|  |  |  |  |
| --- | --- | --- | --- |
| 프로젝트명 | 인디언포커 개발 | 버전 | 1.0 |
| 문서명 | 소프트웨어 최종 보고서 | 작성일 | 11.30 |
| 단계 | 테스팅 | 작성자 | 소공프렌즈 |

**소프트웨어 최종 보고서**

**-소공프렌즈-**

-목차-

내용

[**1.** **개요** 4](#_Toc468333682)

[① 개발시스템의 목표 4](#_Toc468333683)

[**2.** **기능요구사항** 4](#_Toc468333684)

[① 요구사항 분석 4](#_Toc468333685)

[② 시나리오 5](#_Toc468333686)

[**3.** **유즈케이스** 7](#_Toc468333687)

[① 유즈케이스 다이아그램 7](#_Toc468333688)

[② 이벤트 흐름 7](#_Toc468333689)

[**4.** **도메인모델** 12](#_Toc468333690)

[① 도메인 모델링 12](#_Toc468333691)

[**5.** **유즈케이스 실현** 12](#_Toc468333692)

[① 시퀀스 다이어그램 12](#_Toc468333693)

[**6.** **분석모델** 22](#_Toc468333694)

[**7.** **아키텍처 기술서** 23](#_Toc468333695)

[**8.** **유즈케이스 실현** 24](#_Toc468333696)

[① UI design 24](#_Toc468333697)

[**9.** **DB design** 26](#_Toc468333698)

[**10.** **디테일한 Class Diagram** 27](#_Toc468333699)

[**11.** **Component Diagram** 28](#_Toc468333700)

[**12.** **Client** 29](#_Toc468333701)

[① Android Network 29](#_Toc468333702)

[② Socket 통신 29](#_Toc468333703)

[③ Thin Client 29](#_Toc468333704)

[④ 메시지 처리 30](#_Toc468333705)

[**13.** **Server** 31](#_Toc468333706)

[① 13.1 OCSF 프레임 워크를 사용 31](#_Toc468333707)

[② 씬 클라이언트 구조 31](#_Toc468333708)

[③ 메시지 처리 31](#_Toc468333709)

[④ 게임의 핵심 로직을 담당 34](#_Toc468333710)

[**14.** **Class Diagram** 35](#_Toc468333711)

[① Client 35](#_Toc468333712)

[② Server 36](#_Toc468333713)

[**15.** **Deployment Diagram** 37](#_Toc468333714)

[**16.** **Component Diagram** 38](#_Toc468333715)

[**17.** **테스트 케이스** 39](#_Toc468333716)

# **개요**

## 개발시스템의 목표

인디언 포커 게임을 1인 또는 2인이 하며 랭킹을 볼 수 있는 게임 어플리케이션을 개발한다.

# **기능요구사항**

## 요구사항 분석

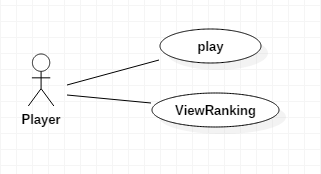
|  |
| --- |
| 요구사항 내용 |
| 1~10까지의 숫자가 그려진 카드 2세트를 가지고 게임을 만들려고 한다. |
| 게임은 카드, 점수, 한 명 혹은 두 명의 Player로 진행된다. |
| 각 Player는 1~10까지의 숫자가 적힌 카드 1세트를 가지고 게임에 참여한다. |
| 각 Player는 기본점수 200점을 가지고 게임을 시작한다. |
| 각 Player는 상대방이 뽑은 한 장의 카드를 볼 수 있다. |
| 각 Player는 자신이 뽑은 카드가 무엇인지 확인할 수 없다. |
| 게임은 카드의 숫자가 높은 Player가 승리하게 된다. |
| 각 Player는 자신의 카드를 보지 못한 상황에서 상대의 카드를 보고 점수를  걸어야 한다. |
| 기본 배당 점수는 10점부터 시작이며 바로 직전에 걸었던 점수보다  적게 걸 수는 없다. |
| 각 Player는 자신이 뽑은 카드가 상대방의 카드보다 높은 점수라고 생각한다면  배당점수를 늘린다. |
| 두 Player간 배당 점수가 같을 경우, 한 턴이 종료되며 배당된 모든 점수는  승리자에게 돌아간다. |
| 단, 두 Player가 같은 숫자 카드를 뽑았을 경우, 배당되었던 점수는 다음 턴 기본 배당점수로 넘어간다. |
| 한 Player가 배당을 포기할 경우, 무조건 패배하게 되며 포기시점까지 배당된  모든 점수는 승리자에게 돌아간다. |
| 단, 포기한 Player의 카드가 숫자 10일 경우, 포기한 Player는 100점을 상대에게 주어야 한다. |
| 게임은 총 10회의 turn을 가지며, 10회이내에 한 Player의 점수가 0점이 되거나 10회가 끝나면 게임이 종료된다. |
| 게임을 하는 도중 취소할 수 있어야 하며 이탈한 Player는 패배하게 된다. |
| 게임이 끝나면 각 Player들의 이름과 최종점수가 Rank에 업로드 된다. |

## 시나리오

|  |  |
| --- | --- |
| Actor Actions | System response |
| 새로운 Player A가 게임을 시작한다. |  |
|  | Player A에게 기본점수 200점을 부여한다. |
| Player A가 카드를 뽑는다. |  |
|  | 1~10숫자중 3을 뽑고 상대Player(Player B)에게 카드를 보여준다. |
| 첫 번째 턴에서 Player A가 상대(Player B)의  카드를 보고 10점을 건다. |  |
| 첫 번째 턴에서 Player A가 상대방(Player B) 보다 높은 점수라고 생각하여 20점을 건다. |  |
|  | 배당 점수를 저장하고 상대방(Player B)의 배당을 기다린다. |
| 상대방(Player B)의 배당 점수와 같은 점수를 건다. |  |
|  | 두 Player의 카드 결과를 출력하고 승리자를 판단한다. |
|  | 저장된 배당 점수를 모두 승리자에게 준다. |
| Player A가 카드를 뽑는다. |  |
|  | 1~10숫자중 5를 뽑고 상대(Player B)에게 카드를 보여준다. |
| 두 번째 턴에서 Player A가 상대카드를 보고 10점을 건다. |  |
| 두 번째 턴에서 Player A가 상대방보다 낮은 점수라고 생각하여 배당을 포기한다. |  |
|  | 두 Player의 카드결과를 출력하고 상대(Player B)를 승자로 판단한다. |
|  | 저장된 배당 점수를 모두 승리자에게 준다. |
| …………….중략……………. |  |
| Player A가 카드를 뽑는다. |  |
|  | 1~10숫자중 10을 뽑고 상대(Player B)에게 카드를 보여준다. |
| 열 번째 턴에서 Player A가 상대카드를 보고 10점을 건다. |  |
| 열 번째 턴에서 Player A가 상대방보다 낮은 점수라고 생각하여 배당을 포기한다. |  |
|  | 두 Player의 카드결과를 출력하고 상대(Player B)를 승자로 판단한다. |
| 포기한 Player A의 카드가 10이였으므로 상대(Player B)에게 100점을 추가로 준다. |  |
|  | 저장된 배당 점수와 추가점수를 모두 승리자에게 준다. |
|  | 10회턴이 모두 끝나 게임을 종료한다. |

# **유즈케이스**

## 유즈케이스 다이아그램



## 이벤트 흐름

1. **기본흐름 - viewRanking**

|  |  |
| --- | --- |
| Actor Actions | System response |
| 1.viewRank로 최종점수를 본다 |  |
|  | 2. 모든 플레이어의 점수를 출력한다. |

