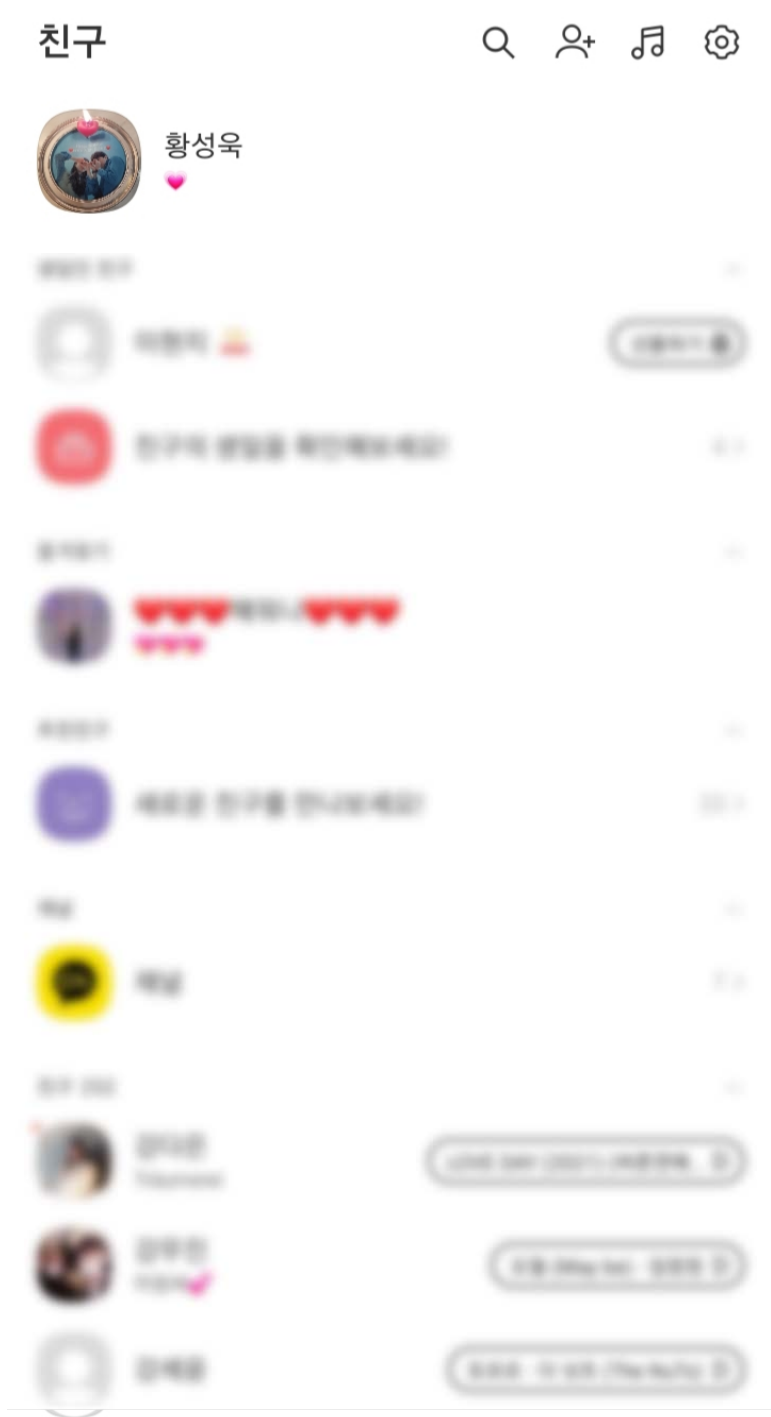
카카오톡
친구 기능 UML

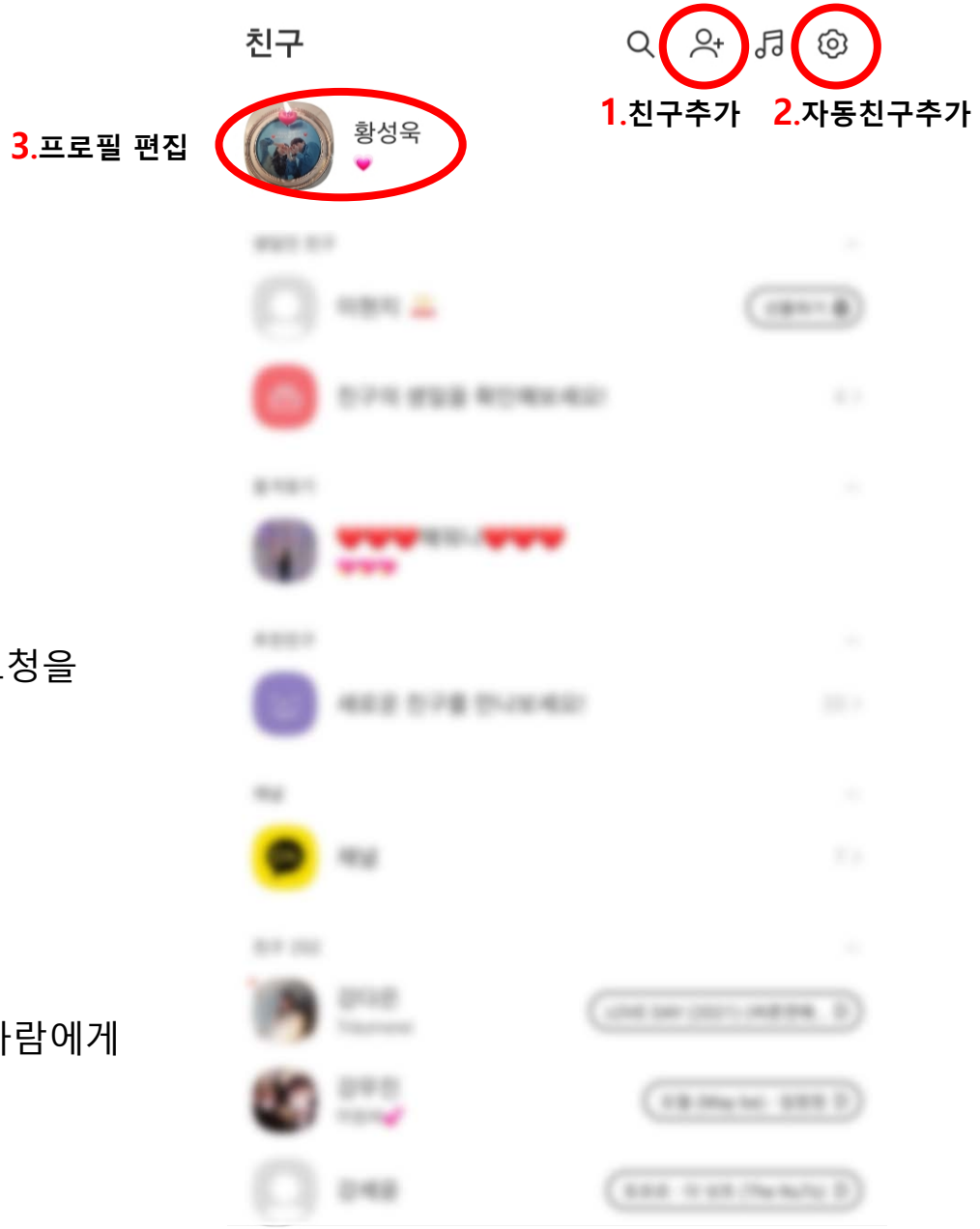
20163091 황성욱
20170020 오석원
20173045 김성욱
20193084 이현지



목차

유스케이스 다이어그램

1. 시스템 상황 분석



(1) 친구 추가

- 이름과 전화번호를 통해 친구를 추가한다.

(2) 자동 친구 추가

- 연락처에 저장된 친구와 카카오톡 친구목록을 동기화한다.
- 자동으로 친구를 추가할 수 있게 설정해 놓으면 사용자가 버튼을 누르는 등 직접 요청을 하지 않아도 친구가 추가된다.
- 연락처에 새로운 전화번호를 저장한 지 얼마 되지 않아 아직 카카오톡 친구목록에 반영이 되지 않았다면 요청을 통해 친구를 추가할 수 있다.

(3) 프로필 편집

- 자신의 프로필 정보(이름, 상태메시지, 프로필사진, 배경 사진)을 편집하여 다른 사람에게 보이게 한다.

1. 시스템 상황 분석

(4) 친구 숨김

- 친구목록에서 보고 싶지 않은 친구가 있다면, 친구 숨김 기능을 통해 보이지 않도록 할 수 있다.
- 친구 숨김을 하면 해당 친구는 친구 숨김 목록으로 이동한다.
- 친구 숨김 목록에서 숨김시간을 예약할수 있다.

(5) 친구 차단

- 친구를 차단 목록에 추가하여 친구목록에서 삭제한다.
- 친구를 차단하면 메시지가 오지 않는다.

