

> 0. 수업시간에 배운 내용:
> - 다양한 컴포넌트의 종류
> - 레이아웃 종류에 따라 달라지는 컴포넌트의 배치 형식
> - 이미지 불러와서 컴포넌트에 쓰기
> - actionListener 다양한 적용법
> 등을 배웠습니다.

> 1. 프로젝트를 진행하며 공부한 내용, 알게된 것
> - 컴포넌트끼리의 결합으로 여러 기능을 가진 컴포넌트를 생성할 수 있음을 알게 되었습니다. (예) JScrollPane과 JTextArea를 합쳐 스크롤 가능한 TextArea 생성
> - JFrame 뿐만 아니라 JPanel, JButton등 여러 가지 컴포넌트들을 상속을 통해 클래스로 만들 수 있게 되었습니다.
> - JFrame 클래스에 여러 패널을 선언해 전환할 수 있게 되었습니다.
> - 버튼이나 텍스트필드를 투명하게 만드는(setOpaque) 메소드, 프레임을 새로 생성할 때 어디에 위치 시키는가에 대한(setLocationRelativeTo) 메소드 등 각 컴포넌트의 다양한 메소드를 공부하고 사용해보았습니다.

3. 프로젝트를 진행하며 해결하지 못한 것, 모르는 것
- JLabel이나 JButton에 이미지를 씌우고 그것을 패널에 붙여 GUI를 구성했는데, 이상하게도 JFrame의 Layout을 absolute로 설정하면 화면의 일부가 잘려나갔습니다. 그것을 수정하기 위해 JFrame의 Layout을 BorderLayout으로 설정하고 패널들을 JFrame의 Center에 add한 뒤 pack() 메소드를 사용하여 올바른 정렬을 유도했으나 무엇 때문인지 의도대로 구현되는 것도 있고 그러지 않은 것도 있어서, 무엇이 원인인지 알아내지 못하였습니다. 다만 필수적인 GUI패널들에는 다 적용이 되어서 구동에는 무리가 없었습니다.
- 프레임 크기에 따른 컴포넌트들의 동적인 크기 변화를 다양하게 시도해보았으나 생각보다 쉽지 않아 완전히 숙달하지 못하였습니다.

4. 공부해야 할 것
- 컴포넌트에 삽입할 이미지가 컴포넌트의 크기에 자동으로 맞춰지게 하는 방법
- 각 Layout의 세부 설정법
- 아직 사용해보지 못한 컴포넌트들(JToggleButton, JComboBox ...)의 기능과 활용법을 공부해야 할 것 같습니다.

정현재 :

1. Multi thread
Multi thread는 하나의 프로세스가 두 가지 이상의 작업을 처리할 수 있도록 하는 것으로, 메인 스레드와 작업 스레드를 만들어 실행합니다.
수업시간에 배운 스레드 처리 우선순위와 동기화 메소드 및 상태 제어 방법들을 기반으로, 주 스레드의 작업을 돕는 데몬 스레드를 만들어봤습니다. 또 스레드 그룹을 형성하여 일괄처리 하는 법도 연습했습니다.
이를 이용하여 프로젝트에 필요한 멀티 채팅 구현을 시도했습니다. (결국 실패했습니다..) 앞으로 구현을 성공시키고 데이터베이스에서 멀티 스레드를 구현하는 연습을 해볼 예정입니다.

2. lamdba expressions

lamdba expressions은 익명함수를 생성하기 위한 식으로, 인터페이스 작성에서 효율적으로 이용할 수 있습니다. 사실 multi thread 구현에 실패하고 병렬 처리를 위한 다른 방법을 찾다가 접하게 되었습니다.

함수적 인터페이스와 람다식 기본 문법을 알고 stream을 구현하여 내부반복자를 이용한 병렬처리를 연습하였습니다. 컬렉션, 배열에서 스트림을 얻어보고 출력하기도 하였습니다.

