**Subgroup Patient**

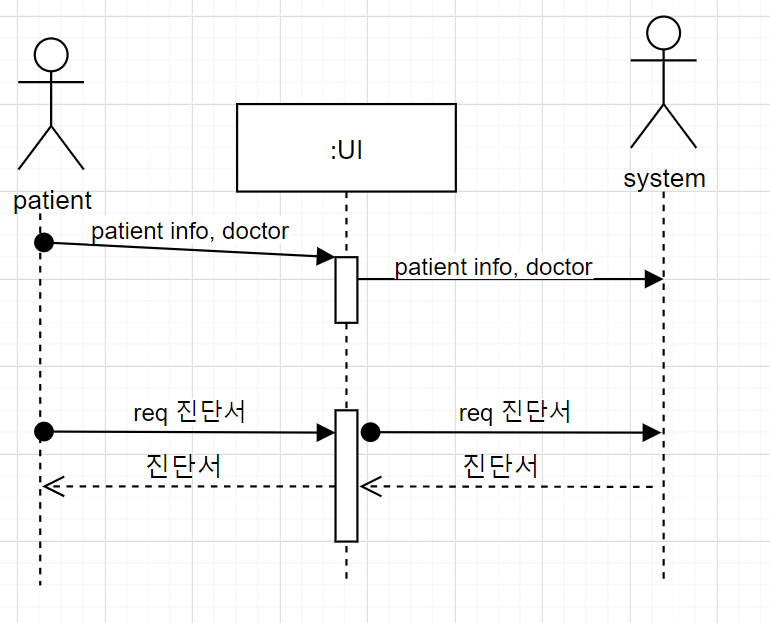
**Object Sequence Diagram & Class Diagram**