(6) 친구목록복구

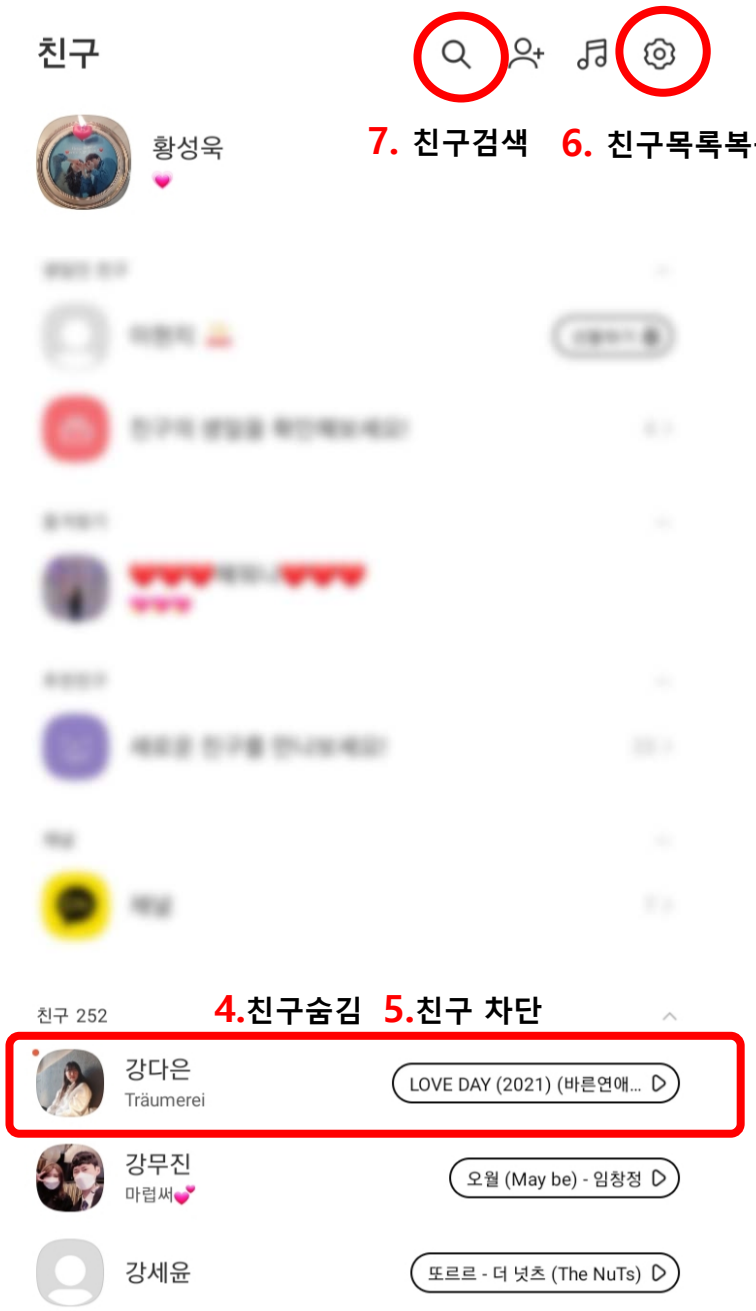
- 차단, 숨김 된 친구목록을 다시 복구한다.

(7) 친구 검색

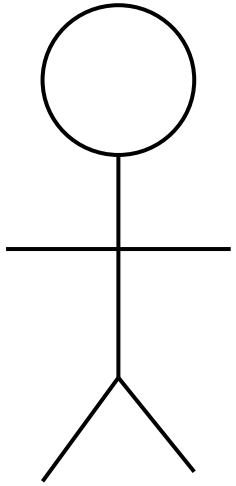
- 친구의 이름이나 연락처를 통해 친구를 검색할 수 있다.

(8) 친구 목록 그룹화

- 친구목록을 그룹화하여 관리할 수 있다.