이후 프로젝트의 인터페이스들을 간결하게 정리하였습니다. 그리고 프로젝트에 필요한 병렬처리들을 stream으로 처리하려 했지만 오히려 더 복잡해지기에 포기했습니다. 앞으로 stream api를 이용한 데이터 grouping을 알아볼 예정입니다.

3. prolog

prolog는 논리형 프로그래밍 언어로, 6가지의 데이터형을 가집니다. 사실과 규칙을 제공하여 데이터베이스를 만들고, 이에 쿼리를 함으로써 프로그램을 수행합니다.

위 언어를 이용하기 위해 swi-prolog을 설치하고 eclipse와 연동하였습니다. 그리고 이 문법의 기본 문법을 익히고 퀵 정렬, 삽입 정렬 알고리즘을 연습했습니다. 그리고 패턴 인식과 백트래킹 구조를 이용하여 동물들을 데이터베이스에 넣고, 몇가지 조건을 입력하면 해당하는 동물을 찾아주는 동물 식별 프로그램을 작성해보았습니다.

이와 같은 연습들을 통해 prolog언어에 익숙해진 후, 프로젝트에 적용했습니다. 그리고 앞으로 이 언어를 이용하여 인공지능 및 딥러닝 공부를 해볼 예정입니다.

주윤지 :

-MVC 디자인 패턴

MVC패턴은 Model-Controller-View패턴의 줄임말이다. Model은 사용자들에게는 보이지 않는 코드가 돌아가는 핵심 기능과 데이터를 가지고 있다. View는 모델로부터 정보를 얻어와 사용자들에게 보여주는 역할을 한다. Controller는 Model에 명령을 보내서 모델의 상태를 변경하고 그에 맞춰 View를 바꿔주는 중간역할을 한다.

> -Socket을 이용한 서버구축
> Java에서 socket 라이브러리를 이용해서 서버를 구축하는 것에 대해 공부하였다. 소켓은 TCP/IP 통신을 하기 위해 사용하는 것인데, 각 포트를 사용하여 통신을 수행하는 도구라고 할 수 있다. 각 프로그램에 포트를 세팅한 후 소켓으로 데이터를 주고받으면서 통신을 하게 된다. 포트는 출입구의 역할, 소켓은 출입구를 통한 데이터 송수신 매체라고 할 수 있다. 소켓은 서버 소켓과 클라이언트 소켓으로 나뉘어져 있는데, 서버 소켓은 클라이언트 소켓의 연결 요청을 대기하고 연결이 되면 클라이언트 소켓을 생성해 통신을 가능하게 해준다. 클라이언트 소켓은 실제로 데이터 송수신이 일어나는 소켓이며 대기하는 것 없이 바로 사용 가능하다.

5.7 잘된 점

5.7.1 UML(Class Diagram, Sequence Diagram)을 기반으로 한 게임 코드 구현

5.7.2 GUI, DB, Game, Socket들의 병합

➔ 이번 프로젝트에서 학습한 내용으로 컴퓨터 네트워크, 네트워크 프로그래밍에서 더욱 자세히 배우고 활용할 수 있을 것이다.

5.7.3 MVC 패턴 유지

5.7.4 성공적인 API 이용

➔ 오픈소스 수업을 들으면 더 많은 오픈 API를 사용 또는 응용이 가능할 것이다.

5.7.5 적절한 역할 분담.

5.7.6 새로운 응용 학습 및 사용

5.8 아쉬운 점

5.8.1 인터페이스 구조 활용의 미흡

5.8.2 구현에 있어 다른 해결법이 보이지 않아 사용하게 된 이중 참조

5.8.3 많은 예외 처리에도 잡아내지 못한 예외 상황(특히 서버 & 클라이언트)

➔ 소프트웨어 공학 개론, 검증 수업을 통해 예외 상황 파악 및 테스트 기법들을 더 자세히 파악하면 다 나은 개발이 가능해질 것이다.