**CAUSW 신동준 20191635  
CAUCSE 이상진 20162191**

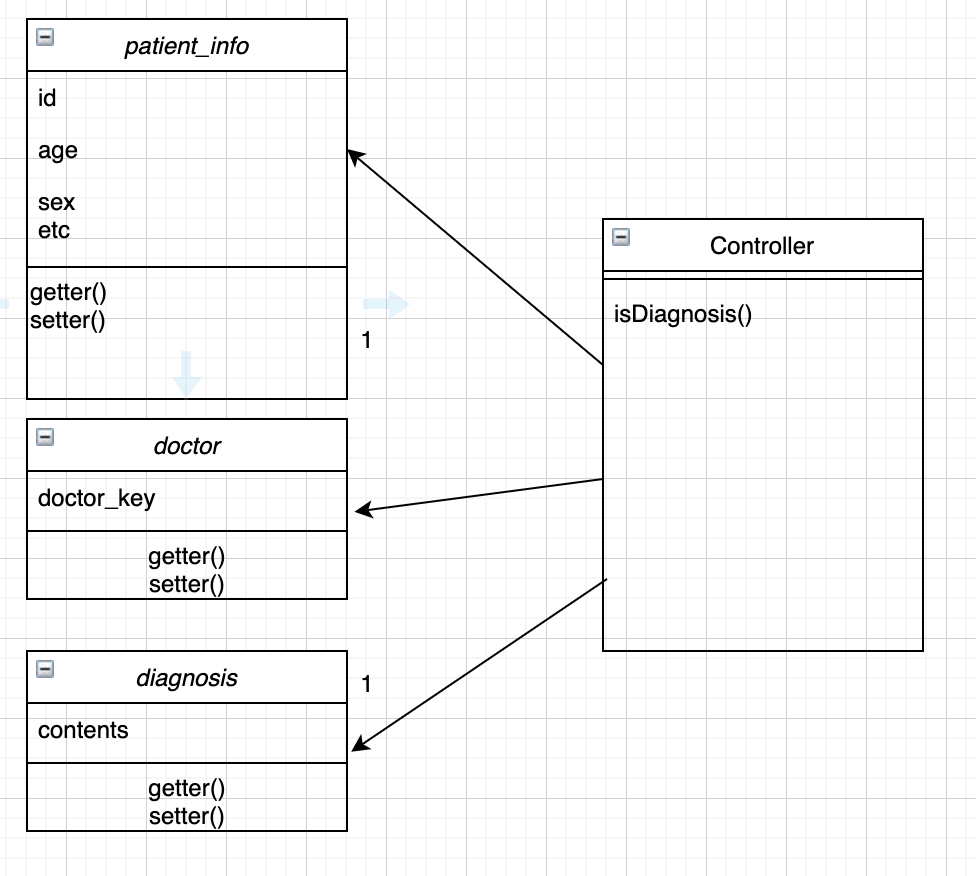
**Index**

1. **UC1-2  
   \* Sequence Diagram  
   \* Class Diagram**
2. **UC4  
   \* Sequence Diagram  
   \* Class Diagram**
3. **UC5  
   \* Sequence Diagram  
   \* Class Diagram**
4. **UC6  
   \* Sequence Diagram  
   \* Class Diagram**