1. **예외흐름 - viewRanking**

|  |  |
| --- | --- |
| Actor Actions | System response |
| 1. viewRank로 최종점수를 본다 |  |
| 2. 전체 랭킹이 아직 없는 경우이다. |  |
|  | 3.Ranking이 존재하지 않다고 출력한다. |

1. **기본흐름-play**

|  |  |
| --- | --- |
| Actor Actions | System response |
| 1. Player가 게임을 시작한다. |  |
|  | 2. Player에게 기본점수 200점을 부여한다. |
| 3. Player가 카드를 뽑는다. |  |
|  | 4. 1~10랜덤 숫자를 뽑고 상대의 Player에게 카드 숫자를 보여준다. |
| 5. Player가 상대의 카드를 보고 10점부터 점수를 건다. |  |
| 6. 자신의 카드가 상대방 보다 높은 점수라고 생각한다면 배당점수를 늘린다. |  |
|  | 7. 배당 점수를 저장하고 상대방의 배당을 기다린다. |
| 8. 상대방의 배당 점수와 같은 점수를 건 경우이다. 한 턴이 종료된다. |  |
|  | 9. 두 Player의 카드 결과를 출력하고 승리자를 판단한다. |
|  | 10. 저장된 배당 점수를 모두 승리자에게 준다. |
|  | 11. 10회턴이 모두 끝나면 게임을 종료한다. |
|  | 12. Player의 승률을 Rank에 저장한다. |

1. **대체흐름 1 – 두 카드의 숫자가 같은 경우**

|  |  |
| --- | --- |
| Actor Actions | System response |
| 1. Player가 게임을 시작한다. |  |
|  | 2. Player에게 기본점수 200 점을 부여한다. |
| 3. Player가 카드를 뽑는다. |  |
|  | 4. 1~10랜덤 숫자를 뽑고 상대의 Player에게 카드 숫자를 보여준다. |
| 5. Player가 상대의 카드를 보고 10점부터 점수를 건다. |  |
| 6. 자신의 카드가 상대방 보다 높은 점수라고 생각한다면 배당점수를 늘린다. |  |
|  | 7. 배당 점수를 저장하고 상대방의 배당을 기다린다. |
| 8. 상대방의 배당 점수와 같은 점수를 건 경우이다. 한 턴이 종료된다 |  |
|  | 9. 두 Player의 카드 결과를 출력하는데 두 카드의 숫자가 같으므로 무승부를 판단하고 지금까지 배당된 점수를 다음 턴의 기본 배당 점수로 넘겨준다. |
|  | 10. 다음 턴을 실행한다. |

1. **대체흐름 2 - Player가 배당을 포기한 경우**

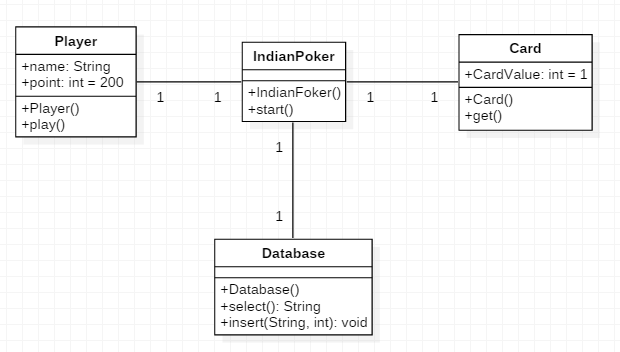
|  |  |
| --- | --- |
| Actor Actions | System response |
| 1. Player가 게임을 시작한다`. |  |
|  | 2. Player에게 기본점수 200 점을 부여한다. |
| 3. Player가 카드를 뽑는다. |  |
|  | 4. 1~10랜덤 숫자를 뽑고 상대의 Player에게 카드 숫자를 보여준다. |
| 5. Player가 상대의 카드를 보고 10점부터 점수를 건다. |  |
| 6. 자신의 카드가 상대방 보다 낮은 점수라고 생각하여 배당을 포기한다. |  |
| 7. 포기한 Player의 카드 결과가 10일 경우이다. 상대Player에게 100점을 추가로 준다. |  |
|  | 8. 두 Player의 카드 결과를 출력하고 상대Player를 승자로 판단한다. |
|  | 9. 저장된 배당 점수를 모두 승리자에게 준다. |
|  | 10. 10회턴이 모두 끝나면 게임을 종료한다. |
|  | 11. Player의 승률을 Rank에 저장한다. |

1. **예외흐름**

|  |  |
| --- | --- |
| Actor Actions | System response |
| 1. Player가 게임을 시작한다. |  |
|  | 2. Player의 점수를 가져온다. |
| 3. Player가 카드를 뽑는다. |  |
|  | 4. 1~10랜덤 숫자를 뽑고 Player에게 상대의 카드를 보여준다. |
| 5. Player가 가진 점수보다 더 많은 점수를 건다. |  |
|  | 6. Player에게 점수가 초과되었다고 알리고 다시 입력하게 된다. |