5.8.4 분담이후 자기 역할 외 부분에 대한 공유 및 파악 미흡

5.9 개인후기

정희승 :

> 2학년 이후 오랜만에 사용하는 자바로 리마인드를 시킬 수 있는 기회라 생각하여 수강하였는데 생각보다 많은 설계론과 구현 방법론, 예외 처리, 소켓, DB, API 응용단까지 학습 및 응용하여 매우 유익한 수업이 되었습니다. 수업을 수강하며 교수님의 진행 속도와 그 이상을 하려 노력하고 이루며 자만심도 생겼었으나, 중간고사 이후 더 확실한 Flow 분석, JVM 구조 파악이 필요하다는 것을 깨닫고 아직 한참 모자라다는 것을 깨닫게 되었습니다. 프로젝트를 수행함에 있어서는 팀장을 하기엔 미숙해서 초기 아이디어 선정, 역할 분담부터 난항이었습니다. 그럼에도 팀원들이 좋은 아이디어를 내어주고 살을 붙혀 주어 좋은 설계가 나온 것 같습니다. 이후 구현에 있어서는 한 개인의 프로젝트가 되지 않도록 각자의 역할을 분담하고 주기적으로 만나며 잘못된 방향으로 나가지 않도록 하고 잘 모르는 부분을 알려주고 공유하는 식으로 진행하였습니다. 모두 공평하게 분담하고 수행했다고 생각할 수는 없지만 나름 다같이 학습하고 발전하면서 동시에 프로젝트가 완성되어가는 모습을 보며, 뿌듯했습니다. 이후 남은 학부생으로의 한학기, 대학원 생활, 회사 생활에 있어 만날 또 다른 팀 프로젝트에서 여기서 학습한 경험을 바탕으로 리더든 팀원이든 올바른 설계와 분업을 할 수 있을 것 같습니다. 유익한 강의 감사했습니다.

김선우 :

> 수업 시간에 배운 내용들 중 아직 익숙하지 않은 부분들이 많았는데 프로젝트를 진행하면서 자연스럽게 복습도 되고, 활용하는 법도 배웠다. 소켓 프로그래밍의 경우, 특히 부족한 부분이 많고 공부할 부분이 많아서 추가적으로 공부해야 하겠다는 생각이 들었다. tts 같은 경우 오픈소스 라이브러리를 사용하는 방법과 스레드 구현을 모두 복습하는 계기가 되었고, DB관리를 맡으면서 공부를 MySQL 또한 공부해보고 싶다는 생각이 들었다. 프로젝트 주제 설정 시 의욕이 앞서서 역량에 벗어나는 주제를 제안했는데 무사히 프로젝트를 완성하게 되어서 너무 뿌듯했다. 아직 부족한 부분이 많지만 이번 프로젝트를 완성함으로써 학습 의지에 불을 지피는 계기가 되었다.

박동현 :

> 이번 프로젝트를 진행하며 GUI를 중점적으로 맡아서 그런지 GUI에 대한 지식을 많이 습득하게 되었습니다. GUI를 구성하는 것 자체는 크게 어려운 일은 아니었지만, 레이아웃을 앱솔루트로 지정하다 보니 컴포넌트의 위치를 하나하나 다 설정해주는 등 손이 많이 가기도 했습니다. 그래도 팀프로젝트를 진행하면서 다양한 시도를 하였고 모르던 메소드도 많이 사용해보게 되어 그 결과 제가 분담받게 된 분야에 대해서 꽤 숙달된 것 같아 기쁘게 생각하고 있습니다. 하지만 역시 프로젝트에서 가장 큰 장애물은 다른 코드들과 GUI를 연결시키는 것이었는데, 서버, DB, GUI, 실제 게임코드 등 여러 구성물들이 조합되어 하나의 결과물로 탄생하는 것에는 개발자들의 노력이 많이 필요하다는 사실을 깨달았습니다. 그리고 설계부터 시작하여 구현까지 시도한 것은 대학교에 입학하여 처음이었는데 마냥 순조롭게 프로젝트가 진행되지는 않았지만 마무리 짓게 되어서 뿌듯한 마음도 있고, 이 경험을 앞으로 진행하게 될 프로젝트에 적용하여 설계부터 제대로 할 수 있는 자원이 되었으면 좋겠습니다.

