2021-1 소프트웨어공학

<2nd Checkpoint>



Professor:

김명호 교수님

Team Name:

고재원

범동권

이해인

장동재

전예원

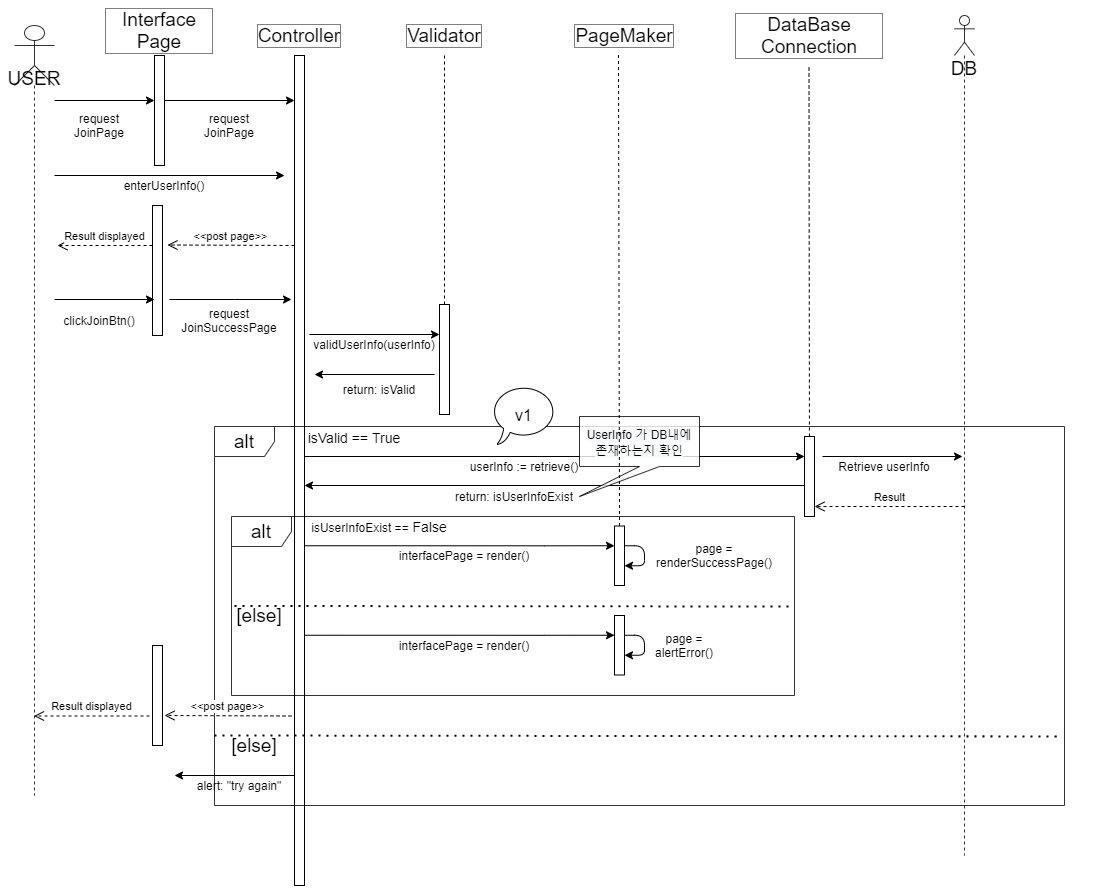
최승원

**SubGroup1**

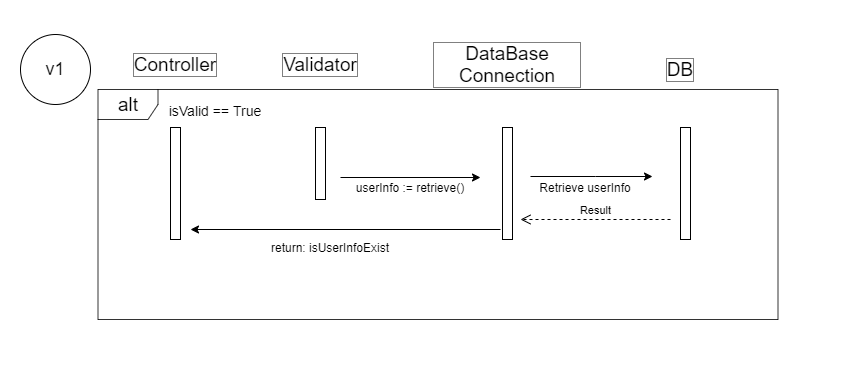
**회원가입, 로그인, 관리자 기능 [범동권, 이해인]**

**UC-1 : Join**

**[sequence diagram]**



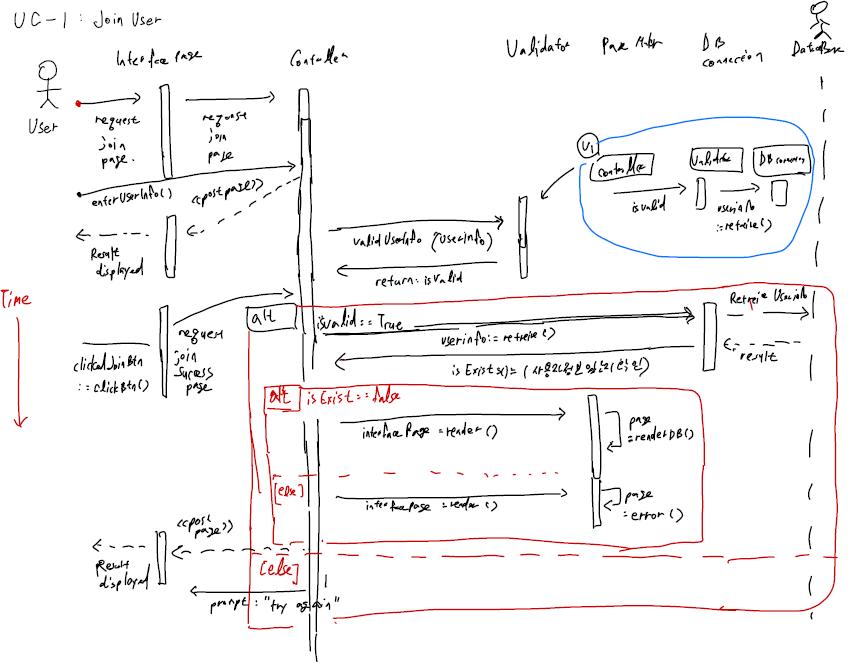
**Variation 1**



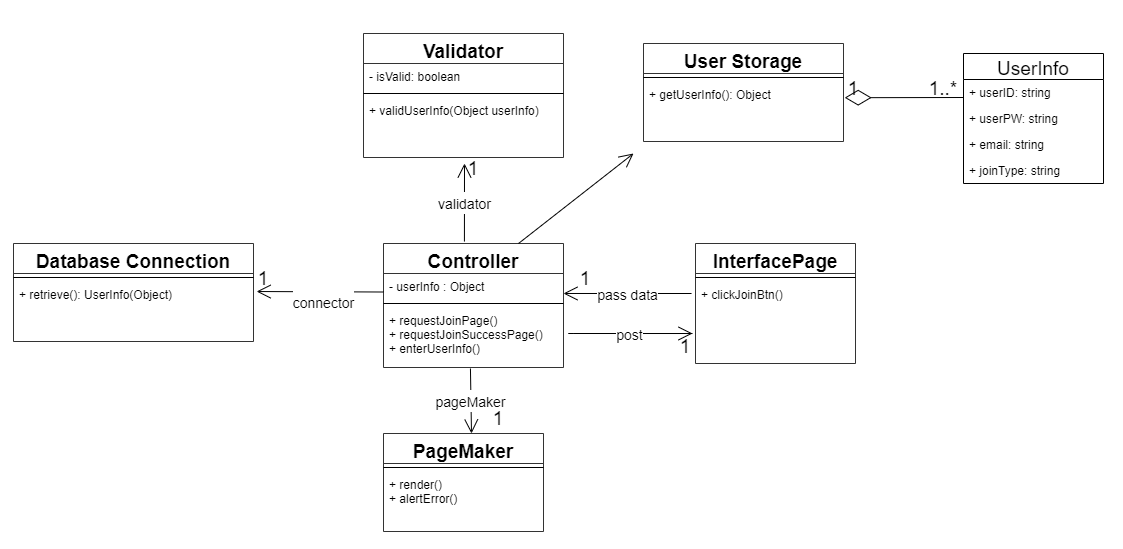
[Process]

초기 설계 당시에는 Controller에 전문성을 부여하여 validator에서 회원가입 양식의 form이 validate하다면 controller에서 valid한 회원가입 정보를 데이터베이스에 전달하는 방식이였다면 variation1에서는 validator에서 자체적으로 검증하고 controller에 valid를 던지는 방식이 아닌 해당 step에서 바로 데이터베이스에 회원가입 정보를 전달하는 방식을 고안하였다. 이를 통해 얻게 되는 이점은 controller에 하중된 책임들을 다른 domain concept로 분산시킴으로써 workload를 밸런스하게 맞출 수 있고 각 object간의 short communication chain이 가능하다.