* **UC1-2: 진단 (환자 부분)**

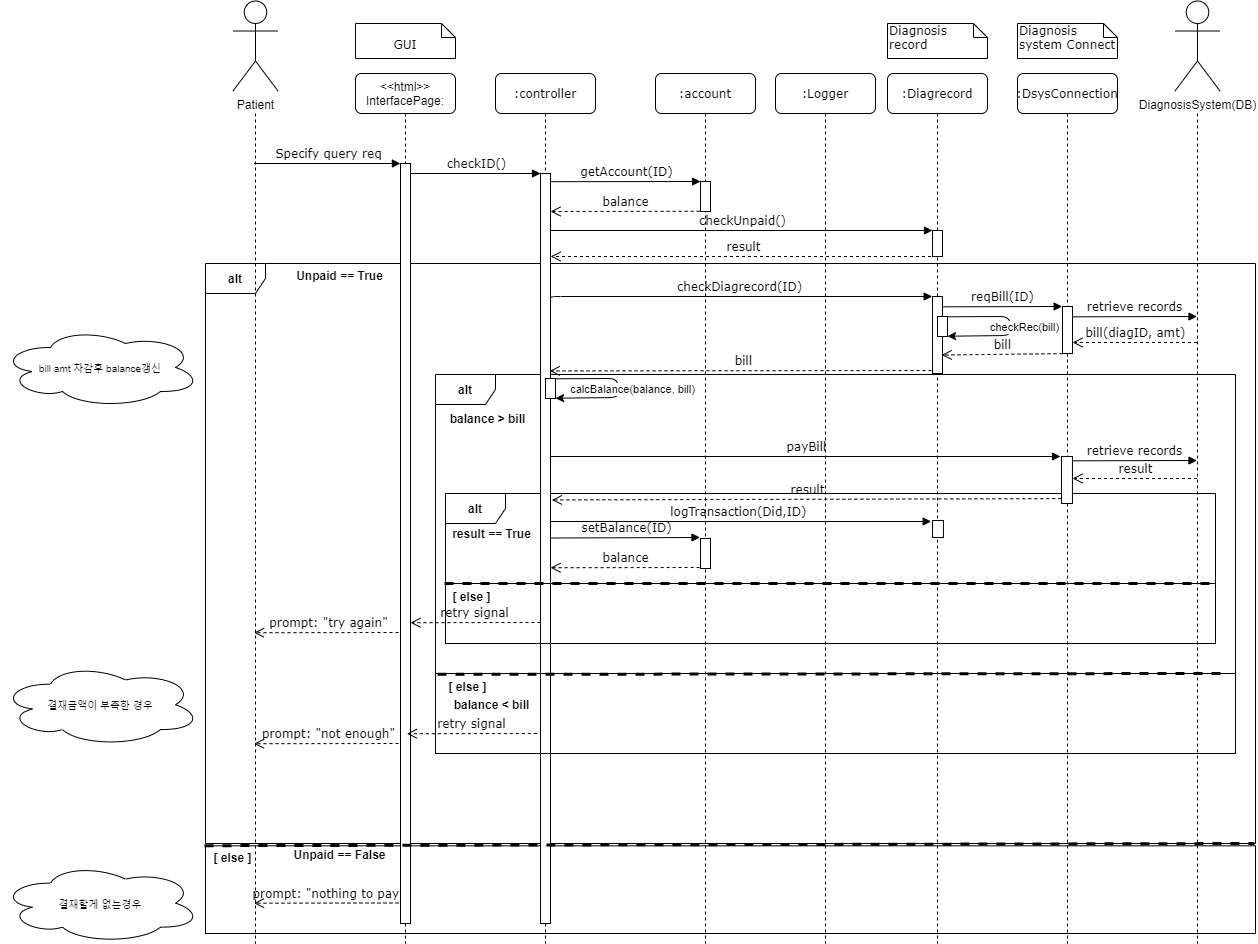
****

환자 진단 부분에서 환자는 시스템에게 자신의 정보와 어떤 닥터에게 정보를 보낼 것 인지에 대한 정보를 시스템에 보낸다. 그 이후 시스템을 통해서 의사가 진단을 완료하고 환자가 진단서에 대한 요청을 보내면 시스템에서 진단 서를 받아온다



클래스 다이어그램은 위의 사진과 같다 controller를 제외한 나머지 클래스는 정보를 담고 있는 객체라서 getter()와 setter()메소드들만 존재한다 controller 에는 정보를 진단서가 있는지 확인해주는 메소드가 있다.

* **UC4: 진료비 계산**

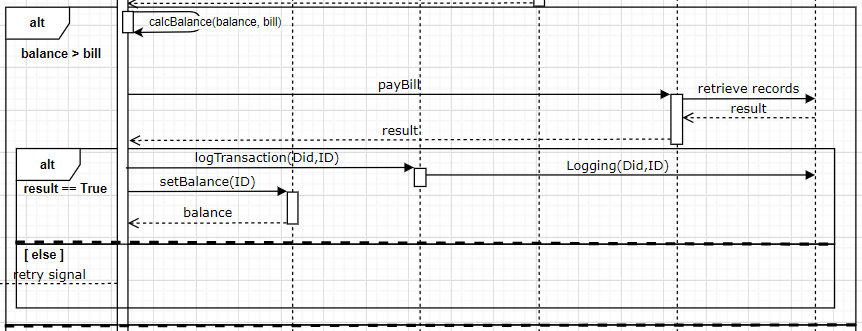


해당 그림은 ‘UC-4: 진료비 계산’에 해당하는 sequence diagram이다.  
actor Patient가 진료비 계산을 계좌에 대한 요청을 보내면 controller는 해당 요청을 보낸 actor의 계좌정보를 받아온다. 이후 controller는 자동적으로 진료기록을 열람하여 아직 비용이 지불되지 않은 진료가 있는지를 확인 후, 비용지불이 이루어져야 할 경우 비용지불 과정을, 아닌 경우 계좌정보를 돌려주는 작업을 진행한다.

controller에서 계좌 정산과 송금 까지의 과정을 모두 맡는 것이 controller의 역할에 부합하지 않을 수도 있다는 생각을 했었지만, 단순 연산을 통한 controller object로의 작업부하가 크지 않으며, 진료비 결제와 관련된 전체 과정을 관리한다는 관점으로 controller를 사용하는 것이 차후 구현의 확장에도 이점이 있을 것으로 판단되어 하나의 object를 통해 작은 작업들도 한꺼번에 처리하도록 설계하였다.

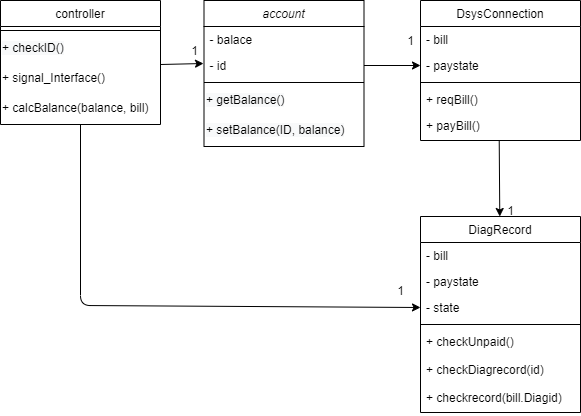
해당 다이어그램을 통한 작업은 actor인 Patient에 의하여 수동적으로 시작되며, actor의 요청이 생길 경우 또, 다른 actor인 Diagnosis System은 Patient의 요청에 수동적으로 반응하며 요청이 들어올 시, Diagnosis System과 연결된 DB내 조회를 통해 요청한 정보를 반환한다.