정현재 :

> 이 프로젝트를 진행하며 코딩을 한 후 MVC pattern에 끼워맞추는 것이 아닌, model, view, controller를 먼저 구성한 후 코드를 작성해 볼 수 있었습니다. 또 프로그램에 사용되는 알고리즘과 구성 요소들의 관계를 고려해 클래스 다이어그램을 작성해 볼 수 있었습니다. 수업시간에 이론을 배우고, 교수님을 따라 그리는 것에서 한걸음 나아가 스스로 생각하여 그리려하니 잘 그려지지 않았습니다. 팀 회의 전에 여러 번 고치고, 처음부터 새로 그려보기도 했음에도 제 클래스 다이어그램보다 더 좋은 다른 팀원의 그림을 보며 많은 것을 배울 수 있었던 것 같습니다.

> 팀에서 분업한 결과, 저는 플레이어가 준비를 마치고, 심판이 카드를 나누어주고, 규칙에 따라 플레이어가 움직이는 게임 자체를 구현하게 되었습니다. 이 과정에서 먼저 Controller와 View를 구분지어 코드를 작성하는 것에 초점을 맞추려 노력했습니다. 예를 들어 블록 클래스와 블록렌더링 클래스를 따로 두어 블록 클래스에는 블록이 가지는 변수와 메소드를, 블록렌더링 클래스에는 블록에 해당하는 gui와 actionlistener들을 구현하였습니다. 이렇게 구현하니 오류나 개선할 점을 수정할 때 매우 편리했고, 역할분담을 하여 처리하니 여러 작업을 동시에 처리하도록 코드를 작성하기도 편했습니다. 다음으로 상대의 블록을 선택하고, 해당 숫자를 예측하는 과정을 pointing이라 하기로 했는데, 이를 구현하는데 어려움이 있었습니다. 블록이 가진 정보들 중 원하는 정보만 가져와서 입력된 정보와 비교해야 했는데, 클래스 상속과 참조만으로 구현하기에는 한계가 있었습니다. 왜냐하면 클래스 간 관계가 is가 아닌 has 관계에 있는 것들이 많았기도 했고, 단순히 참조 처리를 하면 블록들이 하나의 arraylist를 공유하는 등의 문제가 발생했기 때문입니다. 그래서 이를 해결하고자 prolog라는 논리형 프로그래밍 언어를 java eclipse에 적용하여 데이터베이스 쿼리가 용이하도록 하였습니다. 정보를 요청하고 불러오는 과정이 간단해지자 pointing 역시 간단한 코드로 작성할 수 있었습니다. 최종 결과물은 이를 이용하지 않은, 더 좋은 코드가 제출되겠지만 개인적으로 새로운 언어를 공부하고 적용해보는 유익한 경험이었다고 생각합니다. 마지막으로 제네릭 프로그래밍을 적극적으로 이용하려 노력했습니다. 게임에서 플레이어가 다양한 이벤트들을 이용하게 되는데, 이 이벤트들을 불러오는 중간과정의 클래스를 만들어 참조 또는 중복되는 코드를 줄이고자 하였습니다. 이 때 이벤트마다 불러오는 방법, 반환하는 형식 등이 다르기 때문에 제네릭을 이용해야 했습니다. 이처럼 수업시간에 배운 내용들을 최대한 응용하고, 필요하다면 새로운 내용을 공부하기도 하며 프로젝트에 참여하였습니다.

> 하지만 팀플 과정에서 제 자신을 드러내는 것에 대한 두려움이 커서 협동 과정에서 적극적이지 못한 점이 있었고, 소켓 구현에 대한 이해가 부족하여 힘들었습니다. 팀에서 구현하고자 하는 게임에서는, 1:1 채팅이 아닌 멀티 채팅이 필요했습니다. 이에 따라 멀티 스레드 프로그래밍이 필요했고, 이를 응용하는데 어려움이 있었습니다. 여러 스레드가 어떤 변수나 클래스 객체에 접근할 때 제 의도대로 동작하지 않았고, 객체를 공유할 때 문법적인 오류가 없음에도 발생하는 에러에 대응하기가 힘들었습니다.