**[초기 작업 설계 시 UC-1 Sequence Diagram Processing]**

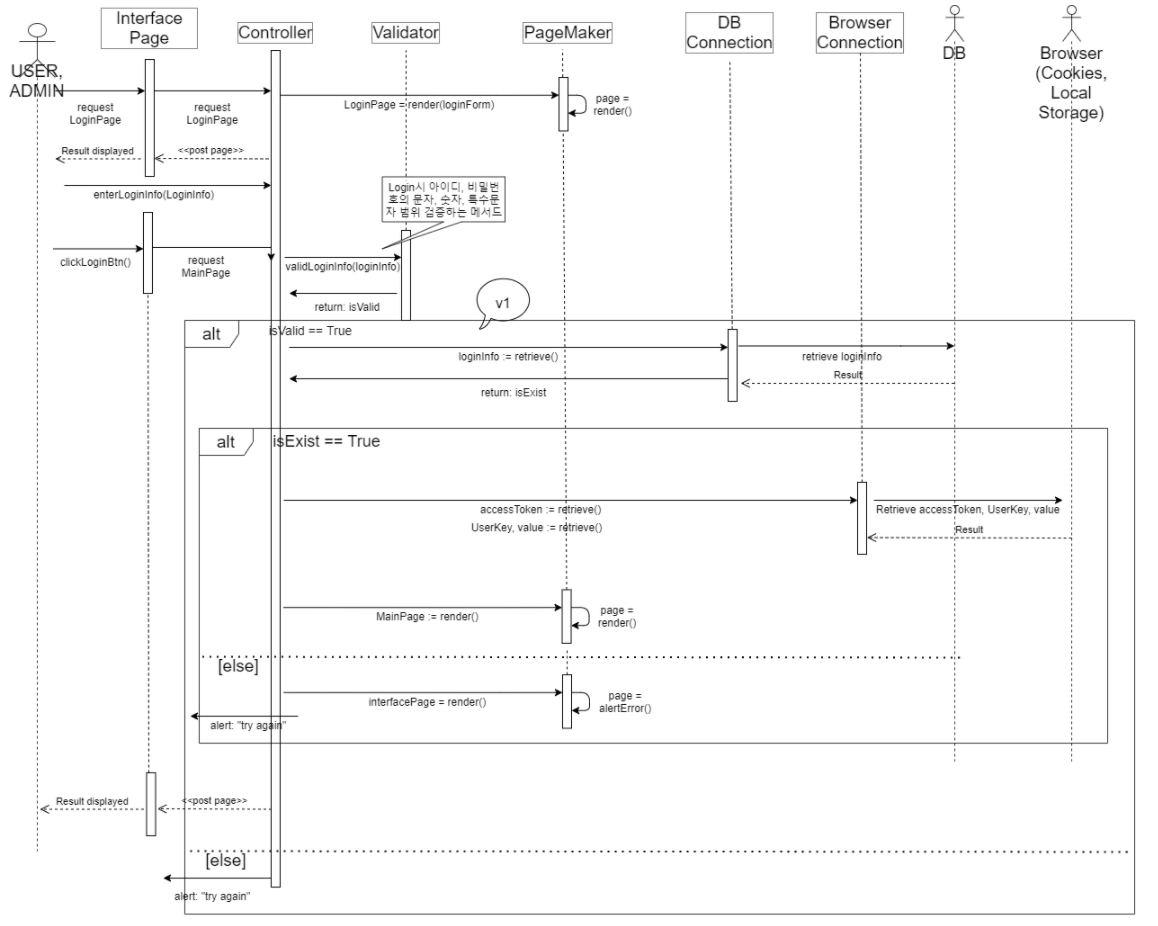


**[class diagram]**

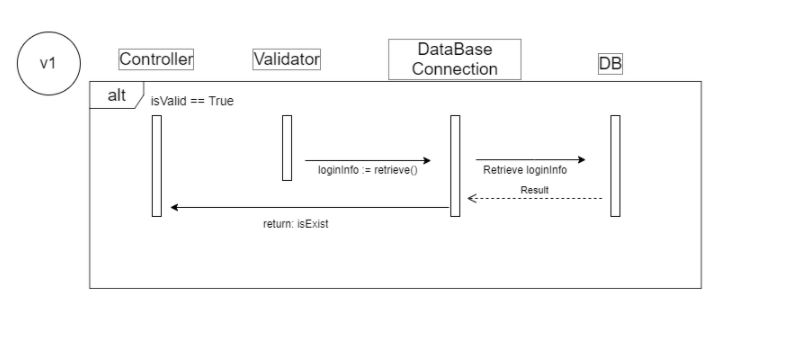


**UC-2 : Login**

**[sequence diagram]**



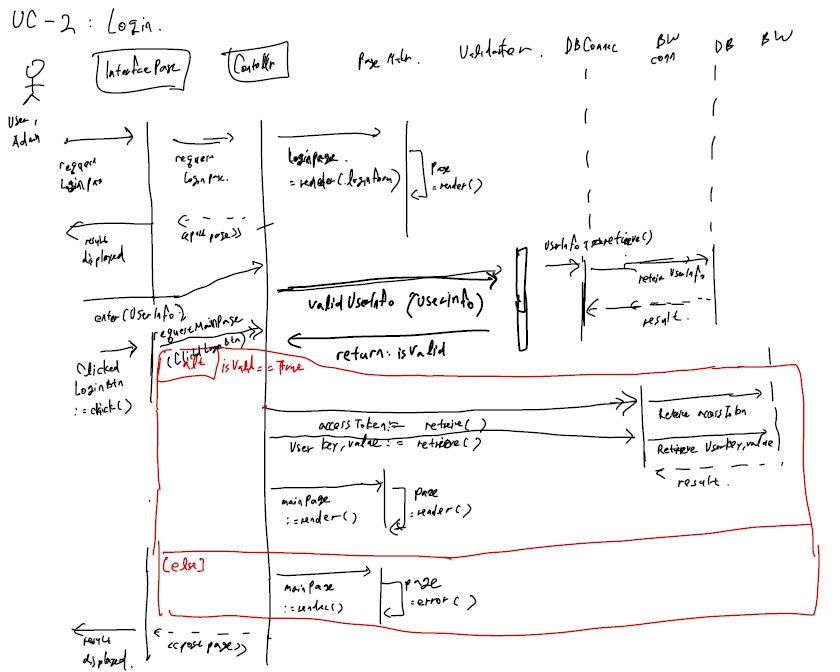
**Variation 1**



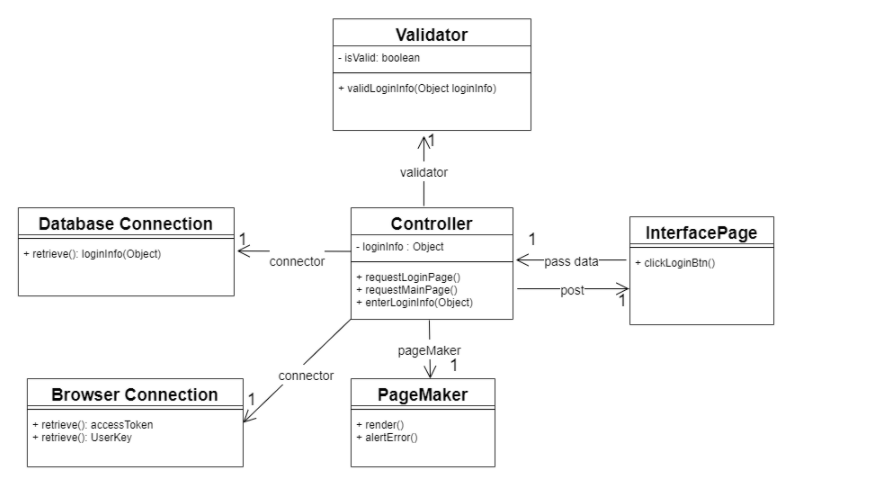
[Process]

Usecase 1과 마찬가지로 초기 설계 당시에는 Controller에 전문성을 부여하여 validator에서 로그인의 정규표현방식이 validate하다면 controller에서 valid하다는 것을 리턴받고 valid한 로그인 정보를 데이터베이스에 전달하는 방식이였다면 variation1에서는 validator에서 자체적으로 검사하고 controller에 valid를 던지는 방식이 아닌 해당 step에서 바로 데이터베이스에 로그인 정보를 전달하는 방식을 고안하였다. 이를 통해 얻게 되는 이점은 마찬가지로 controller에 하중된 책임들을 다른 domain concept로 분산시킴으로써 workload를 밸런스하게 맞출 수 있고 각 object간의 short communication chain이 가능하다.