# **도메인모델**

## 도메인 모델링

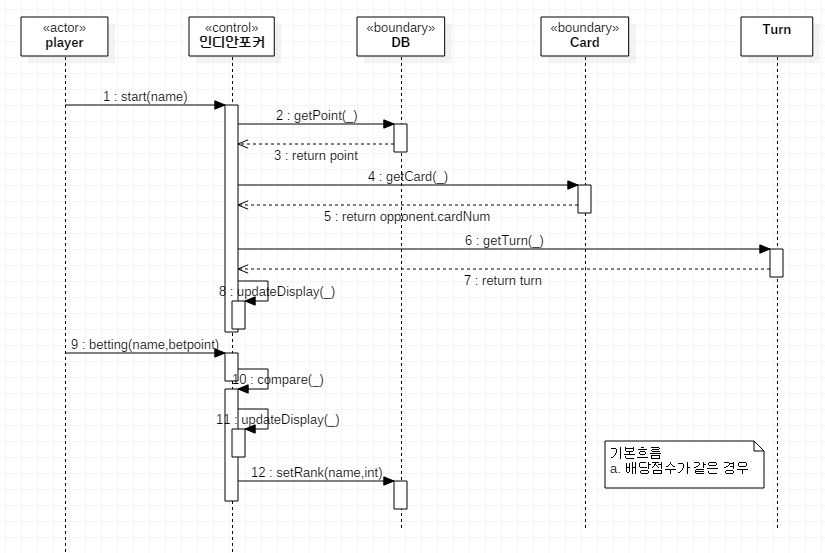


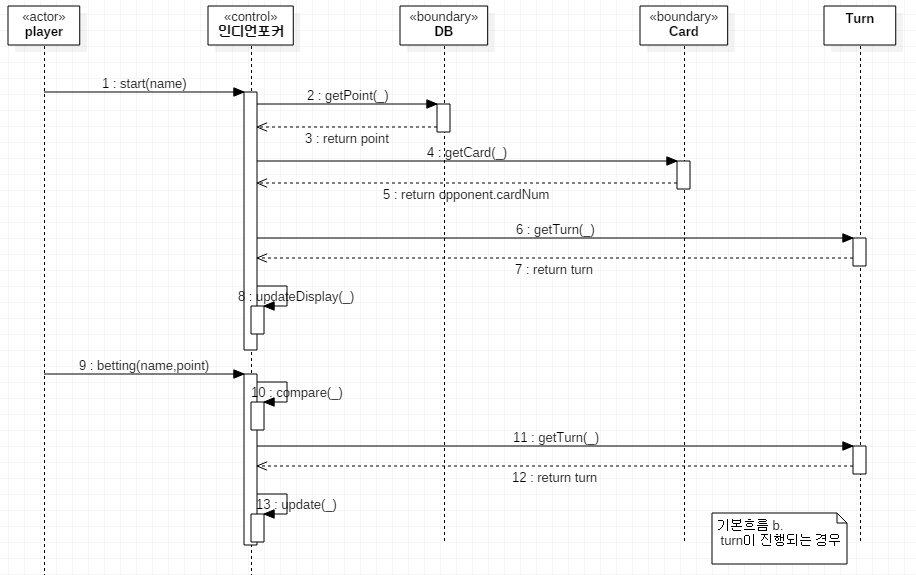
# **유즈케이스 실현**

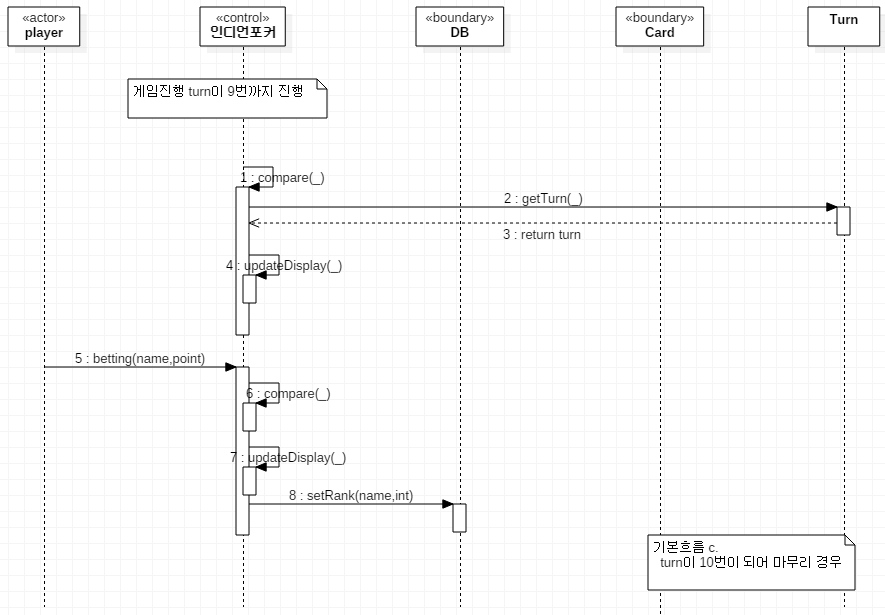
## 시퀀스 다이어그램

1. **기본 흐름**
   * 1. **기본흐름-Play**

|  |  |
| --- | --- |
| Actor Actions | System response |
| 1. Player가 게임을 시작한다. |  |
|  | 2. Player에게 기본점수 200점을 부여한다. |
| 3. Player가 카드를 뽑는다. |  |
|  | 4. 1~10랜덤 숫자를 뽑고 상대의 Player에게 카드 숫자를 보여준다. |
| 5. Player가 상대의 카드를 보고 10점부터 점수를 건다. |  |
| 6. 자신의 카드가 상대방 보다 높은 점수라고 생각한다면 배당점수를 늘린다. |  |
|  | 7. 배당 점수를 저장하고 상대방의 배당을 기다린다. |
| 8. 상대방의 배당 점수와 같은 점수를 건 경우이다. 한 턴이 종료된다. |  |
|  | 9. 두 Player의 카드 결과를 출력하고 승리자를 판단한다. |
|  | 10. 저장된 배당 점수를 모두 승리자에게 준다. |
|  | 11. 10회턴이 모두 끝나면 게임을 종료한다. |
|  | 12.Player의 승률을 Rank에 저장한다. |

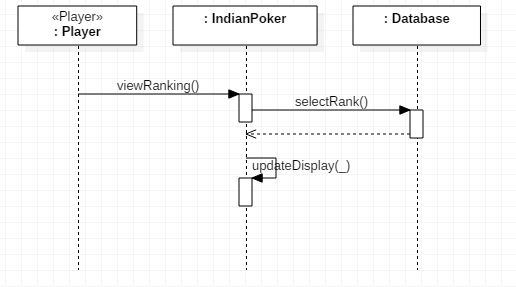






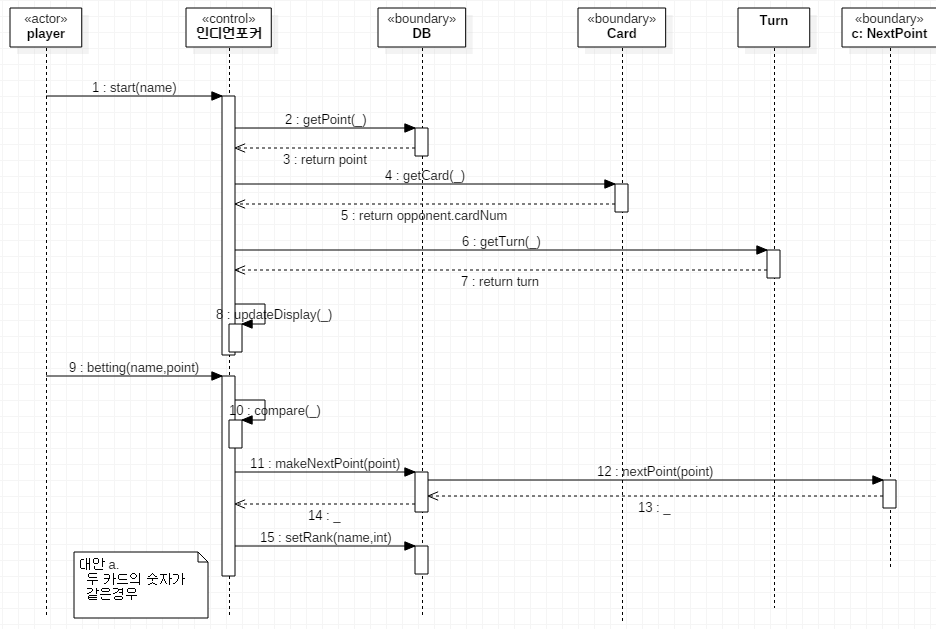
* + 1. **기본흐름 - viewRanking**

|  |  |
| --- | --- |
| Actor Actions | System response |
| 1.viewRank로 최종점수를 본다 |  |
|  | 2. 모든 플레이어의 점수를 출력한다. |