2. 액터 분석



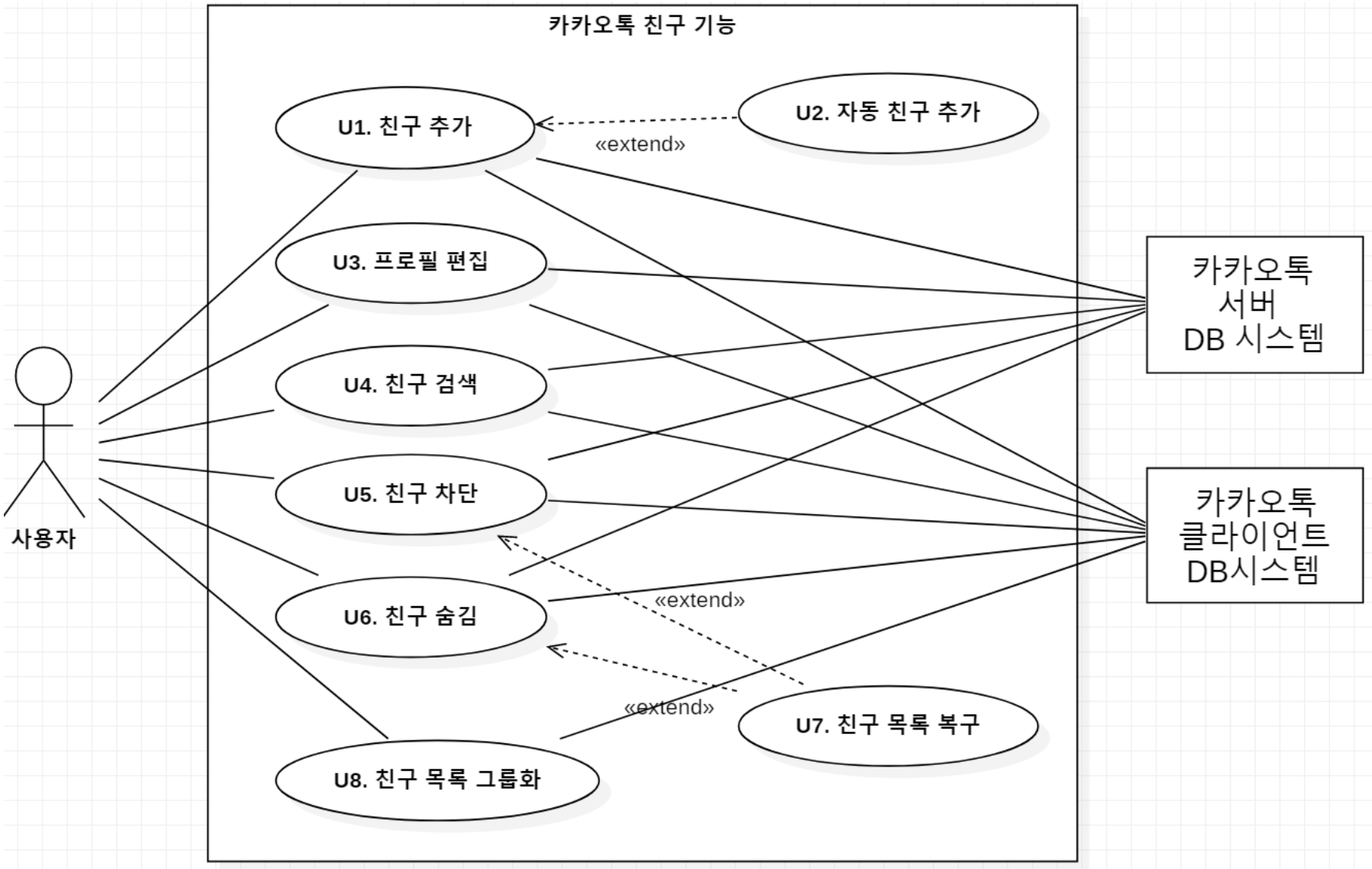
사용자

카카오톡
클라이언트
DB 시스템

카카오톡 서버
DB시스템

3. 유스케이스 식별

4. 유스케이스 다이어그램



5. 유스케이스 명세서

✓U1. 친구 추가

- **액터명** : 사용자, 카카오톡DB 서버 시스템, 카카오톡DB 클라이언트 시스템
- **유스케이스 개요** : 사용자는 이름과 연락처를 입력하여 친구를 추가한다.
- **사전 조건** - 사용자가 카카오톡에 로그인 되어있어야 한다.
- **이벤트 흐름**

➢ *정상 흐름*

1) 친구 추가를 요청한다.(액터)
2) 이름, 연락처 입력을 요구한다.(시스템)
3) 이름, 연락처를 입력한다.(액터)
4) 카카오톡 전체 목록에서 해당연락처를 가진 상대를 검색하여 확인한다.(시스템)
5) 찾은 상대를 친구 추가한다.(시스템)

➢ *선택 흐름*

- 본인 이름과 연락처를 입력할 경우 "나 자신은 영원한 인생의 친구입니다." 메시지를 보여주고 친구가 추가되지 않는다.
- 존재하지 않는 연락처를 입력할 경우 "유효하지 않은 전화번호입니다. 입력하신 번호를 다시 한 번 확인해 주세요." 메시지를 보여주고 친구가 추가되지 않는다.
- 최대 친구 수를 넘으면 "현재 친구 수가 최대 친구 수 15000명을 넘어 더 이상 추가할 수 없습니다." 메시지를 보여주고 친구가 추가되지 않는다.

5. 유스케이스 명세서

✓ U2. 자동친구추가

- **액터명** : 사용자, 카카오톡DB 서버 시스템, 카카오톡DB 클라이언트 시스템
- **유스케이스 개요** : 휴대폰 연락처에 저장된 정보를 토대로 카카오톡 친구 목록에 불러온다.
- **사전 조건**

- 카카오톡에 로그인 되어있어야 한다.
- 연락처에 새로운 전화번호를 저장했을 때 아직 카카오톡 친구 목록에 동기화가 되지않은 상태여야 한다.

- **이벤트 흐름**

➢ *정상 흐름*

1) 자동친구추가를 요청한다.(액터)
2) 카카오톡 DB 시스템에 저장되어 있던 휴대폰 연락처 목록을 불러온다.(시스템)
3) 카카오톡 친구 목록과 휴대폰 연락처 목록을 비교한다.(시스템)
4) 카카오톡 친구 목록과 휴대폰 연락처 목록을 동기화한다.(시스템)

➢ *선택 흐름*

▪ 없음

5. 유스케이스 명세서

✓ U3. 프로필 편집

- **액터명** : 사용자, 카카오톡DB 서버 시스템, 카카오톡DB 클라이언트 시스템
- **유스케이스 개요** : 사용자의 프로필(이름, 상태메세지, 프로필사진, 배경사진)을 편집한다.
- **사전 조건** - 사용자의 프로필이어야 한다.
- **이벤트 흐름**

➢ *정상 흐름*

1) 나를 포함하는 출력된 친구목록에서 나의 프로필을 조회한다.(액터)
2) 프로필 조회를 종료할지 프로필 편집을 할 것인지 요청한다.(시스템)
3) 프로필 편집 기능을 실행한다.(사용자)
4) 프로필 편집 항목(이름, 상태메시지, 프로필 사진, 배경 사진) 수정을 요청한다.(시스템)
5) 해당 편집 항목을 수정하고 저장한다.(사용자)
6) 시스템에 수정된 사용자 프로필 정보를 저장 후 성공했다는 메시지를 출력 후 종료한다.(시스템)

➢ *선택 흐름*

- 1)에서 친구의 프로필을 보는 경우에는 친구 프로필을 출력 후, 프로필 조회 종료 요청만 받는다.
- 2)에서 종료를 선택한 경우에는 프로필 조회를 종료한다.
- 6)에서 시스템 연결이 안되는 경우에는 "데이터 로드를 하는데 실패하였습니다"라는 메시지를 출력한다.