**[초기 작업 설계 시 UC-2 Sequence Diagram Processing]**

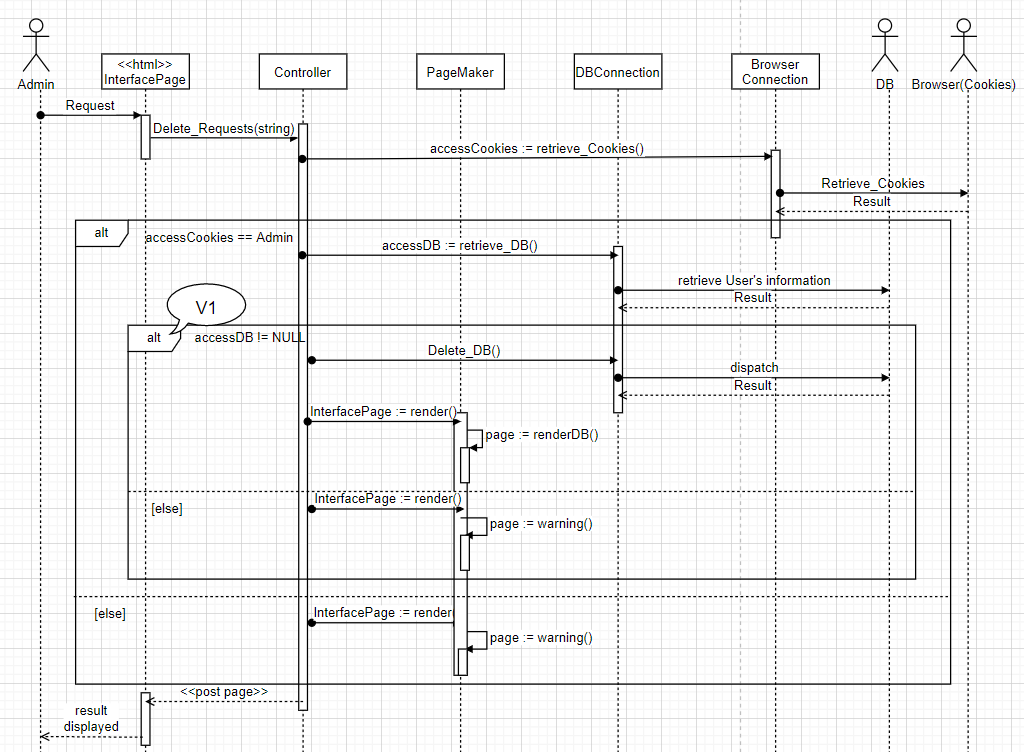


**[class diagram]**

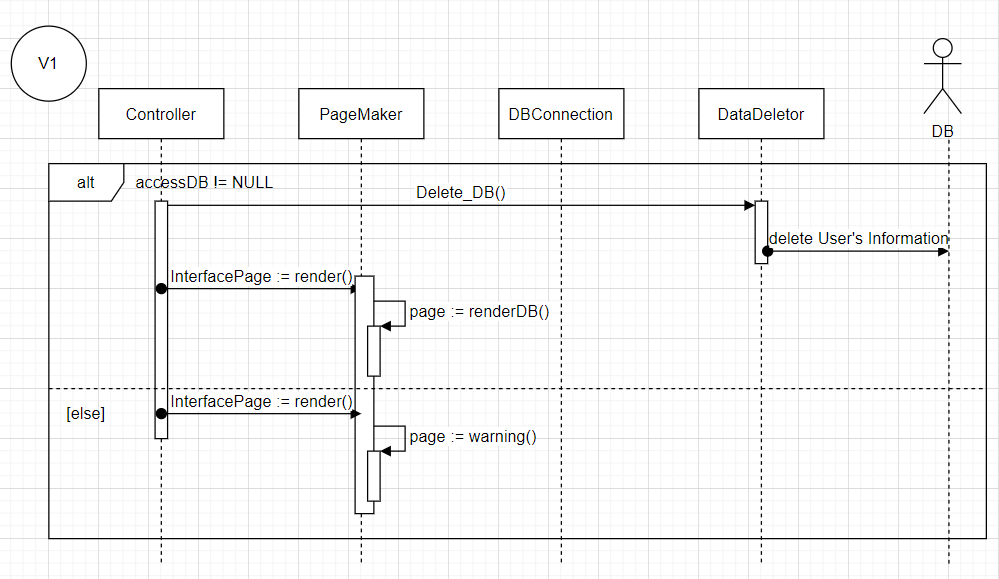


**UC-3 : DeleteUser**

**[sequence diagram]**



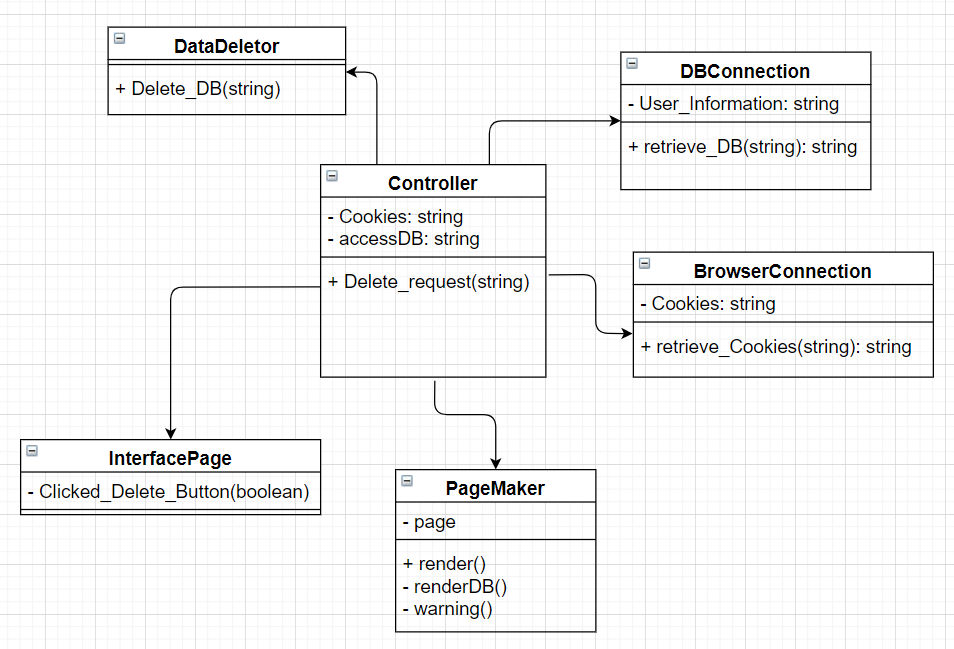
**Variation 1**



[Process]

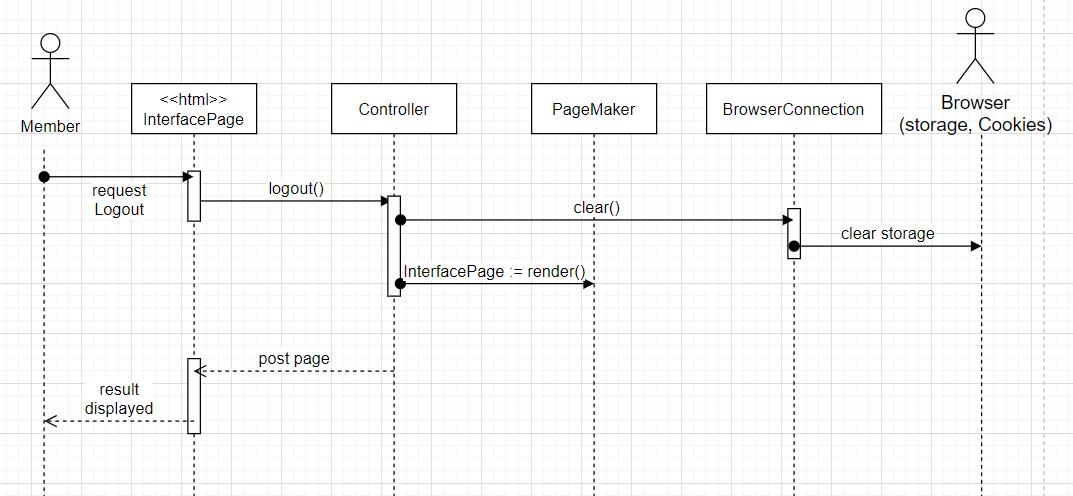
원래의 DBConnection이 data를 추출하고, 특정 data를 삭제하는 일을 맡았다면, variation1에서는 DataDeletor라는 새로운 object를 만들어 DBConnection은 data를 추출하는 일만 하고, DataDeletor는 data를 삭제하는 일만 한다. 이렇게 해서 각 object의 전문성을 높일 수 있다.