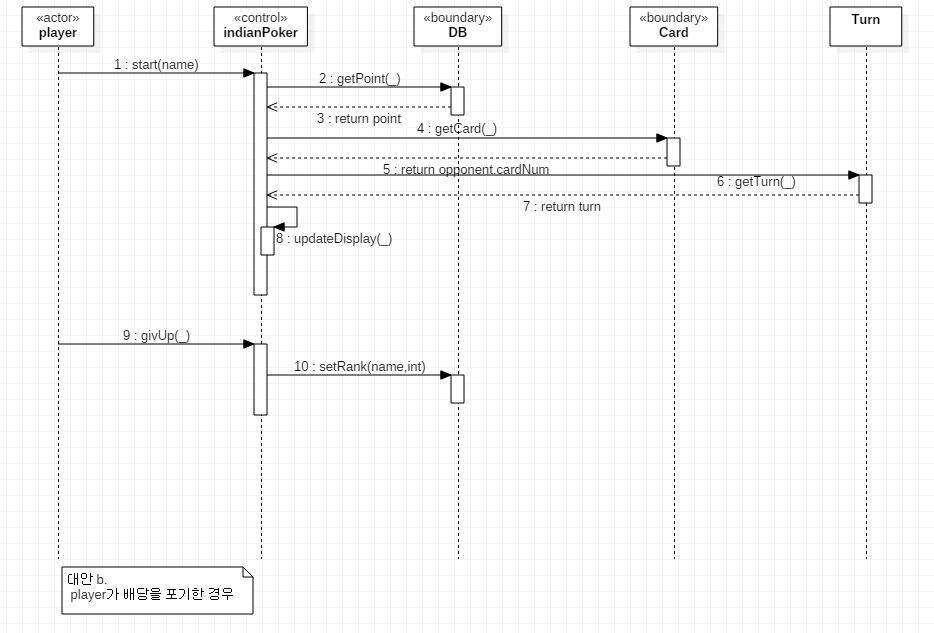
1. **대안흐름**
   * 1. **대체흐름 1 – 두 카드의 숫자가 같은 경우**

|  |  |
| --- | --- |
| Actor Actions | System response |
| 1. Player가 게임을 시작한다. |  |
|  | 2. Player에게 기본점수 200 점을 부여한다. |
| 3. Player가 카드를 뽑는다. |  |
|  | 4. 1~10랜덤 숫자를 뽑고 상대의 Player에게 카드 숫자를 보여준다. |
| 5. Player가 상대의 카드를 보고 10점부터 점수를 건다. |  |
| 6. 자신의 카드가 상대방 보다 높은 점수라고 생각한다면 배당점수를 늘린다. |  |
|  | 7. 배당 점수를 저장하고 상대방의 배당을 기다린다. |
| 8. 상대방의 배당 점수와 같은 점수를 건 경우이다. 한 턴이 종료된다 |  |
|  | 9. 두 Player의 카드 결과를 출력하는데 두 카드의 숫자가 같으므로 무승부를 판단하고 지금까지 배당된 점수를 다음 턴의 기본 배당 점수로 넘겨준다. |
|  | 10. 다음 턴을 실행한다. |



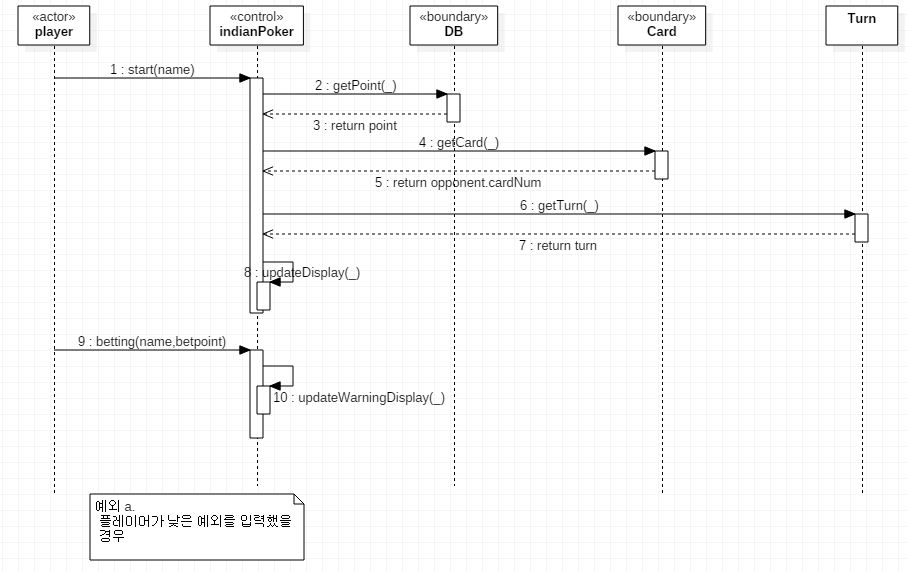
* + 1. **대체흐름 2 - Player가 배당을 포기한 경우**

|  |  |
| --- | --- |
| Actor Actions | System response |
| 1. Player가 게임을 시작한다`. |  |
|  | 2. Player에게 기본점수 200 점을 부여한다. |
| 3. Player가 카드를 뽑는다. |  |
|  | 4. 1~10랜덤 숫자를 뽑고 상대의 Player에게 카드 숫자를 보여준다. |
| 5. Player가 상대의 카드를 보고 10점부터 점수를 건다. |  |
| 6. 자신의 카드가 상대방 보다 낮은 점수라고 생각하여 배당을 포기한다. |  |
| 7. 포기한 Player의 카드 결과가 10일 경우이다. 상대Player에게 100점을 추가로 준다. |  |
|  | 8. 두 Player의 카드 결과를 출력하고 상대Player를 승자로 판단한다. |
|  | 9. 저장된 배당 점수를 모두 승리자에게 준다. |
|  | 10. 10회턴이 모두 끝나면 게임을 종료한다. |
|  | 11.Player의 승률을 Rank에 저장한다. |



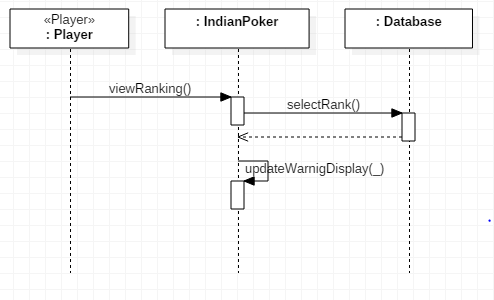
1. **예외흐름**

|  |  |
| --- | --- |
| Actor Actions | System response |
| 1. Player가 게임을 시작한다. |  |
|  | 2. Player의 점수를 가져온다. |
| 3. Player가 카드를 뽑는다. |  |
|  | 4. 1~10랜덤 숫자를 뽑고Player에게 상대의 카드를 보여준다. |
| 5.Player가 가진 점수보다 더 많은 점수를 건다. |  |
|  | 6.Player에게 점수가 초과되었다고 알리고 다시 입력하게 된다. |

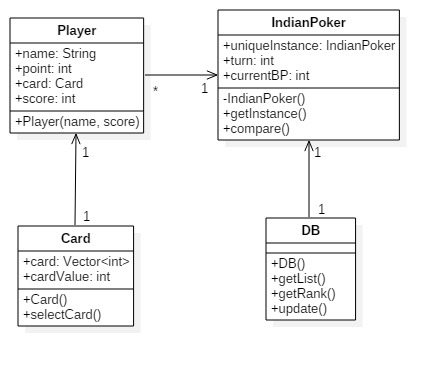


* + 1. **예외흐름 - viewRanking**

|  |  |
| --- | --- |
| Actor Actions | System response |
| 1.viewRank로 최종점수를 본다 |  |
| 2. 전체 랭킹이 아직 없는 경우이다. |  |
|  | 3.Ranking이 존재하지 않다고 출력한다. |

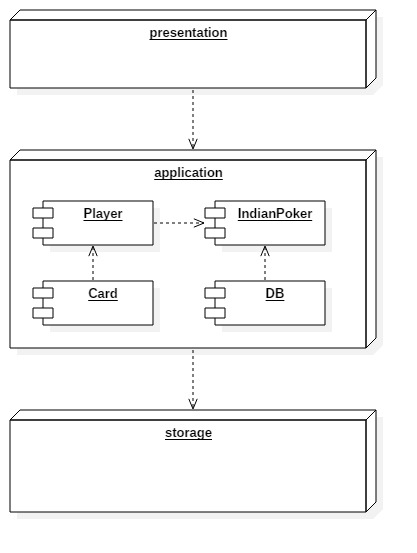


# **분석모델**



|  |  |
| --- | --- |
| 클래스 명 | Role & Relation |
| Player | 인디언 포커를 실행하는 actor |
| IndianPoker | Control 스테레오 타입.DB에서 플레이어에 대한 정보, Card에서 카드에 대한 정보, Turn에서 실행 횟수에 대한 정보를 받아와서 저장한다. 게임을 실행해서 배팅정보와 카드비교를 저장하고 플레이어에게 게임상황을 보여준다. |
| Card | 1~10 숫자카드를 2세트 가지고 있고, 랜덤으로 숫자카드를 뽑을 수 있도록 카드숫자순서를 랜덤으로 저장한다. |
| DB | Entity 스테레오 타입. 플레이어의 이름과 점수 저장하고, 플레이어의 승패를 DB에 기록한다. |

# **아키텍처 기술서**



3tier-아키텍처로 개발한다.