요청된 결제가 성공할 시, Diagnosis System에는 계산된 결제 결과를 저장하기 위한 DB가 따로 있으며, 이 DB내에는 결제로그가 저장된다. 로그가 저장된 이후, 계좌 내 잔고를 update하여 주어 actor에게 보이는 잔고내역을 갱신한다. 만일 결제가 전체 중 어느 과정 에서라도 실패하는 경우 결제 과정을 재 실행하는 요청을 patient actor에게 보내어 과정을 다시 진행하도록 만든다.

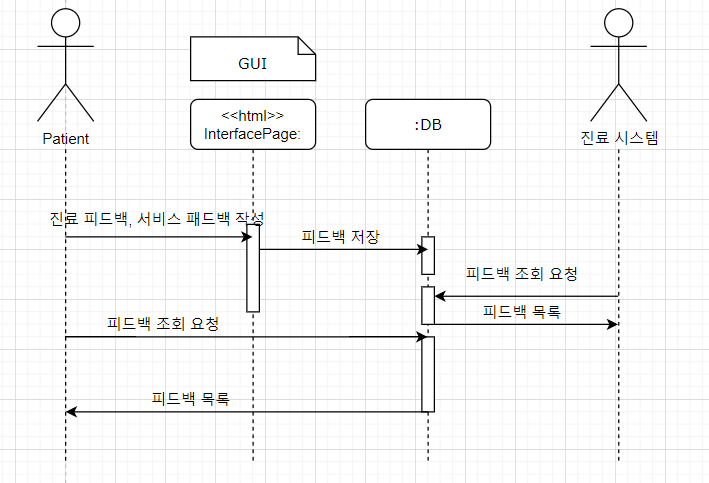


설계의 명시성과 구현의 편의를 위해 결재의 각 과정에서 발생할 수 있는 예외 상황들에 대해서는 sequence diagram에 표기하지 않았다. (파란색으로 표시한 부분이 결제과정에서 예외적 상황들이 나올 수 있는 곳이다.)

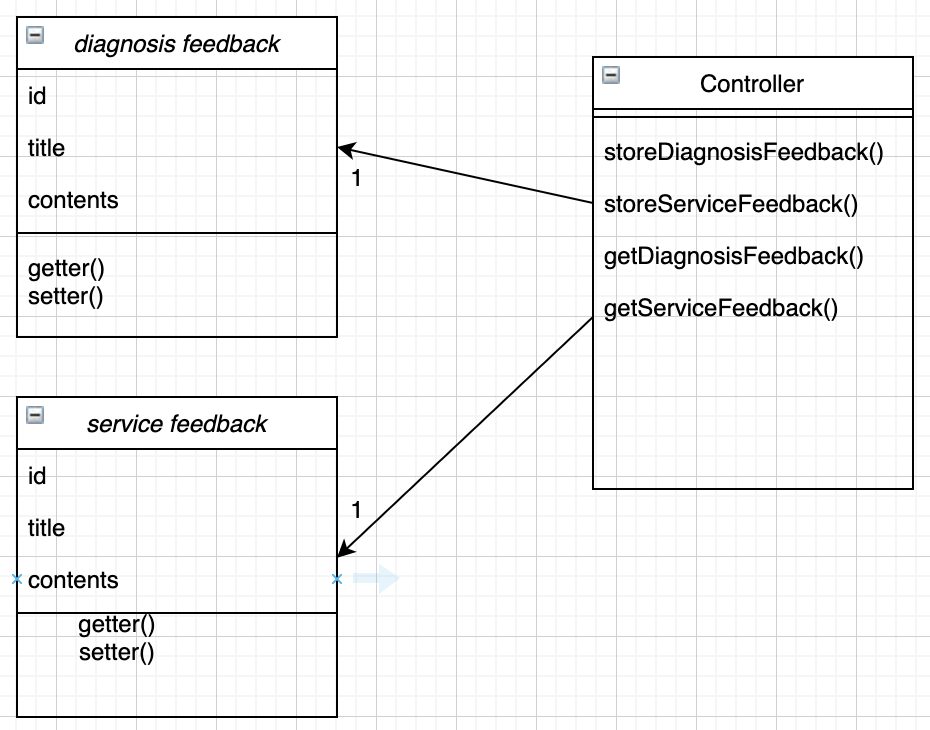
‘UC-4: 진료비 계산’의 클래스 다이어그램은 다음과 같다.