**[class diagram]**

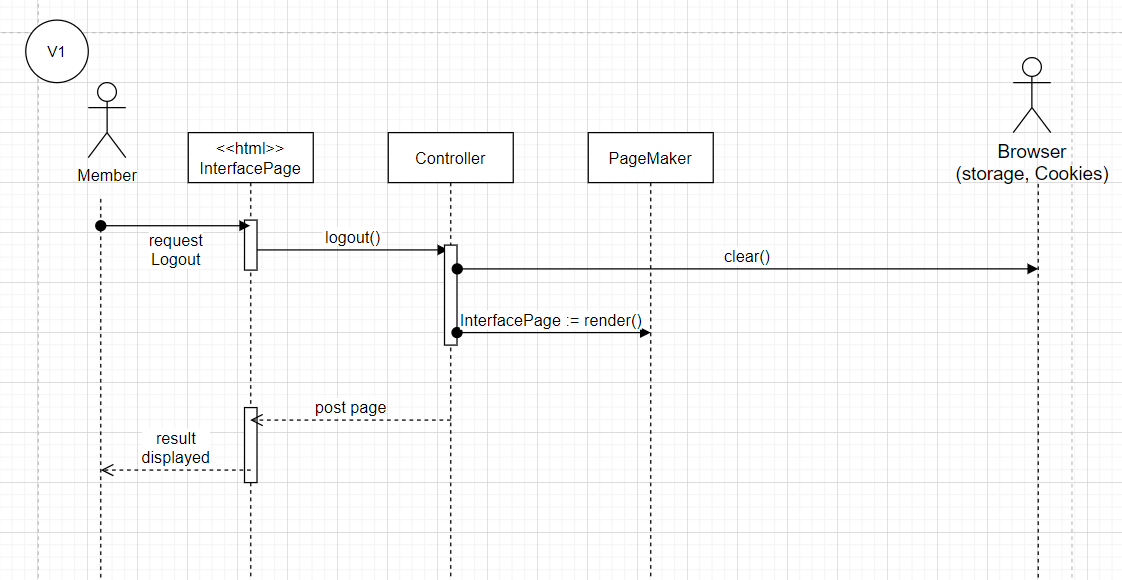


**UC-3 : Logout**

**[sequence diagram]**



**Variation 1**

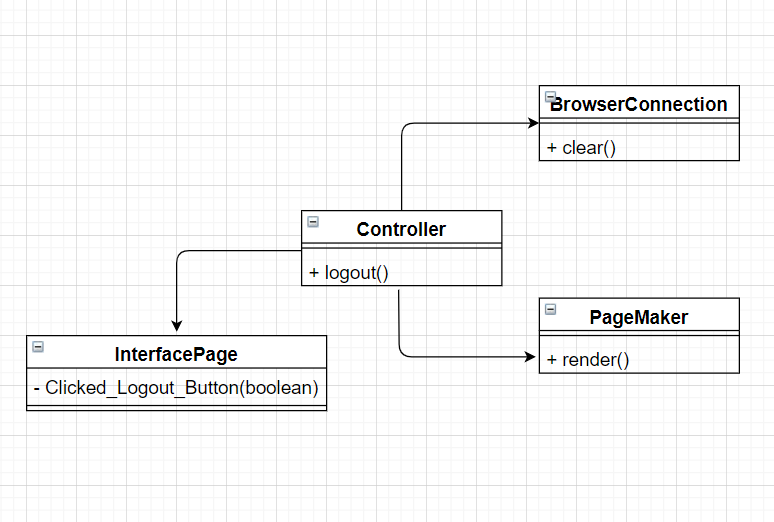


[Process]

chain을 최소화하기 위해 BrowserConnection을 제거하고 그 object의 responsibilities를 controller에게 부여한다.

(chain을 최소화했지만, controller의 전문성이 떨어지고, cohesion이 증가한다. 따라서 BrowserConnection을 제거하지 않는 것으로 결정.)

**[class diagram]**

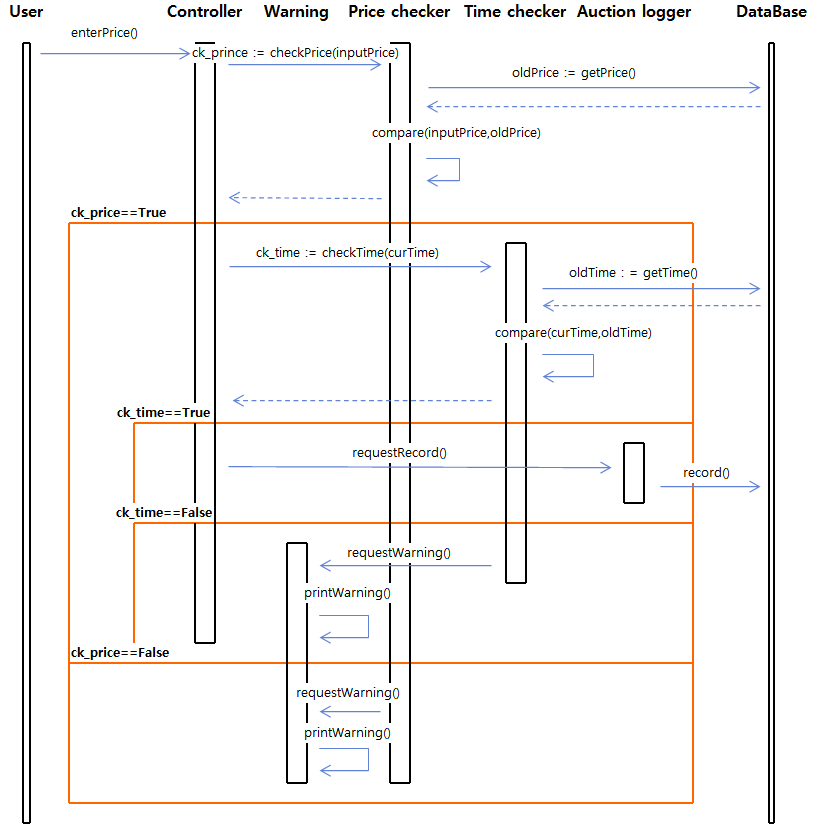


**SubGroup2**

**경매, 채팅 기능 [고재원, 최승원]**

**UC-11 : 경매하기**

**[sequence diagram]**

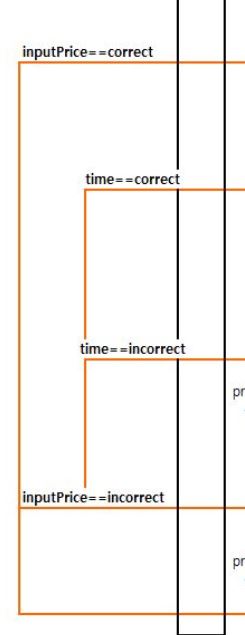


**[Variation 및 이에 대한 논의]**

Q1) Controller가 모든 sequence를 컨트롤 하는 것과 요청을 하고 나머지 sub doer가 요청을 수행해야 하는가?

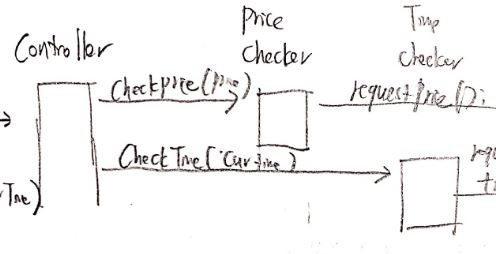
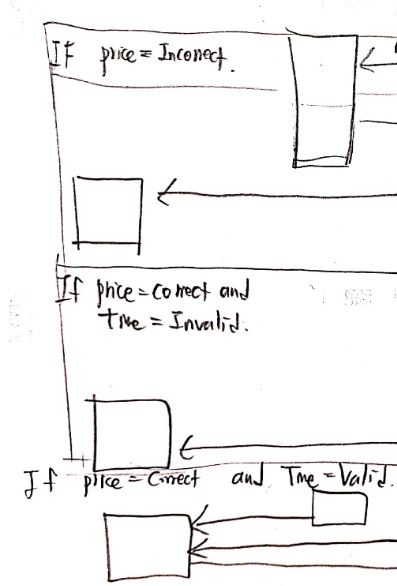
A) 논의 결과, Controller가 모든 sequence를 수행하기보다는, 각자의 역할을 수행 할 수 있는 Checker를 만들었으므로 Checker들에게 이러한 역할을 맡기는 것이 좋다고 생각하여 Controller는 그저 요청을 보낼 뿐이고, Checker들이 알아서 명령을 실행하고 결과를 도출하도록 하였다.

Q2) 경매를 할 때 요구조건에 따라 1)금액이 현재 경매 가격보다 높아야 하며, 2) 다른 사용자가 금액을 입찰한 후 몇 분 동안은 입찰을 할 수 없어야 한다. 따라서 경매 가격, 입찰 시간을 변수로 받아서 연산을 한다. 이때, 최승원 조원의 방법(**Variation** **1**)은 먼저 가격이 적당한지 검사 후, 입찰할 수 있는 시간인지 검사를 하는 반면, 고재원 조원의 경우(**Variation** **2**) 두 변수의 값이 적당한지 둘 다 검사 후 경우를 나누어 입찰 여부를 판단하였다. 이 두 방법 중 어떤 방법을 선택해야 하는가?

******(Variaton 1)**

설명 : 가격을 먼저 검사하고, 옳은 값이 맞는지 아닌 지에

따라 시간을 검사하여 입찰 여부를 결정한다.