5. 유스케이스 명세서

✓ U4. 친구 검색

- **액터명** : 사용자, 카카오톡DB 서버 시스템, 카카오톡DB 클라이언트 시스템
- **유스케이스 개요** : 친구정보(이름, 전화번호)를 통해 친구목록에서 친구를 검색하여 찾을 수 있다.
- **사전 조건** - 친구 목록에 포함된 친구만 검색이 된다.
- **이벤트 흐름**

➢ *정상 흐름*

1) 친구검색 기능을 실행한다.(액터)
2) 검색어 입력을 요청한다.(시스템)
3) 검색어를 입력한다.(시스템)
4) 서버를 통해 친구목록에서 해당 검색어가 친구의 이름, 전화번호에 포함된 친구들을 찾아 모두 출력한다.

➢ *선택 흐름*

▪ 6)에서 시스템 연결이 안되는 경우에는 "데이터 로드를 하는데 실패하였습니다"라는 메시지를 출력한다.

5. 유스케이스 명세서

✓ U5. 친구 차단

- **액터명** : 사용자, 카카오톡DB 서버 시스템, 카카오톡DB 클라이언트 시스템
- **유스케이스 개요** : 사용자가 친구를 차단한다.
- **사전 조건** - 친구목록에 친구가 추가되어 있어야 한다.
- **이벤트 흐름**

➢ *정상 흐름*

1) 친구목록에서 차단할 친구를 선택한다.(사용자)
2) 나 자신이 아닌 친구라면 진짜 차단할 것인지 취소를 할 것인지 물음을 요청한다.(시스템)
3) 차단한다고 응답한다.(사용자)
4) 시스템에 연결하여 해당 친구를 차단목록에 추가하고 해당 친구가 나에게 메시지를 못오도록 설정한 후 숨김 성공 메시지를 출력한다.(시스템)

➢ *선택 흐름*

- 2)에서 나를 선택하면 친구 숨김 기능이 실행되지 않는다.
- 3)에서 취소를 할 것이라고 응답을 하면 실행을 종료한다.
- 4)에서 시스템 연결이 안되는 경우에는 "데이터 로드를 하는데 실패하였습니다"라는 메시지를 출력한다.

5. 유스케이스 명세서

✓ U6. 친구 숨김

- **액터명** : 사용자, 카카오톡DB 서버 시스템, 카카오톡DB 클라이언트 시스템
- **유스케이스 개요** : 친구목록에서 친구를 숨길 수 있다.
- **사전 조건** - 친구 목록에 친구가 추가되어 있어야 한다.
- **이벤트 흐름**

➢ *정상 흐름*

1) 친구목록에서 숨길 친구를 선택한다.(사용자)
2) 나 자신이 아닌 친구라면 진짜 숨길 것인지 취소를 할 것인지 물음을 요청한다.(시스템)
3) 숨긴다고 응답한다.(사용자)
4) DB 클라이언트 시스템에 연결하여 해당 친구를 숨김목록에 추가하고 숨김 성공 메시지를 출력한다.(시스템)

➢ *선택 흐름*

- 2)에서 나를 선택하면 친구 숨김 기능이 실행되지 않는다.
- 3)에서 취소를 할 것이라고 응답을 하면 실행을 종료한다.
- 4)에서 DB 클라이언트 시스템에 안되는 경우에는 "데이터 로드를 하는데 실패하였습니다"라는 메시지를 출력한다.

5. 유스케이스 명세서

✓ U7. 친구 목록 복구

- **액터명** : 사용자, 카카오톡DB 서버 시스템, 카카오톡DB 클라이언트 시스템
- **유스케이스 개요** : 차단, 숨김 된 친구목록을 다시 복구한다.
- **사전 조건**

-친구가 차단되어 있어야 한다.

-친구가 숨김 되어 있어야 한다.

- **이벤트 흐름**

➢ *정상 흐름*

1) 목록에서 복구할 친구를 요청한다.(액터)
2) 요청된 친구를 복구한다.(시스템)

➢ *선택 흐름*

▪ 숨김목록 복구시 복구, 차단, 삭제 중 선택할수 있다.

5. 유스케이스 명세서

✓ U8. 친구 목록 그룹화

- **액터명** : 사용자, 카카오톡DB 클라이언트 시스템
- **유스케이스 개요** : 친구목록을 폴더별로 나누어 그룹화하여 관리한다
- **사전조건**

-친구목록에 추가 되어있어야 한다.

- **이벤트 흐름**

➢ *정상 흐름*

1) 폴더명을 정하여 폴더를 생성한다.(액터)
2) 목록에서 그룹화할 친구를 요청한다.(액터)

➢ *선택 흐름*

- 한 폴더에 추가되어 있는 친구는 동일한 폴더에 추가하려고 할 때 "이미 폴더안에 추가된 친구입니다"라는 알림창이 뜨고 친구를 추가할 수 없다.
- 폴더명은 중복이 되지 않는다.