* **UC5: 진료 피드백**

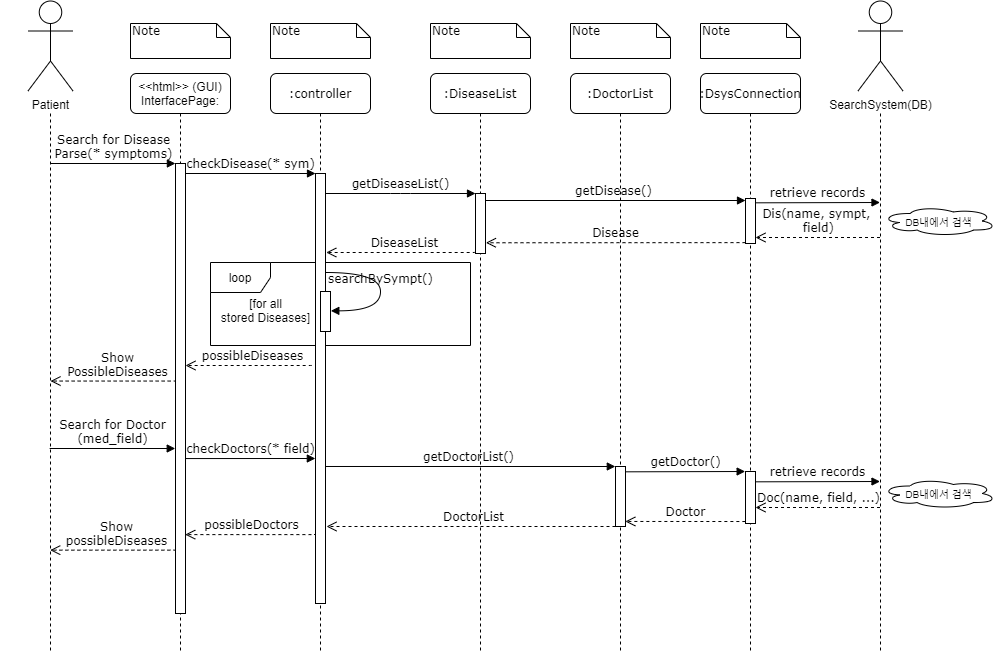


이 시퀀스 다이어그램에서 환자는 피드백을 작성해서 db에 저장한다.  
그리고 어드민과 의사는 진료 시스템을 통해서DB에 피드백 조회를 요청하고 피드백 목록을 받는다. 환자는 DB에 피드백 조회를 요청하고 피드백 목록을 받는다.



클래스 다이어그램은 다음과 같다. Controller 에는 피드백들을 관리하는 메소드 들이있다. 각각의 피드백 메소드에는 사용자id, 제목, 내용 테이블이 있고 이들의 정보를 수정하거나 가져 올 수 있는 메소드들이 있다.

* **UC6 : 검색**



해당 그림은 ‘UC-6: 검색’에 해당하는 sequence diagram이다.  
Actor인 Patient는 자신의 증상들을 작성하여 GUI를 통해 controller로 보내게 되고, controller에서는 SearchSystem내에 있는 DB에서 해당 증상들에 대한 질병을 조회하여, 질병들을 DiseaseList object로 보내게 된다. 이후, 해당 질병리스트 내의 정보들은 다시 controller로 넘어가게 되어, 증상과 연관성이 높은 질병들을 탐색한 후, 가능한 질병목록들을 다시 Actor patient에게로 넘겨준다.

이후, Actor인 Patient가 자신을 진료해 줄 의사를 검색하게 된다면, 이전 과정에서 조회된 질병의 분야에 따라 Patient가 능동적으로 선택하여, 해당 field에 해당하는 Doctor를 찾는 과정을 거치게 된다. 이 과정에서는 doctorList를 얻기 위해 controller가 doctorList object를 통해 SearchSystem까지 요청을 하게 되고, DB에서는 해당 분야에 맞는 의사들을 조회하여, 반환하게 된다.  
Actor Patient는 진료가 가능한 의사들의 목록을 받게 되고, 이 목록을 통해 patient는 이후 진료 예약을 진행할 수 있게 된다.

‘UC-46: 검색’의 클래스 다이어그램은 다음과 같다.