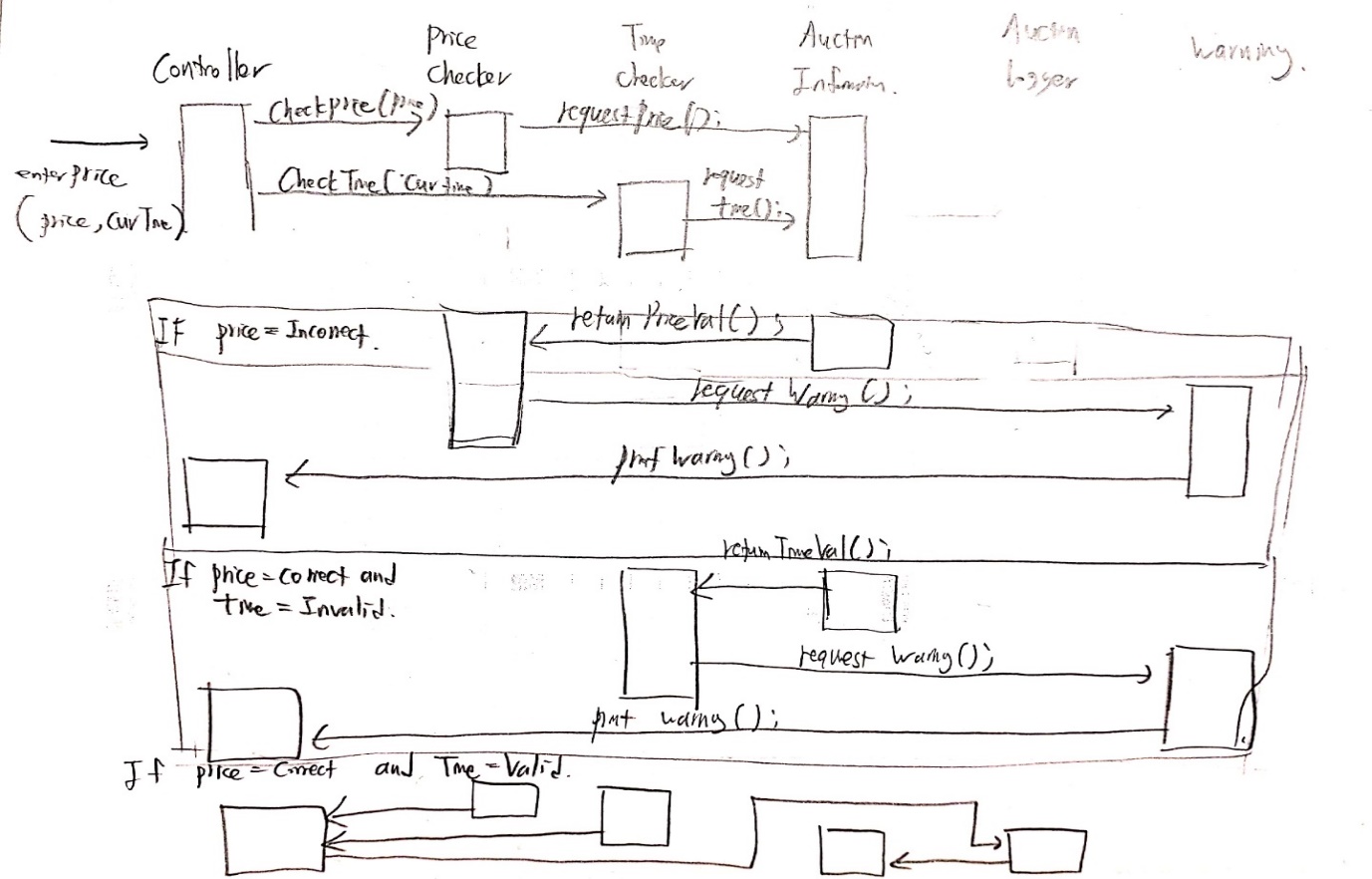
**(Variaton 2)**

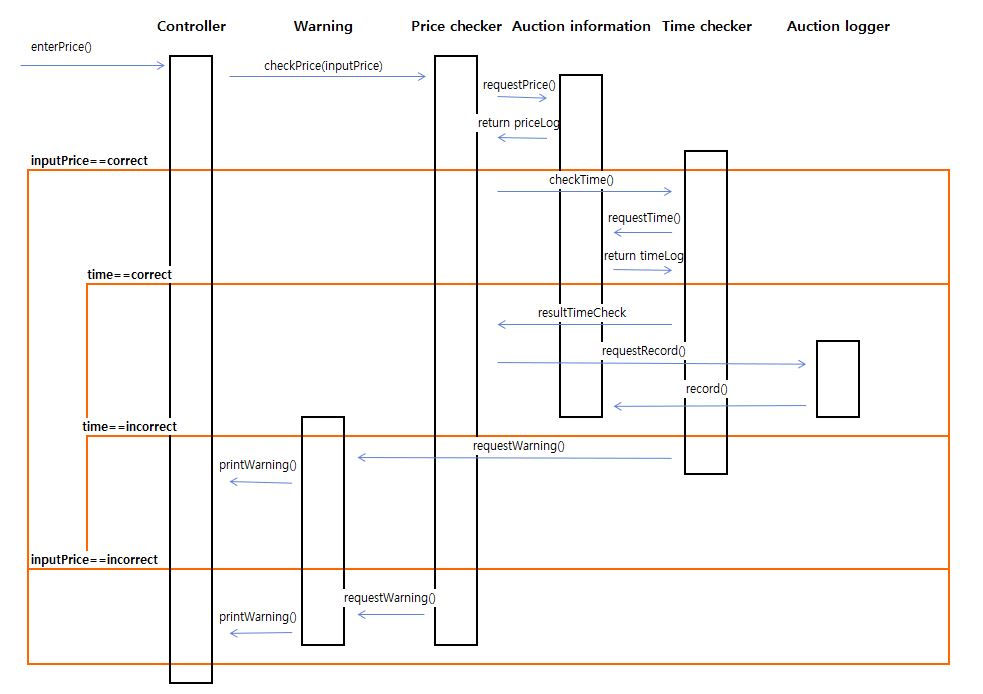
설명 : 가격과 시간을 모두 검사 후, 두 값의 옳고

그름에 따라 입찰여부를 결정한다.

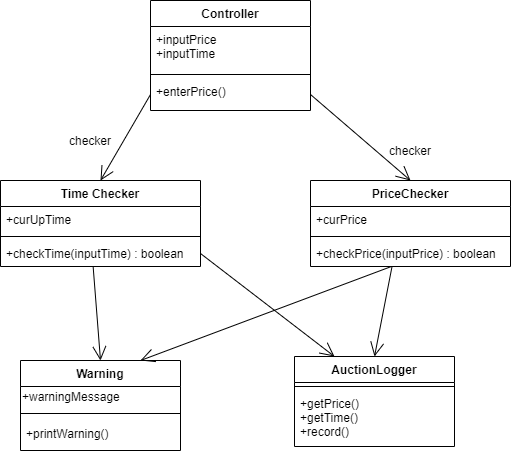
A) 논의 결과, 가격이 올바르지 않으면 입찰 시간에 관계없이 입찰이 불가하기 때문에 입찰 시간이 올바른 지 검사할 필요가 없으므로 가격을 먼저 검사하는 방법을 선택할 시, 시간 복잡도를 고려하였을 때, 시간/가격 두개를 모두 검사하는 **Variation** **2** 의 방법보다 **Variation** **1**의 방법을 채택하기로 하였다.

**[초기 작업 설계 시 UC-11 Sequence Diagram Processing]**



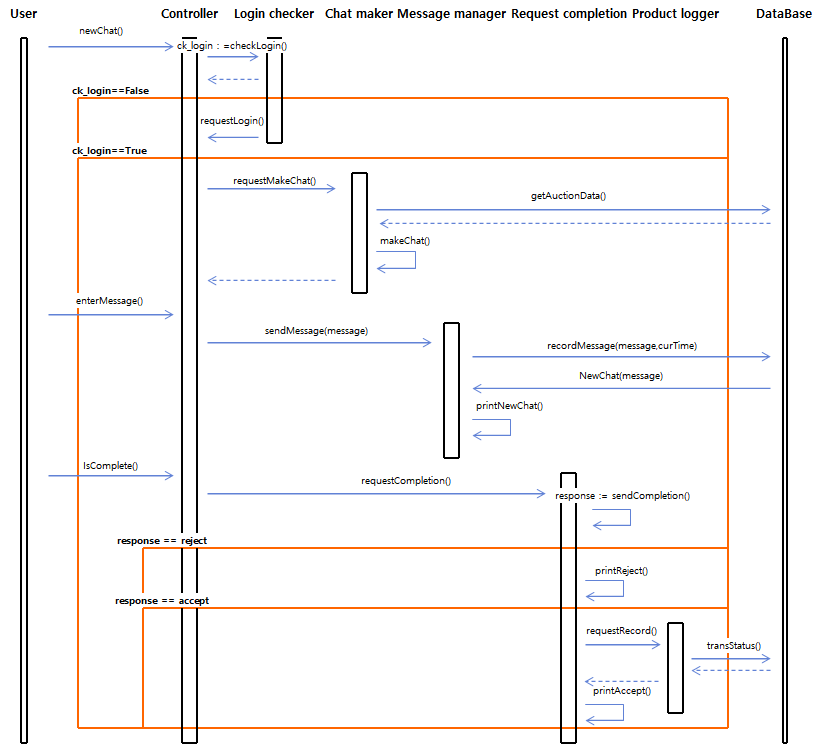


**[class diagram]**



**UC-12 : 채팅하기**

**[sequence diagram]**



**[Variation 및 이에 대한 논의]**

Q) Chat Maker, Chat Sender 등 Sequence 적으로 기능이 모호한 것 들이 있다. 또한 이름이 그것의 내용을 충분히 담지 못하는 것 같다. (ex: Auction Information)