> 따라서 앞으로 소켓 구현에 대한 공부를 더 해보고, 팀원들의 도움에서 벗어나 스스로 구현할 수 있는 능력을 기르고 싶습니다. 또한 스레드를 응용하는데 필요한 이론들을 정확하게 이해하고 프로그래밍 해보고 싶습니다.

주윤지 :

> 이번 프로젝트에서 클래스 구현을 맡았었습니다.
> 클래스 설계의 경우에는 팀원 다 같이 만나서 하였지만, 구현을 하다 보니 관계도도 잘 맞지 않고 불필요한 메소드도 많고 오히려 필요한 메소드는 없는 것을 보고 클래스 설계가 잘못되었다는 것을 깨닫고 3번정도 다시 설계를 했습니다. 하지만, 결국에는 끝까지 비슷한 부분에서 알고리즘이 잘 맞지 않아 다른 팀원분들이 다시 설계를 해주었습니다.

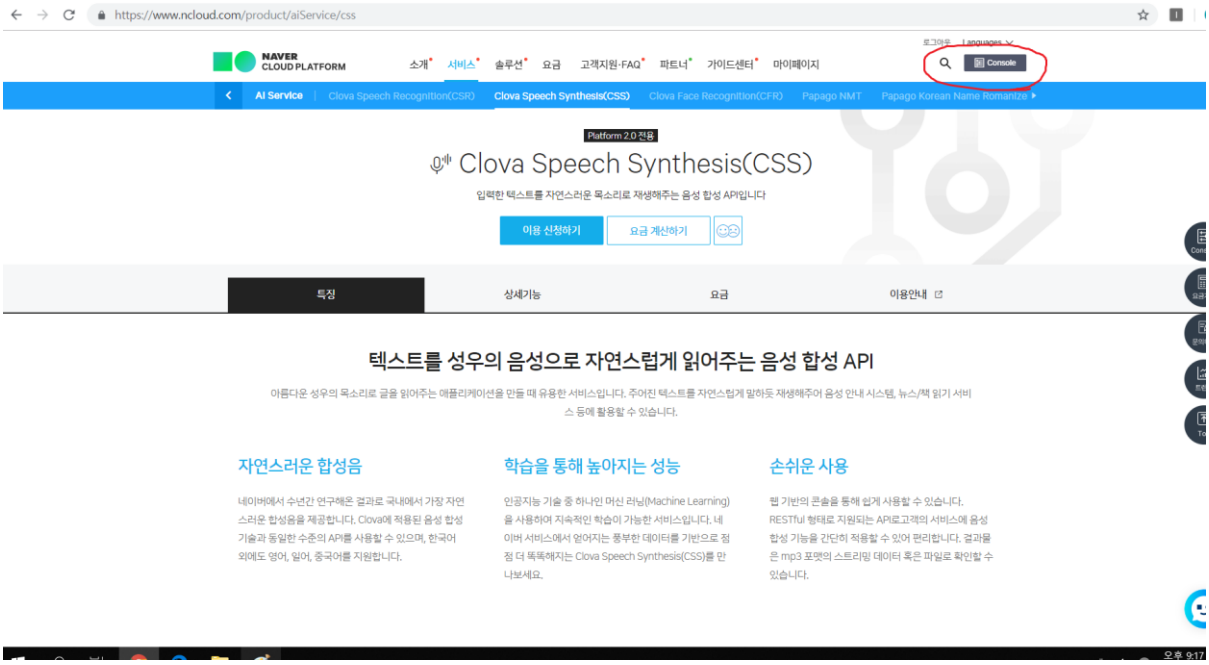
> 또한, 클래스 구현의 경우에는, 다른 클래스들을 구현할 때도 많이 어려웠지만 특히 Referee와 Player의 클래스를 설계할 때 이 게임의 핵심인 Pointing메소드를 구현하는 것이 제일 어려웠습니다. 예외처리를 해야 할 것도 많았고, 여러 클래스들의 변수나 메소드와 연관 지어서 만들어야 해서 어려웠습니다.