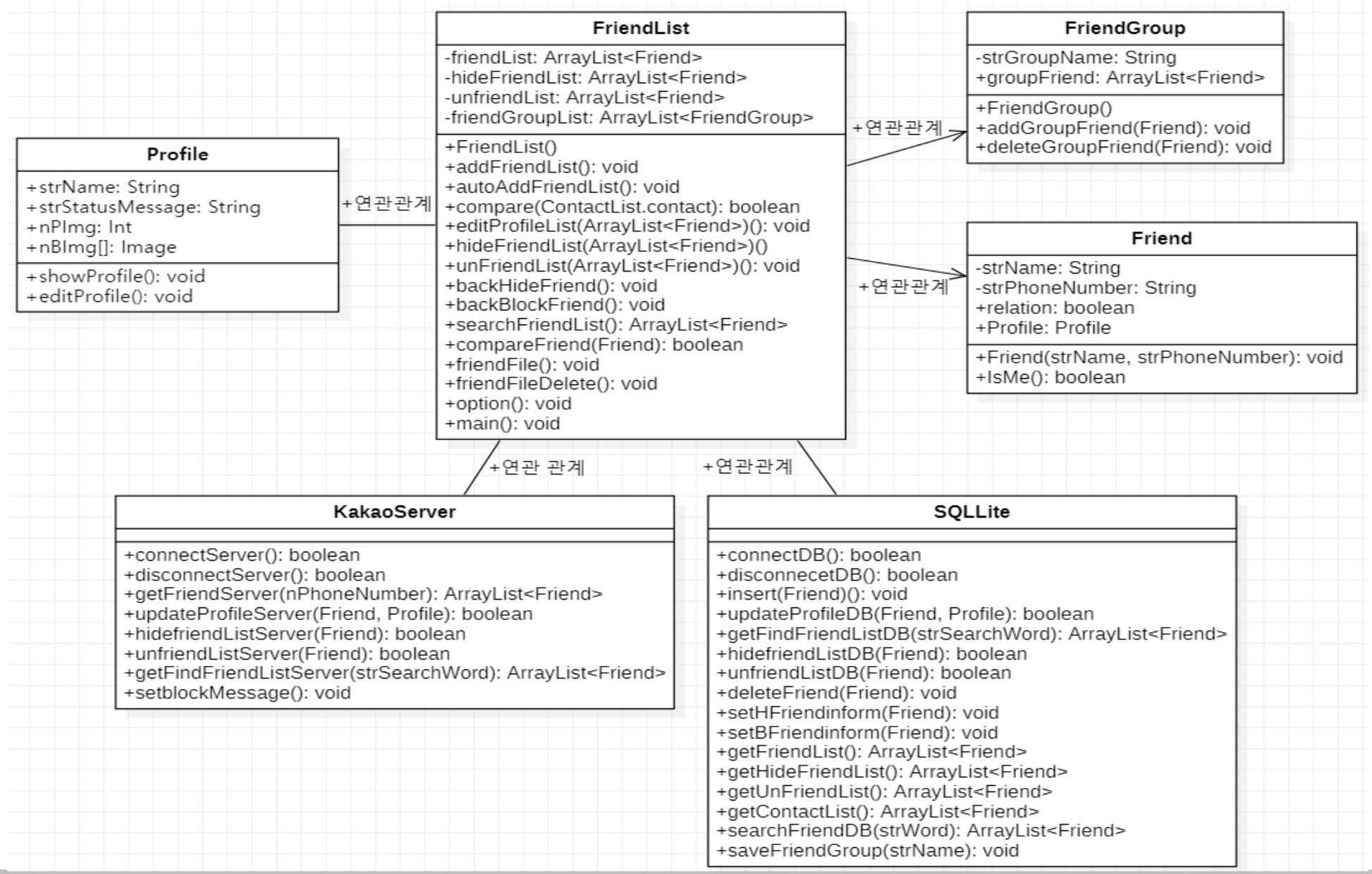
클래스 다이어그램

1. 클래스 & 메서드 추출

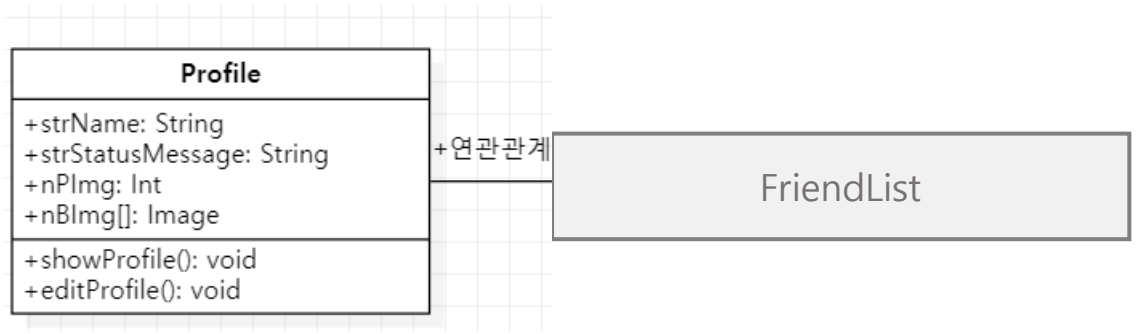
클래스 추출	메서드 추출		
Friend (친구) FriendList (친구 목록) FriendProfile (친구 프로필) MyProfile(본인 프로필) SQLLite (카카오톡 클라이언트 DB시스템) KakaoServer (카카오서버 DB시스템) FriendGroup (친구그룹목록)	showProfile(프로필출력) editProfile(프로필편집) addFriendList(친구추가) autoAddFriendList (자동친구추가) compare(연락처 비교) hideFriendList(친구숨김) unFriendList(친구차단) backHideFriend(숨김친구복구) backBlockFriend(차단친구복구) searchFriendList(친구목록검색) compareFriend (친구그룹목록비교) saveFriendGroup(그룹저장) friendFile(그룹목록생성)	friendFileDelete(그룹목록삭제) addGroupFriend(그룹친구추가) deleteGroupFriend(그룹친구삭제) IsMe(프로필구분) connectServer(서버연결) getFriendServer (서버데이터불러오기)' updateProfileServer (프로필정보 서버 업데이트) getFindFriendListServer (서버에서 친구 검색) hideUserProfile(친구숨김요청) option(옵션요청) setHFriendinform(친구숨김정보)	setBFriendinform(차단숨김정보) deleteFriend(친구삭제) connectDB(연결) Insert(친구추가) getFriendList(친구목록요청) getHideFriendList(숨김목록요청) getUnFriendList(차단목록요청) getContactList(연락처요청) getFindFriendListDB(친구검색DB) disconnectDB(DB연결끊기) disconnectServer(서버연결끊기) unfriendListServer(차단친구전달) setBlockMessage(메시지 차단) hidefriendListServer(숨김친구전달) hidefriendListDB(숨김친구DB전달)