A) 논의 결과,

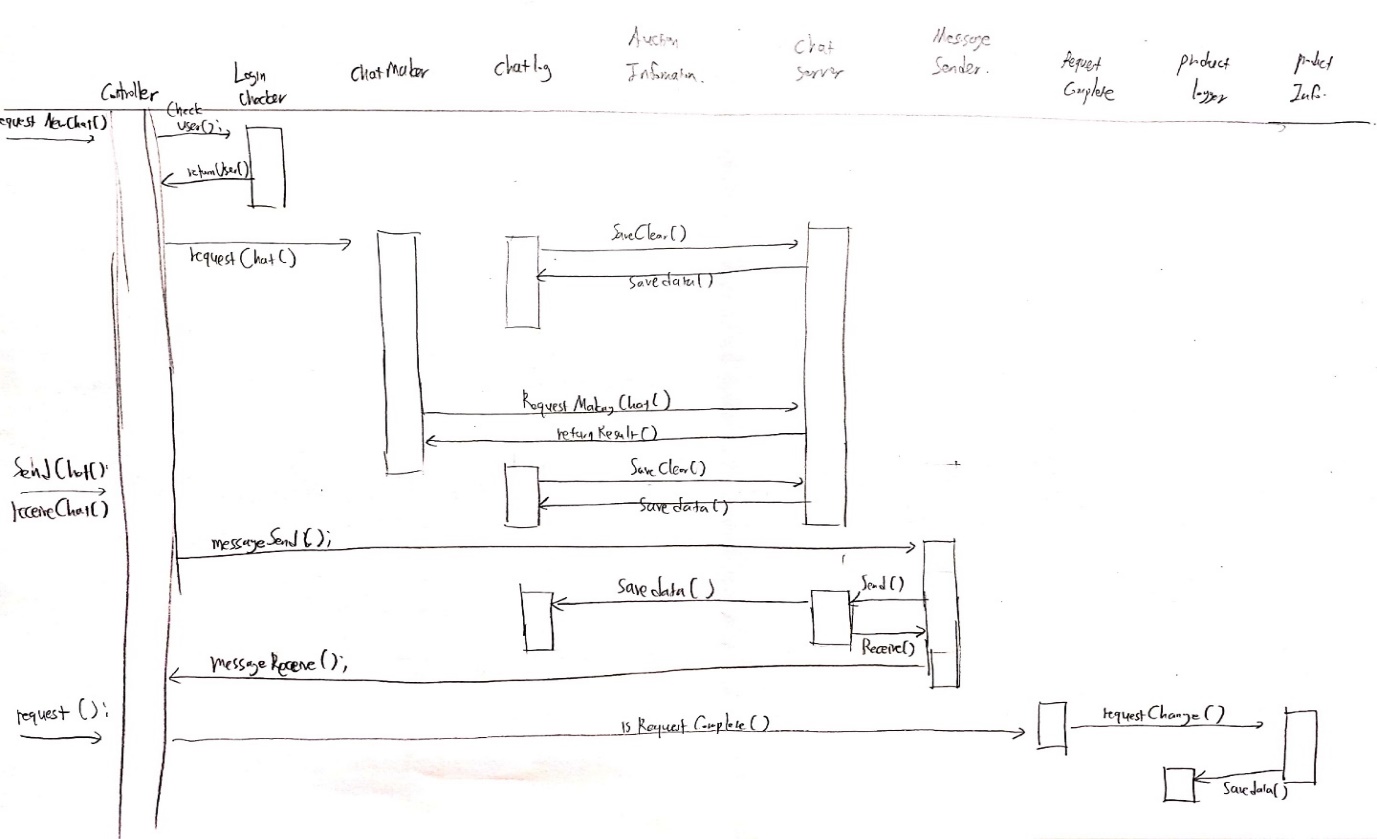
1. Chat Maker는 단지 채팅방을 개설하는 역할만을 한다.

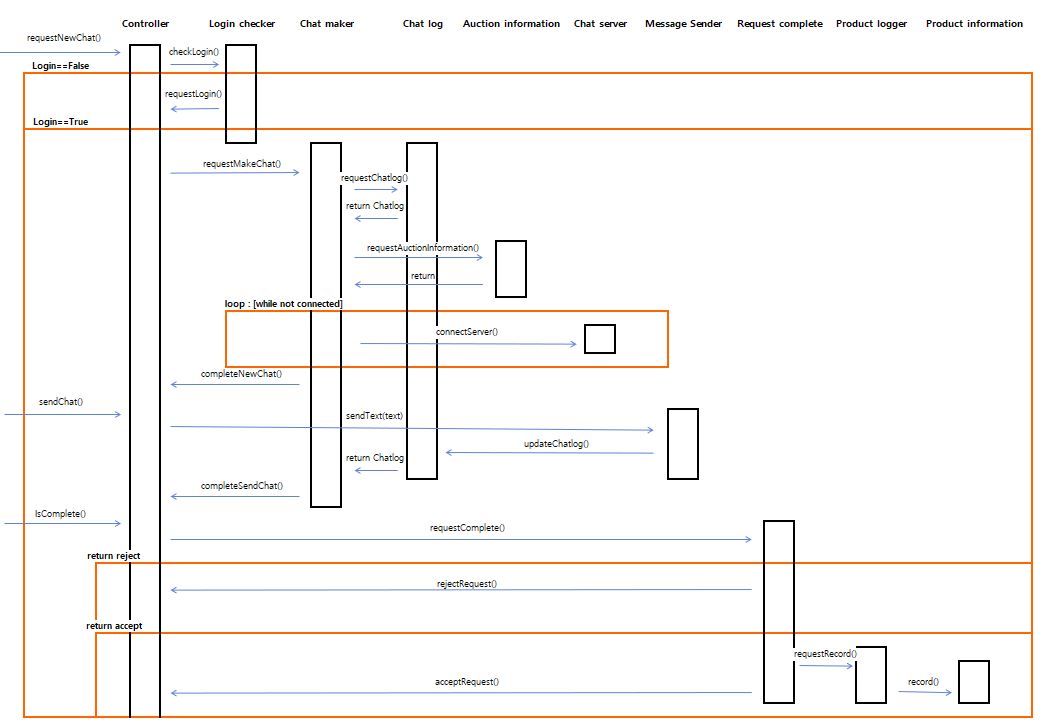
2. Chat Sender -> Chat Manager로 명칭을 바꾸고, 이것이 채팅을 보내거나(send), 받거나(receive) 하는 것을 책임지도록 한다.

3. Auction Information -> Auction DB로 명칭을 바꾸어 경매에 대한 데이터를 담고 있음을 표현할 수 있도록 한다.

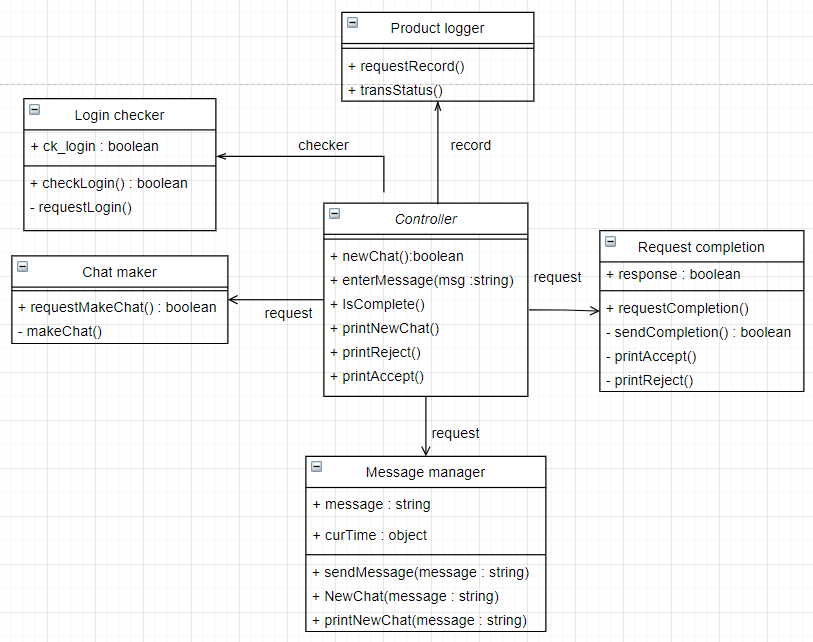
4. 추가로 기타 이름 및 함수를 변경하여 Sequence를 이해하기 쉽도록 한다.

**[초기 작업 설계 시 UC-12 Sequence Diagram Processing]**





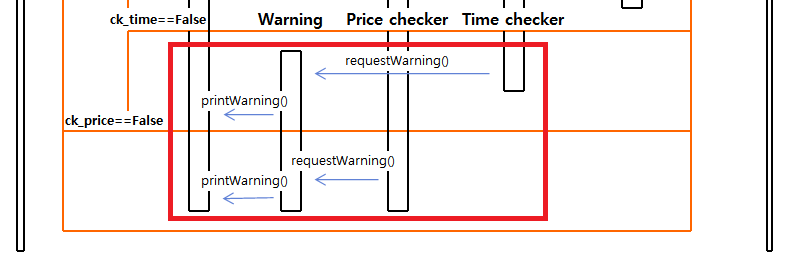
**[class diagram]**



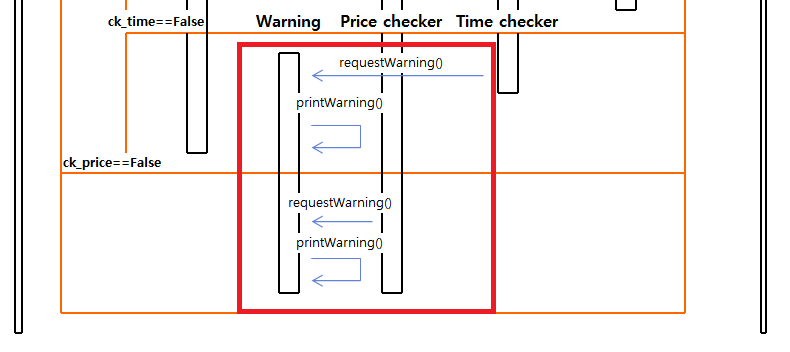
**[+ UC-11과 UC-12 모두 해당되는 Variation 및 이에 대한 논의]**