> 여러모로 많은 부분에서 부족함을 느꼈었고, 팀원들과 원활하게 의사소통을 못해서 팀에게 많은 도움이 되지 못한 것 같습니다. 다음에 팀프로젝트를 할 때는 구현능력도 키우고, 알고리즘도 잘 이해해서 적극적으로 팀에게 도움이 되도록 노력하겠습니다.

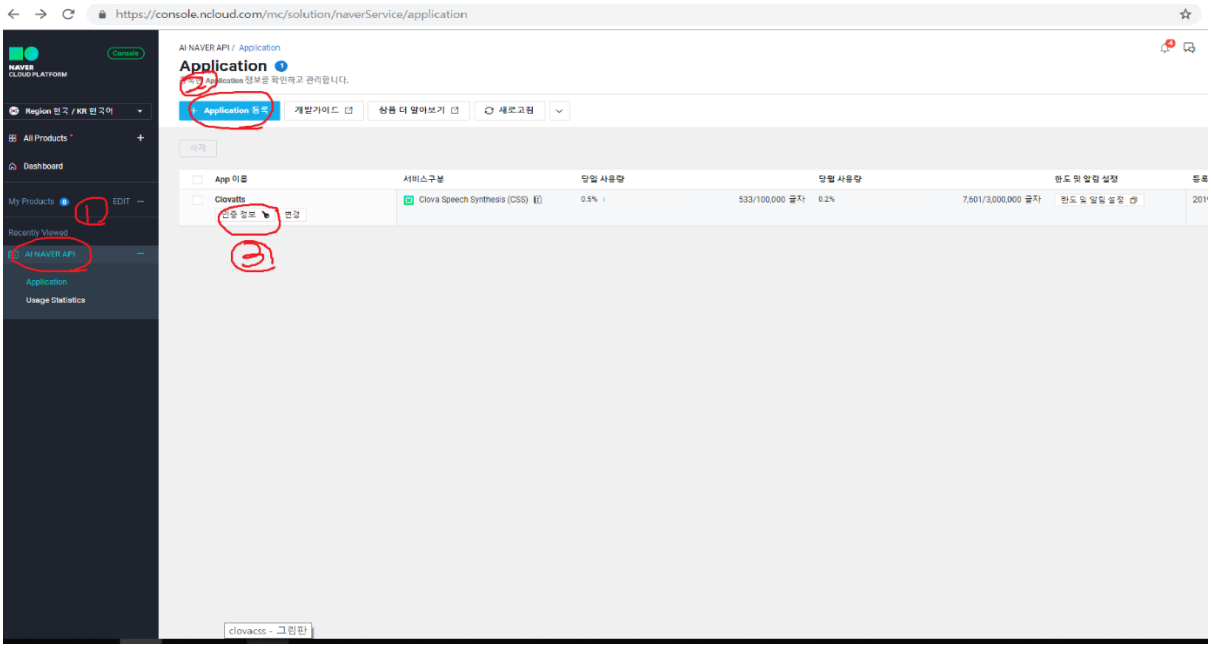
6. 별첨

6.1 네이버 TTS API(Clava css) 사용법

6.1.1 네이버 n클라우드 플랫폼 에 접속 후, 로그인한다.



6.1.2 우측 상단의 console 창을 누르면 다음과 같은 창이 뜨는데 application을 누른 후, 사용할 application을 등록한다. 등록이 되면 밑에 app이 생기는데 인증 정보 창에 있는 id와 시크릿 값을 통해 app url에 요청을 보낼 수 있다.



6.1.3 CONSOLE 창이 아닌 'API 참조서' 창으로 접속하면 네이버n클라우드 플랫폼에서 지원하는 모든 api에 대한 사용설명을 볼 수 있다. 한글로 자세히 설명이 되어 있어서 사용하기 정말 편하다.