추상화	객체추출	인터페이스 추출
Profile (프로필)	friendList (친구 목록) hideFriendList (숨김 친구 목록) unfriendList (차단 친구 목록) friendGrouplist(친구 그룹화) groupFriend(친구그룹화선택)	X

2. 클래스 다이어그램


FriendList
-friendList: ArrayList<Friend>
-hideFriendList: ArrayList<Friend>
-unfriendList: ArrayList<Friend>
-friendGroupList: ArrayList<FriendGroup>
+FriendList()
+addFriendList(): void
+autoAddFriendList(): void
+compare(ContactList.contact): boolean
+editProfileList(ArrayList<Friend>)(): void
+hideFriendList(ArrayList<Friend>)()
+unFriendList(ArrayList<Friend>)(): void
+backHideFriend(): void
+backBlockFriend(): void
+searchFriendList(): ArrayList<Friend>
+compareFriend(Friend): boolean
+friendFile(): void
+friendFileDelete(): void
+option(): void
+main(): void
FriendGroup
-strGroupName: String
+groupFriend: ArrayList<Friend>
+FriendGroup()
+addGroupFriend(Friend): void
+deleteGroupFriend(Friend): void
+연관관계
Profile
+strName: String
+strStatusMessage: String
+nPImg: Int
+nBImg[]: Image
+showProfile(): void
+editProfile(): void
+연관관계
Friend
-strName: String
-strPhoneNumber: String
+relation: boolean
+Profile: Profile
+Friend(strName, strPhoneNumber): void
+IsMe(): boolean
+연관관계
+연관 관계
+연관관계
KakaoServer
+connectServer(): boolean
+disconnectServer(): boolean
+getFriendServer(nPhoneNumber): ArrayList<Friend>
+updateProfileServer(Friend, Profile): boolean
+hidefriendListServer(Friend): boolean
+unfriendListServer(Friend): boolean
+getFindFriendListServer(strSearchWord): ArrayList<Friend>
+setblockMessage(): void
SQLLite
+connectDB(): boolean
+disconnecetDB(): boolean
+insert(Friend)(): void
+updateProfileDB(Friend, Profile): boolean
+getFindFriendListDB(strSearchWord): ArrayList<Friend>
+hidefriendListDB(Friend): boolean
+unfriendListDB(Friend): boolean
+deleteFriend(Friend): void
+setHFriendinform(Friend): void
+setBFriendinform(Friend): void
+getFriendList(): ArrayList<Friend>
+getHideFriendList(): ArrayList<Friend>
+getUnFriendList(): ArrayList<Friend>
+getContactList(): ArrayList<Friend>
+searchFriendDB(strWord): ArrayList<Friend>
+saveFriendGroup(strName): void