Q1) 메시지 출력의 주체 : Controller vs Warning Concept

1. Controller가 Warning에게 요청을 받아 직접 출력한다.



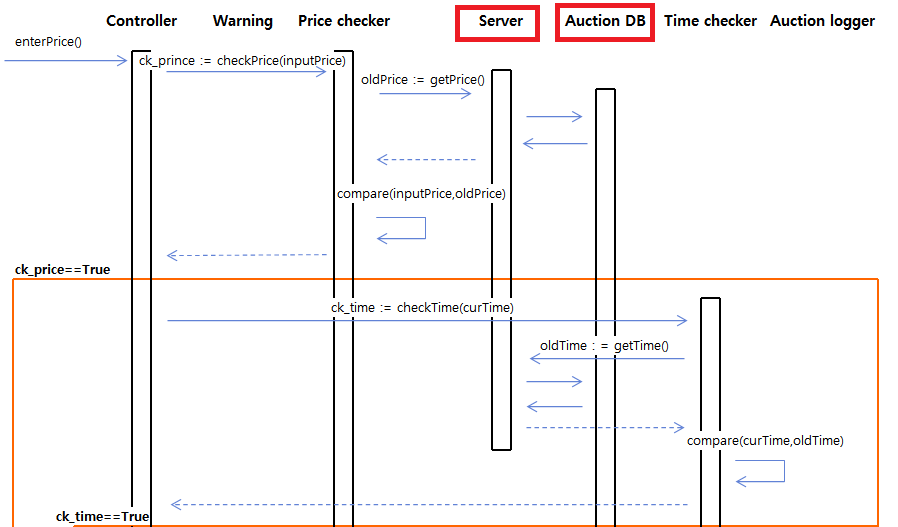
1. Warning 내부에서 출력한다. (선택)



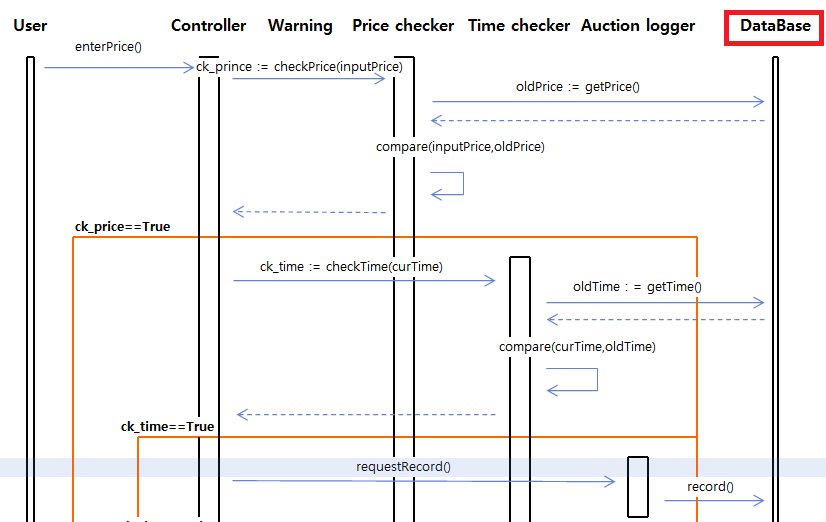
A) 논의 결과, 경고 메시지는 Controller보다는 Warning에서 맡아서 하는 것이 Concept적으로도 맞다고 생각하여 Warning 내부적으로 출력을 담당하기로 하였다. Controller는 다른 Concept들에게 요청하고 결과를 받는 역할정도로 한정 지었다. (UC-12에서 완료요청에 대한 결과 출력도 마찬가지로 Request completion에서 담당하여 출력한다.)

Q2) Server, log(DB) concept vs DB (Actor)

1) 서버와 데이터베이스를 클래스로 나타낸 경우



2) 서버를 제외시키고 데이터베이스를 Actor로 나타낸 경우 (선택)



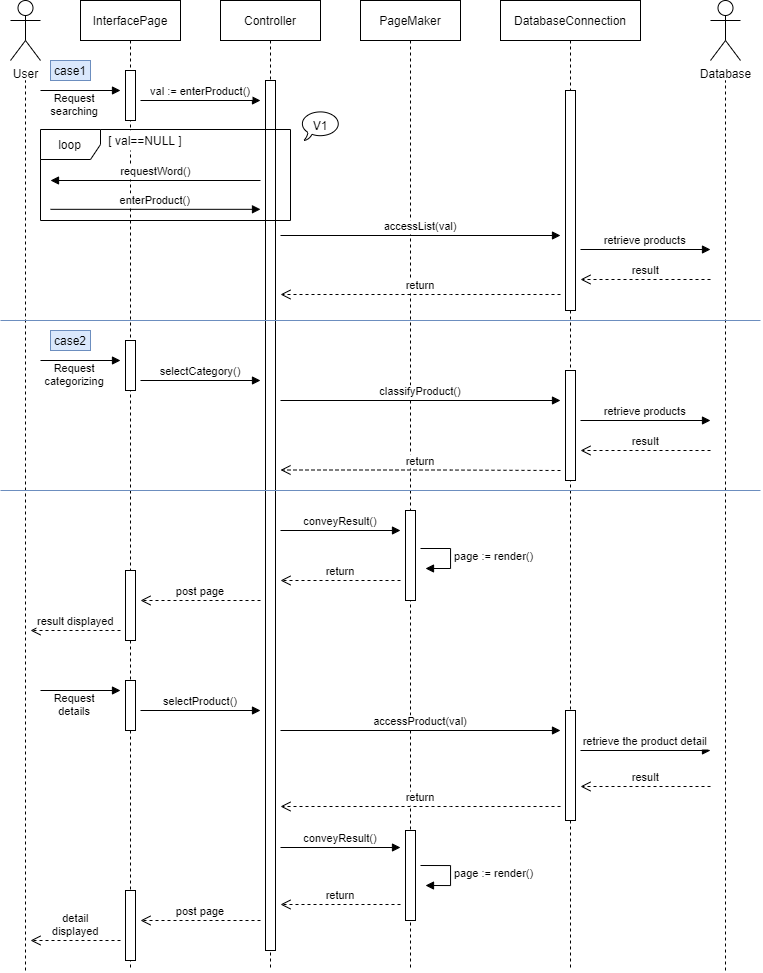
A) 논의 결과 : Class diagram을 만들면서 server, log(DB)는 하나의 클래스라기보다는 Actor에 더 가까운 역할을 수행하고 있었다. 또한 Server는 생략하였는데 DB로 접근하기 위해서는 당연히 Server를 통해야 하고 이 기능에서 Server만의 독자적인 기능이 있는 것은 아니었으며 SDD의 가독성을 떨어뜨리기 때문에 생략하였다. 따라서 설계 및 Sequence를 표현하는데 더 적합하다고 생각하여 이 Server와 log concepts를 제거하고 DB라는 Actor를 추가하였다.