단, 분석에서는 응용계층만 다루기 때문에 다른 계층의 객체는 고려하지 않는다.

Application안에서는 Player에 사용자 정보를 가져와 play하면 IndianPoker가 Card에서 상대 선수의 숫자카드를 가져오고 DB에서는 기본 점수를 가져오거나 플레이어의 rank를 볼 수 있고, Turn에서 실행횟수를 받아와서 인디언포커게임을 10회 실행가능 하도록 한다.

# **유즈케이스 실현**

## UI design

1. **메인 화면**



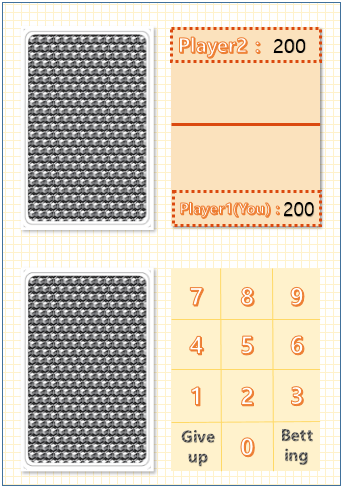
앱을 처음 시작했을 때 보이는 화면이다. start 버튼을 누르면 게임을 시작하는 화면으로 전환하고 Ranking 버튼을 누르면 랭킹을 보여주는 화면으로 전환한다.

1. **Highscore 화면**



Highscore 보기 화면이다. 메인 화면에서 "Ranking"버튼을 누르면 나오는 화면으로 현재까지 게임을 한 플레이어들의 점수를 높은 순으로 정렬하여 1위부터 보여주는 화면이다.

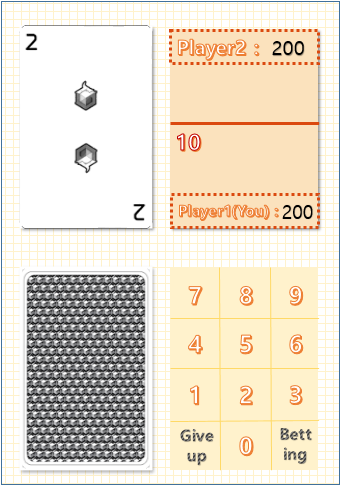
1. **게임 진행 첫 화면**

****

Player1과 Player2가 접속한 후 게임이 start될 때의 화면으로, Player1과 Player2는 기본점수 200점을 배당 받고 게임을 시작하게 된다. 왼쪽 상단은 상대방의 카드, 왼쪽 하단은 내가 뽑은 카드, 오른쪽 상단은 상대와 내가 베팅한 점수판, 오른쪽 하단에는 점수 베팅을 위한 숫자 버튼, 배당포기를 위한 Give up버튼, 그리고 점수를 배당한 후 제출하는 Betting 버튼이 존재한다.

이 화면은 아직 Player1과 Player2가 자신의 카드를 뽑기 전의 화면이다.

1. **게임 진행 두 번째 화면**

****

Player1과 Player2가 각자의 카드를 뽑은 후의 화면으로, 상대(Player2)가 뽑은 카드는 무엇인지 나에게 보이지만(왼쪽 상단의 카드), 내가 뽑은 카드는 무엇인지 화면 상에서 보이지 않는다(왼쪽 하단의 카드). 각 Player들은 상대의 카드를 보고 나의 카드숫자를 예측하여 점수를 배당한다. 기본 배당 점수는 10점이고, 이 화면은 Player1이 점수 10점을 배당한 상태의 화면이다.

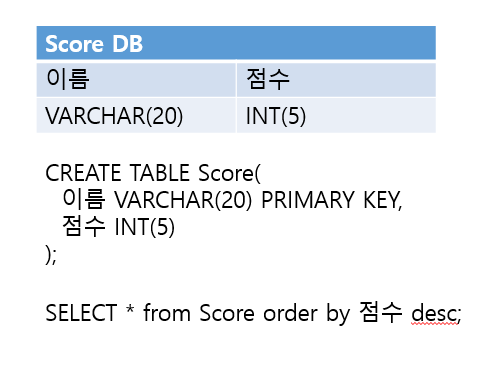
1. **게임 종료 화면**



게임이 종료되었을 때 나타나는 화면이다. 승자가 누구인지 알려주고 확인을 누르면 메인 화면으로 돌아가게 된다. Player1과 Player2 모두에게 동일하게 나타나는 화면이다.

# **DB design**

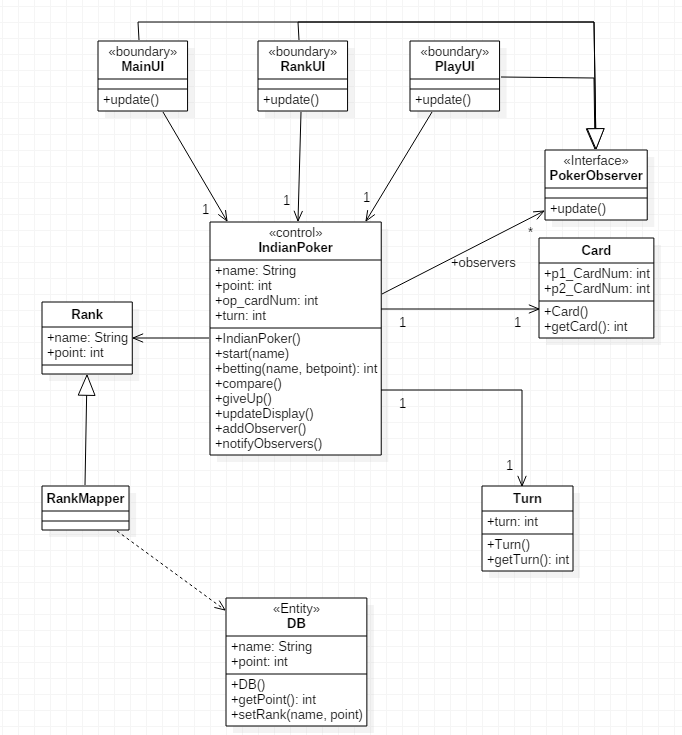
데이터베이스 스키마



게임이 종료된 후 랭킹을 보기 위한 플레이어의 이름과 최종 점수를 기록하는 데이터베이스이다. Score 테이블의 속성 중 이름은 varchar(20)타입, 점수는 int(5)타입이며 이름을 기본키로 설정하였다. 또한 랭킹을 확인할 때는 점수가 높은 순으로 정렬하여 확인하도록 select문 설계하였다.

# **디테일한 Class Diagram**

클래스 다이어그램



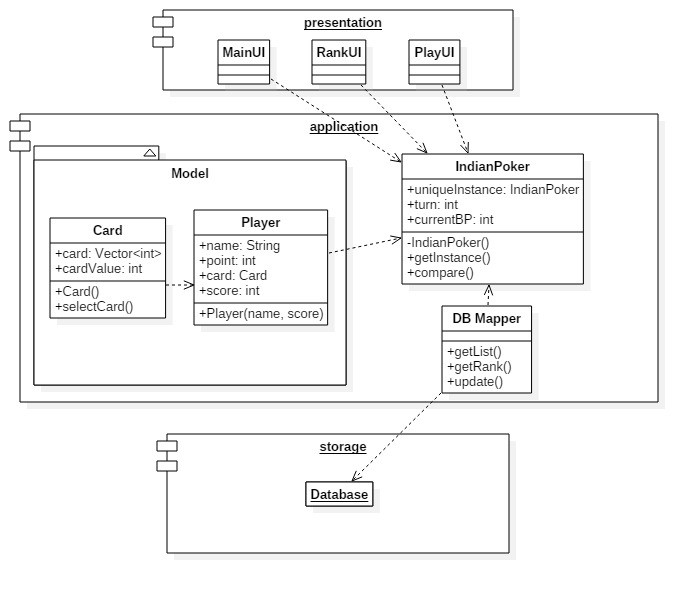
기존 분석 단계에서 만들었던 클래스다이어그램에서 수정된 부분을 말한다. UI부분을 구현해줄 MainUI, RankUI, PlayUI 클래스가 추가 되었다. Observer 패턴을 이용해서 UI와 Control을 연결해주기 위해 Observer 인터페이스를 만든다.

Rank를 저장할 Rank 클래스를 추가하고 DB의 테이블로 매핑해주기 위한 RankMapper클래스도 만든다.

Player 클래스는 control과 겹치고 actor이기 때문에 삭제하였다.

.

# **Component Diagram**



클래스 다이아그램을 바탕으로 MVC패턴을 적용하여 컴포넌트 다이아그램을 작성하였다. 표현은 응용계층의 영향을 받지만 위로 올라가는 형태를 MVC패턴은 나타낼 수 없기 때문에 Observer패턴을 사용하여 인터페이스를 두고 update를 받도록 하였다. 또한 저장계층은 DB스키마를 적용하기 때문에 사용자 정보를 DB와 응용계층이 서로 이어지도록 mapper클래스를 이용하였다.

# **Client**

## Android Network

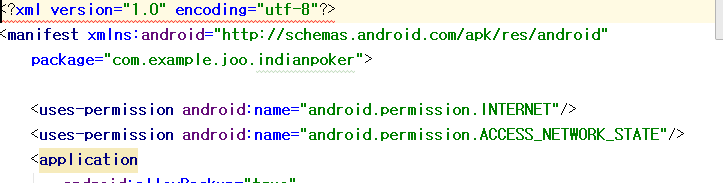
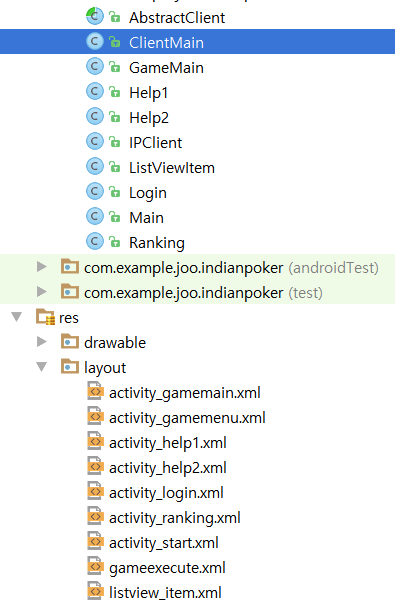
Manifest.xml에서 네트워크 접근 구문 삽입. 네트워크 제한을 풀어주는 구문 삽입

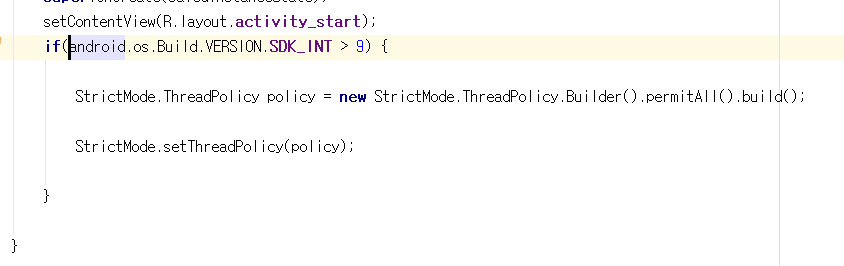
## Socket 통신

ClientMain클래스에서 서버의 IP주소와 port번호 삽입

## Thin Client

MVC패턴에서 View를 담당. 3-tier의 presentation layer를 담당



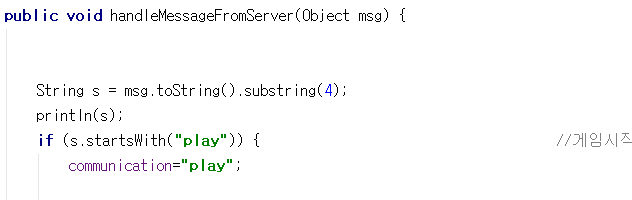


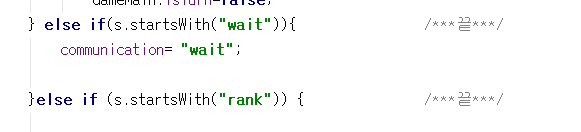
## 메시지 처리

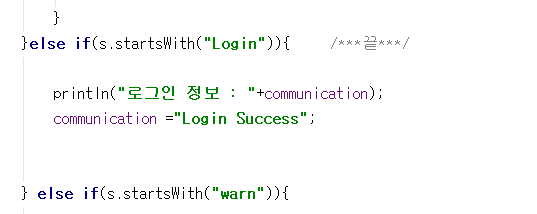
Eclipse 서버에서 handleMessageFromServer()를 통해 메시지를 전달받음

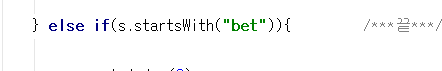
해당 메시지의 형식에 따라 UI에 나타낸다.

클라이언트에서는 해당 버튼에 따른 결과정보를 sendToServer()를 통해 전달









# **Server**

## 13.1 OCSF 프레임 워크를 사용

AbstractServer 클래스를 상속받음. ConnecetionToClient 클래스로 client 관리함.