**SubGroup3**

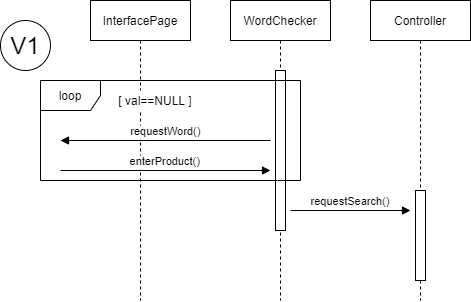
**매물검색, 관리 기능 [장동재, 전예원]**

**UC-5 : 매물 검색**

**[sequence diagram]**



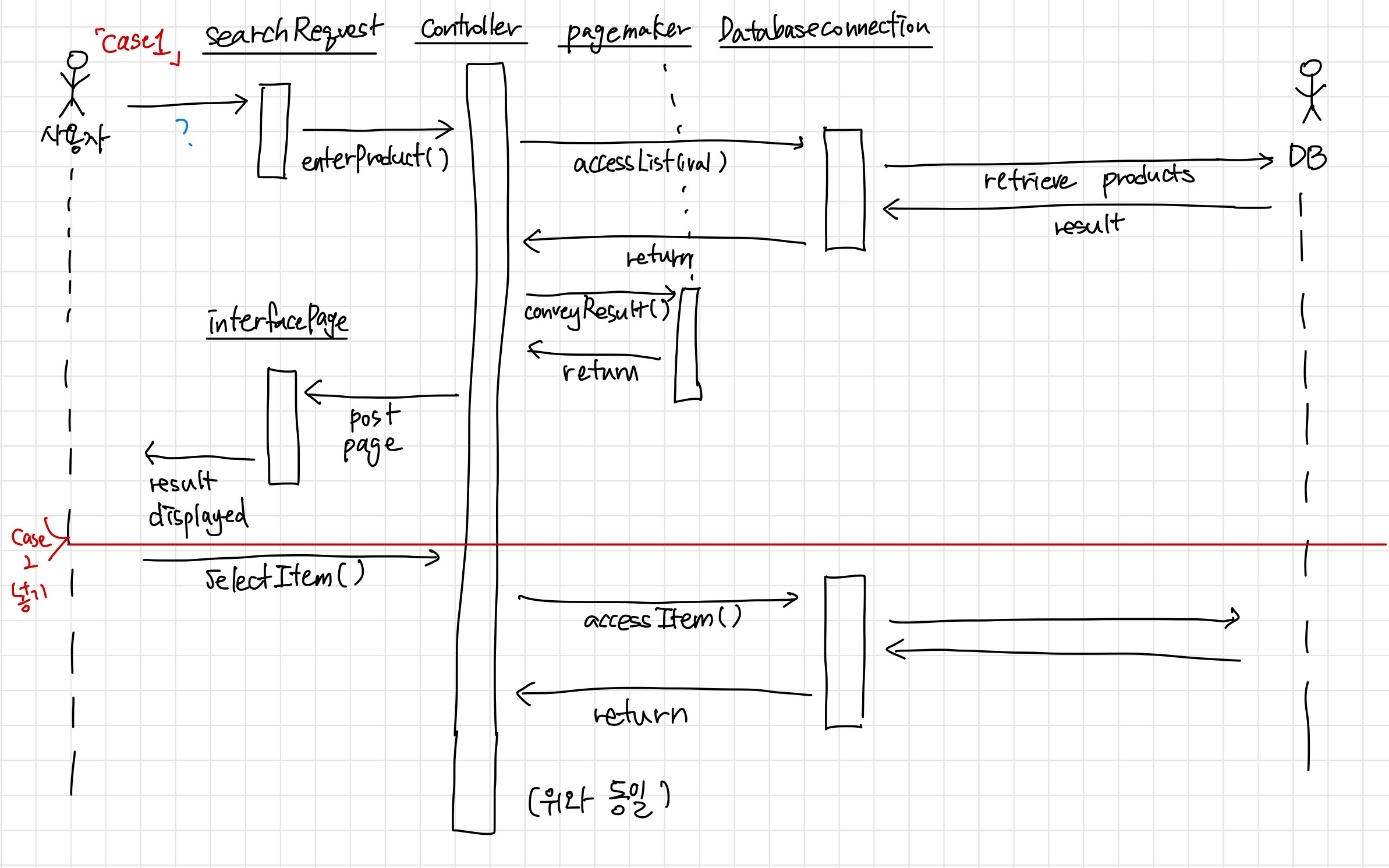
**Variation 1**

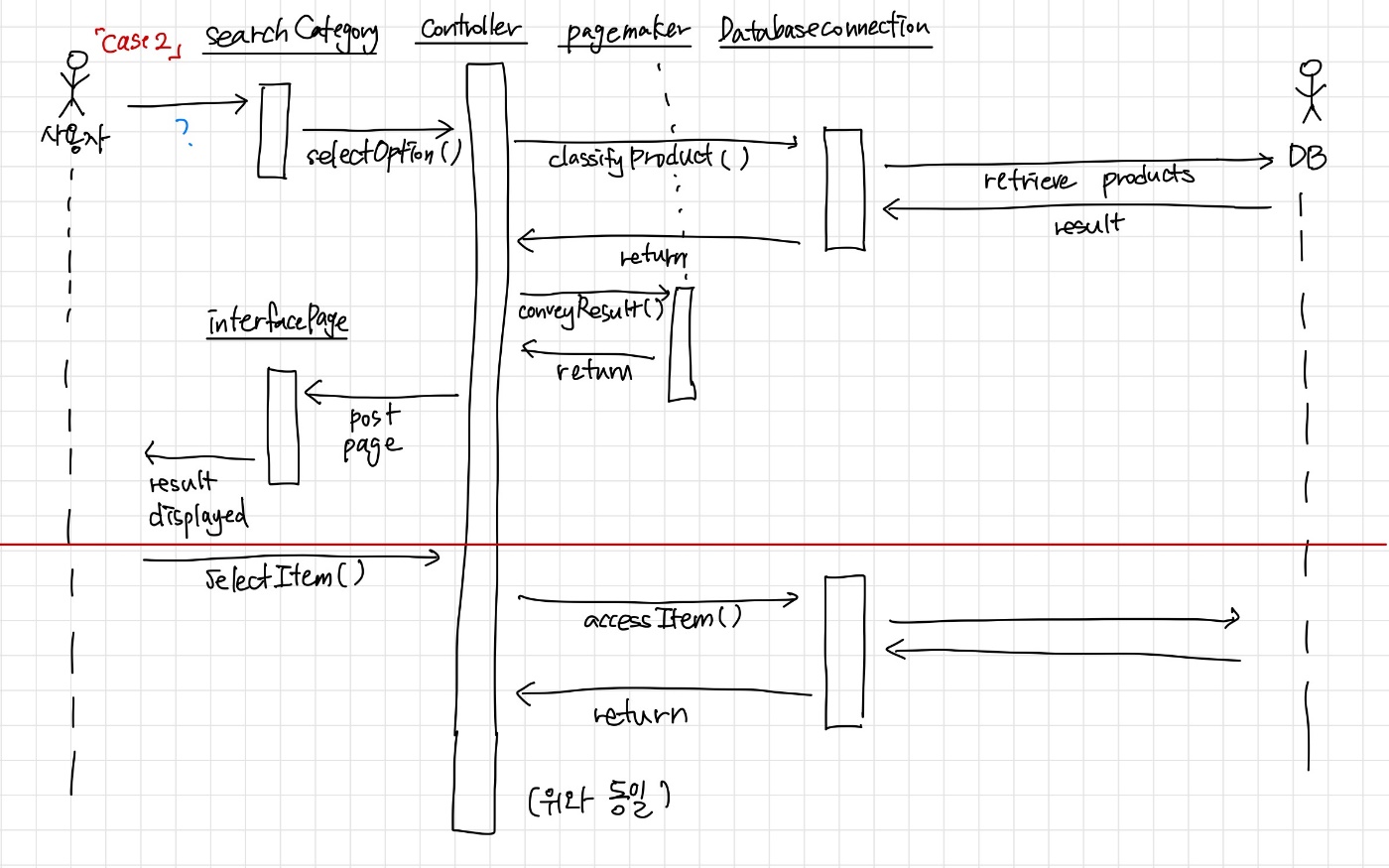


[Process]

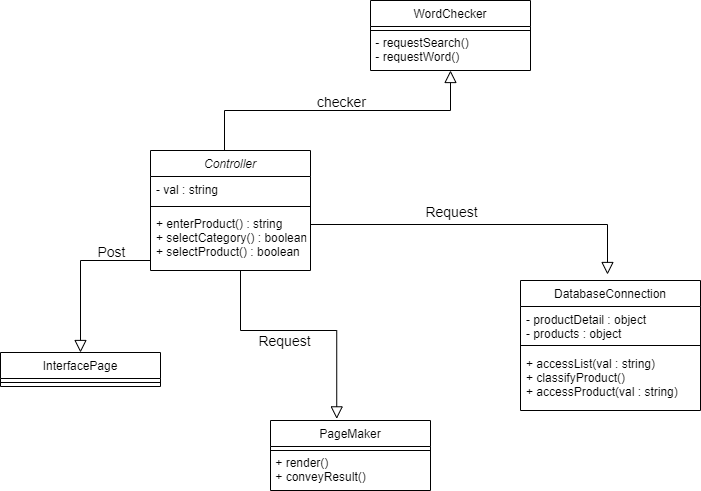
기존의 sequence diagram에서는 사용자가 입력한 val값을 Controller가 바로 받게 되어 그 값이 NULL이 되지 않을 때까지 다시 입력값을 요청하게 되어 있다. Variation1에서는 Controller에게 할당되는 일을 줄여줄 수 있게 하기 위해 WordChecker라는 새로운 Class를 Controller 이전에 추가한다. val값이 NULL일 경우 Controller에게 전달하기 전에 WordChecker가 새로운 val을 사용자에게 요청하도록 하고, val값이 생긴 경우에 Controller에게 다음 명령을 요청한다.

**[초기 작업 설계 시 UC-11 Sequence Diagram Processing]**



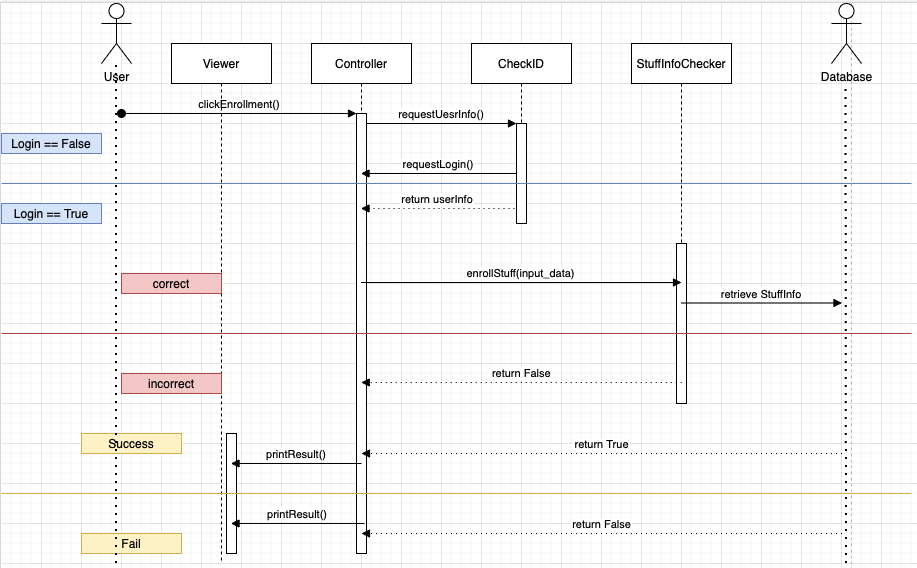


**[class diagram]**



**UC-7 : 매물 등록**

**[sequence diagram]**



**[class diagram]**