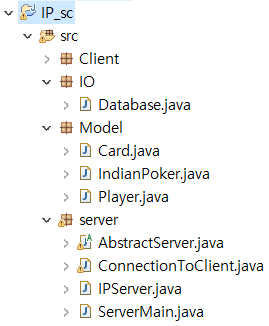
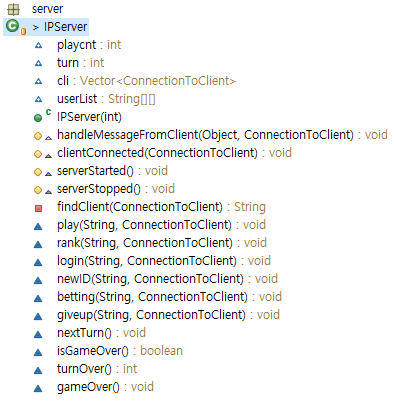
## 씬 클라이언트 구조

MVC패턴의 Model, Controller 부분을 담당. 3-tier의 application layer를 담당

## 메시지 처리

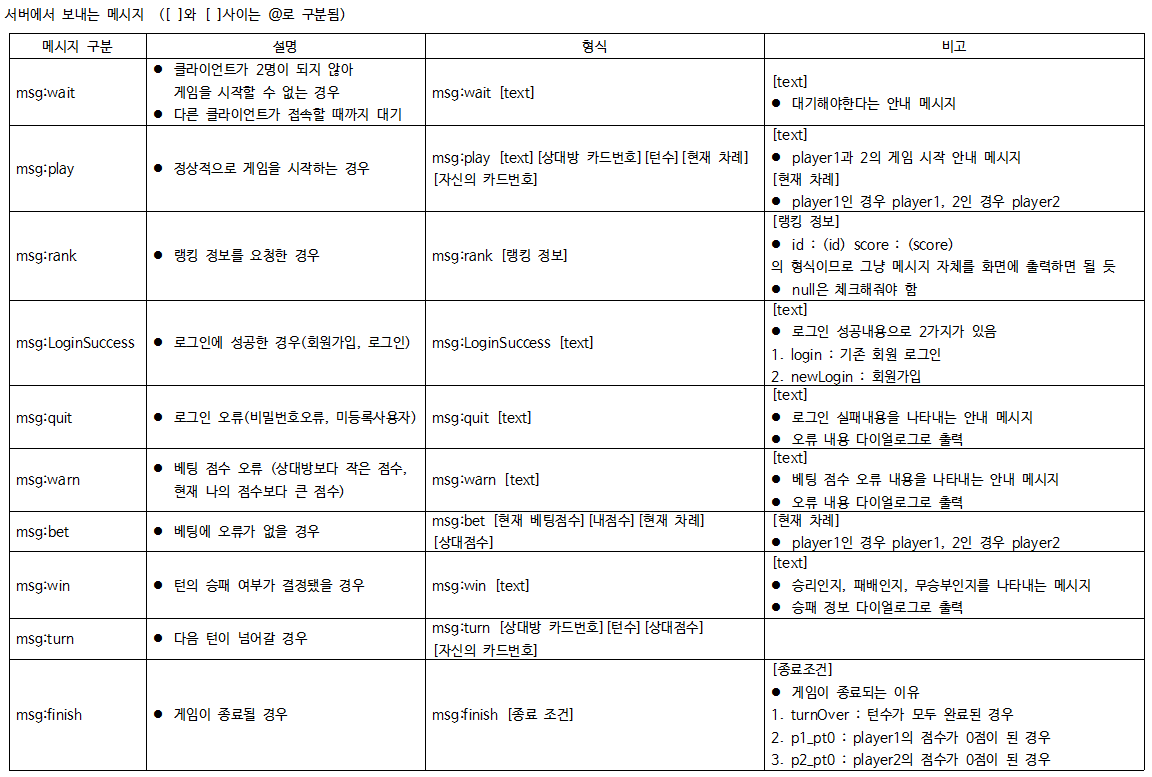
handleMessageFromClient 메소드로 Client의 요청에 따라 맞는 기능 수행 후,

메시지 전달. 클라이언트의 요청에 따라 기능 수행 후, 수행 결과를 client에게 전송





* + 1. **13.3.1 서버에서 클라이언트로 보내는 메시지 형식**

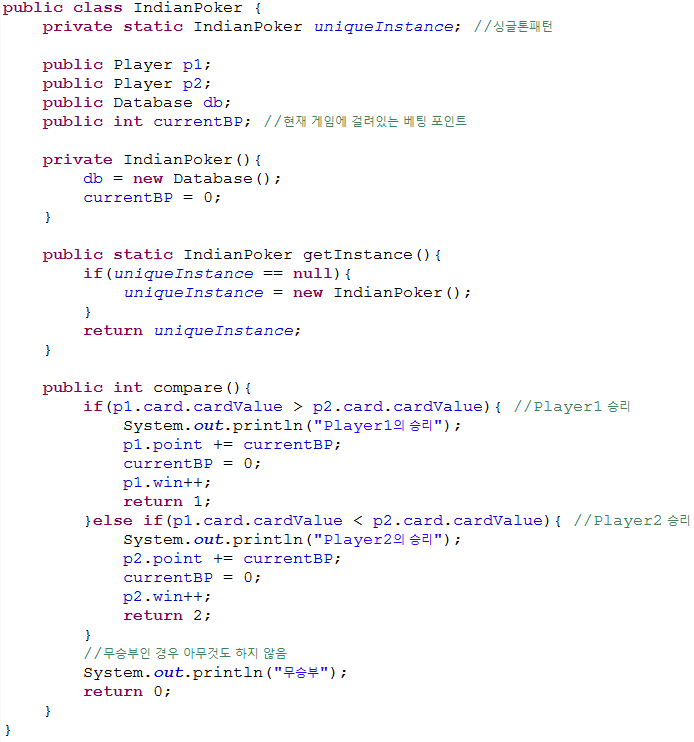


클라이언트에서는 이 형식에 맞춰 메시지를 파싱하여 메시지에 담긴 정보를 알아냄

## 게임의 핵심 로직을 담당

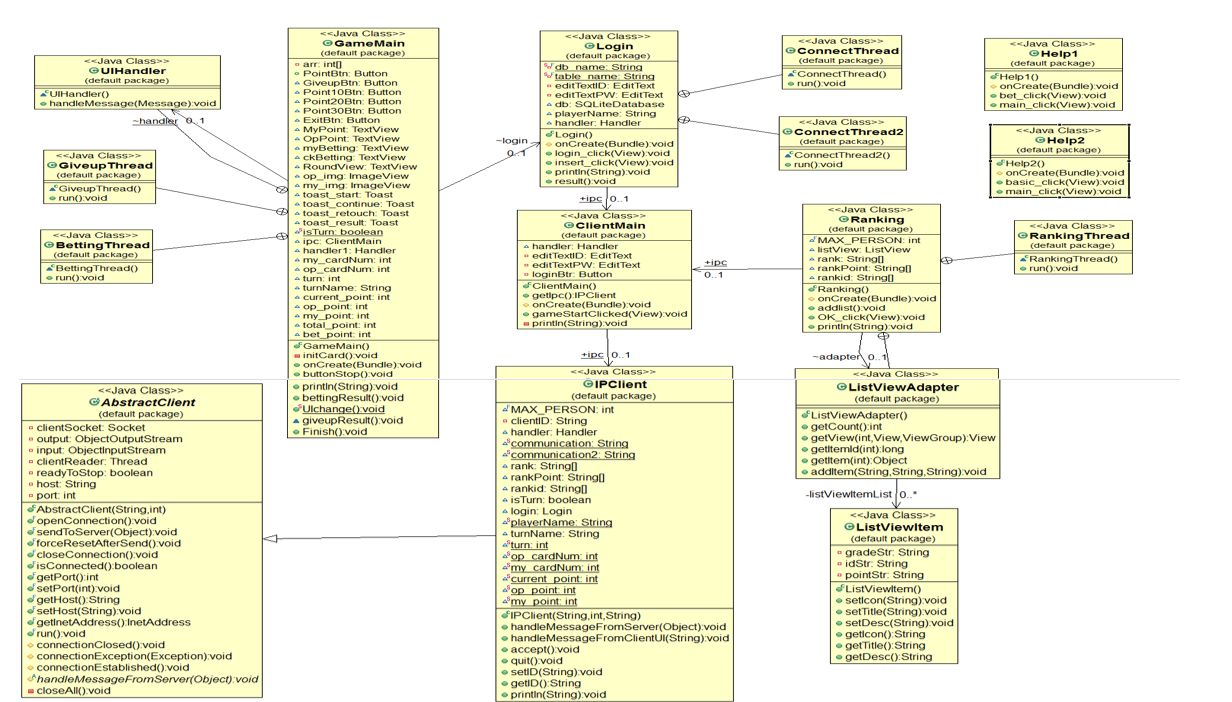
게임 규칙이 바뀔 경우, client와 관계 없이 model에 해당하는 클래스만 수정하면 됨   
🡪 유지보수가 용이

Singleton 패턴 사용으로 유일한 객체를 생성함.

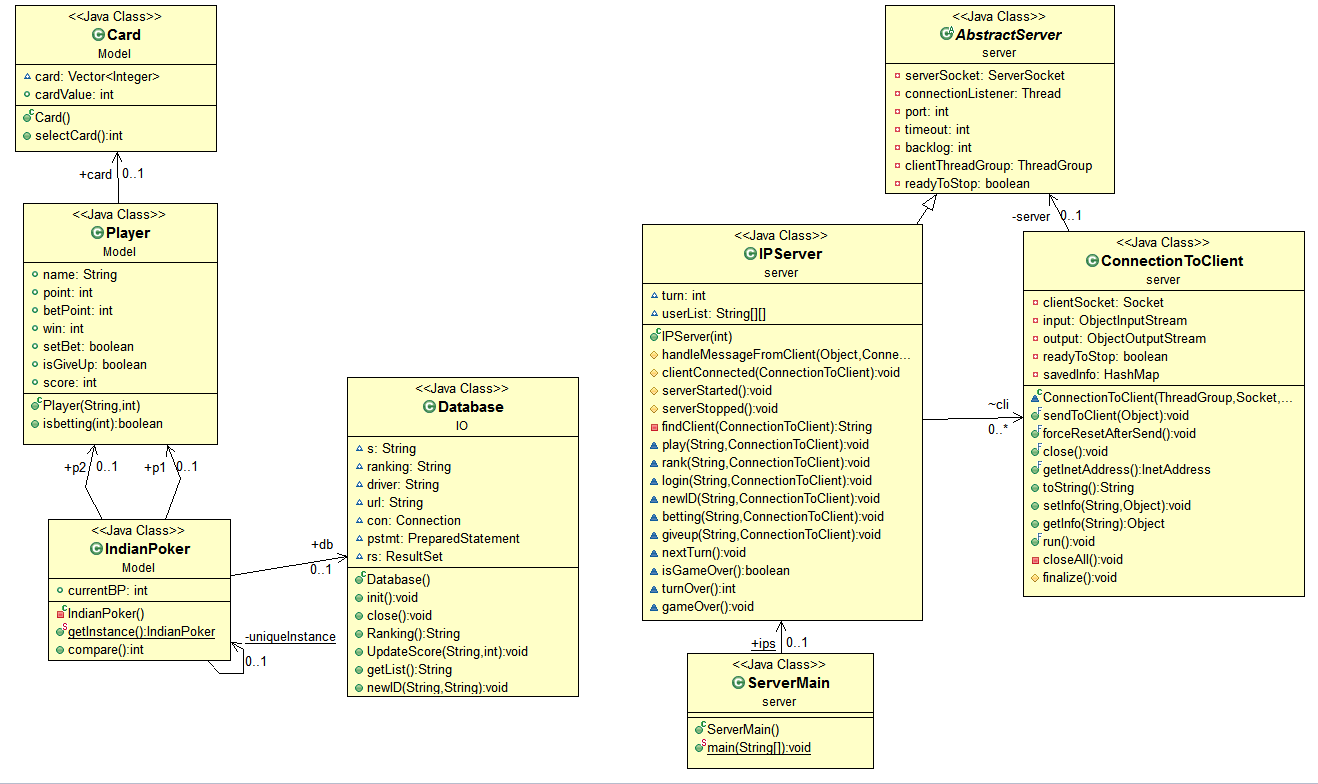


# **Class Diagram**

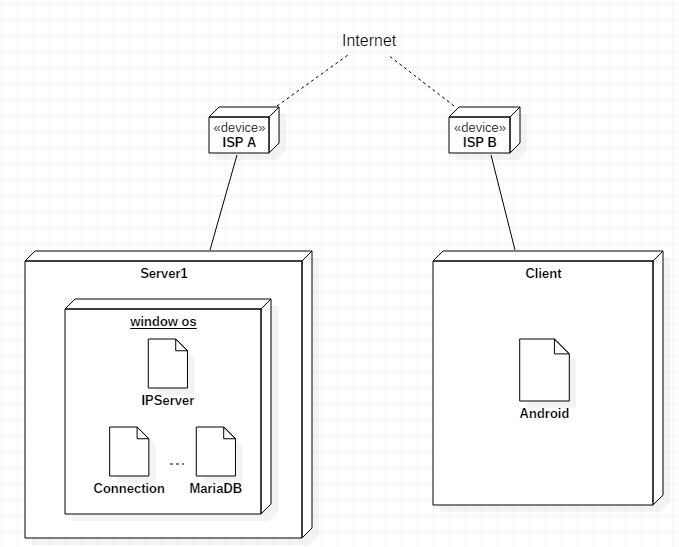
## Client



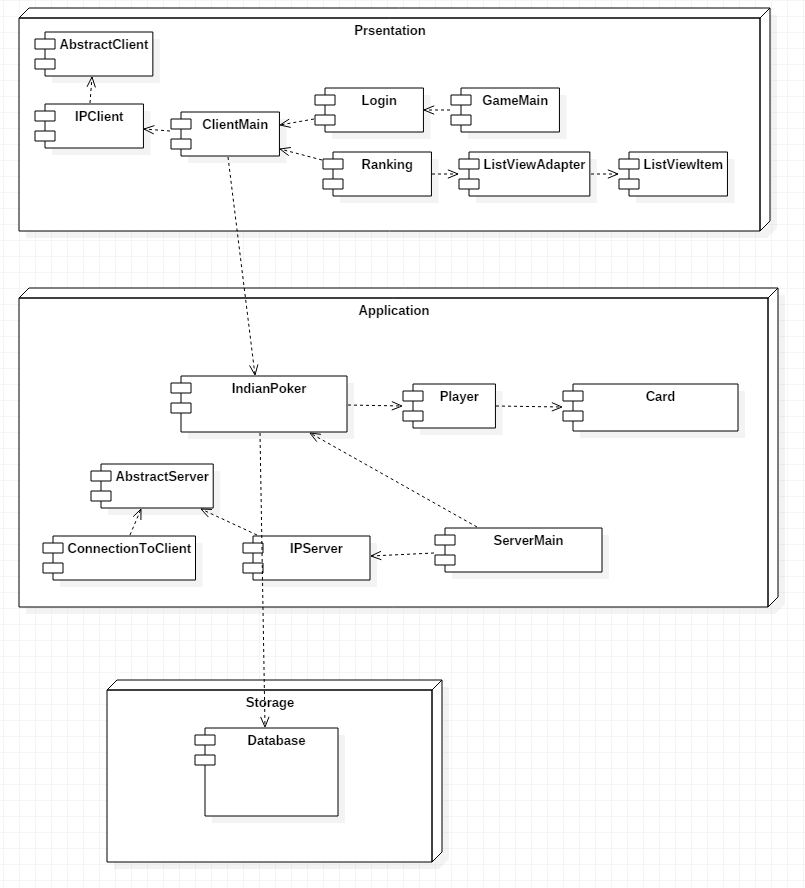
## Server



# **Deployment Diagram**



# **Component Diagram**



# **테스트 케이스**