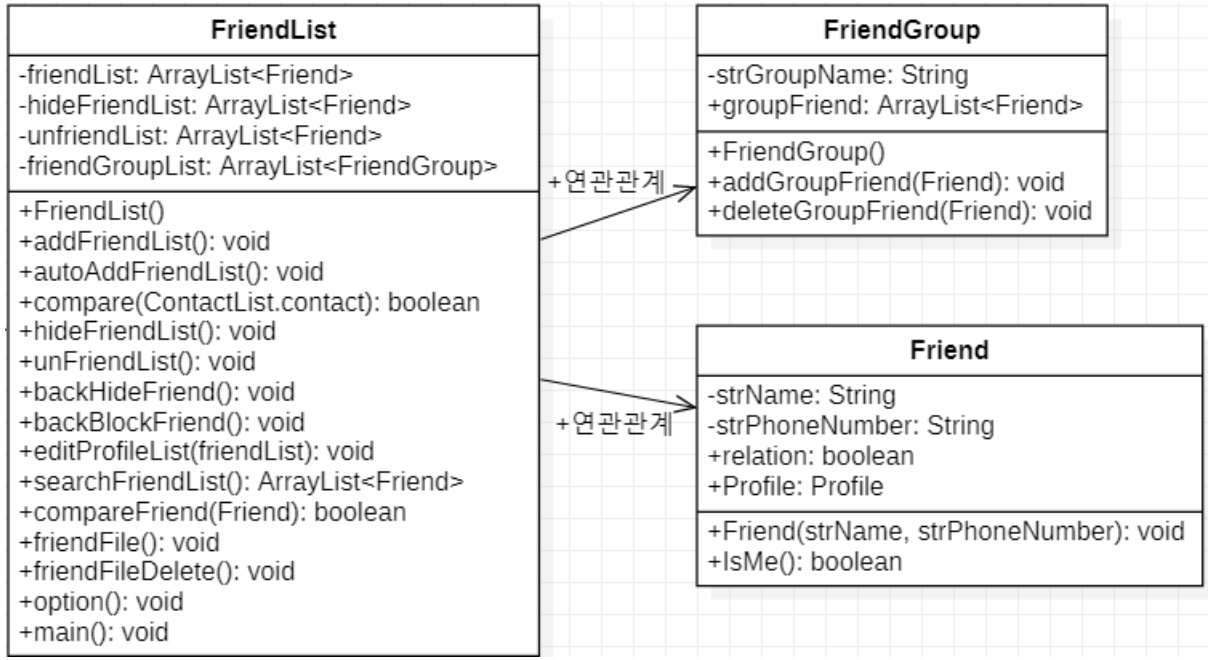
2.1 프로필 관련 클래스



(1) Profile

- 변수로는 이름, 상태메시지, 프로필 사진, 배경 사진 있음
- 프로필에 해당하는 변수를 반환하는 showProfile() 가 있음
- 프로필 정보를 수정하는 editProfile() 메소드가 있음

2.2 친구 목록 관련 클래스

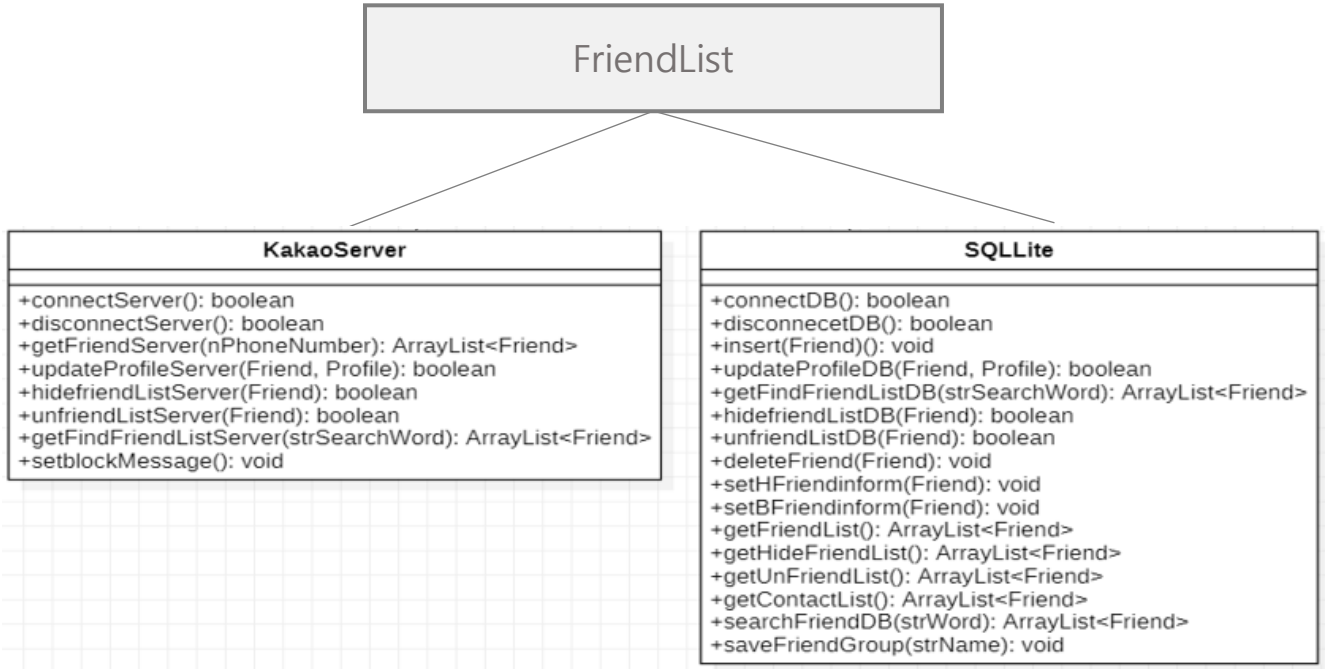


(1) FriendList
- 각 기능들이 메소드로 구현되어 있는 메인 클래스
- Friend와 FriendGroup, profile 클래스 객체를 생성하여 각 클래스의 메소드에 접근함

(2) Friend
- 친구 정보(이름, 전화번호)를 저장하는 클래스
- FriendList 클래스에서 ArrayList에 Friend 클래스 객체를 넣음
- 겟 셋 메소드 생략

(3) **FriendGroup**
- 하나의 그룹 정보를 저장하는 클래스
- FriendGroup 클래스에서 ArrayList에 Friend 클래스 객체를 넣음

2.3 데이터베이스 관련 클래스



(1) KakaoServer

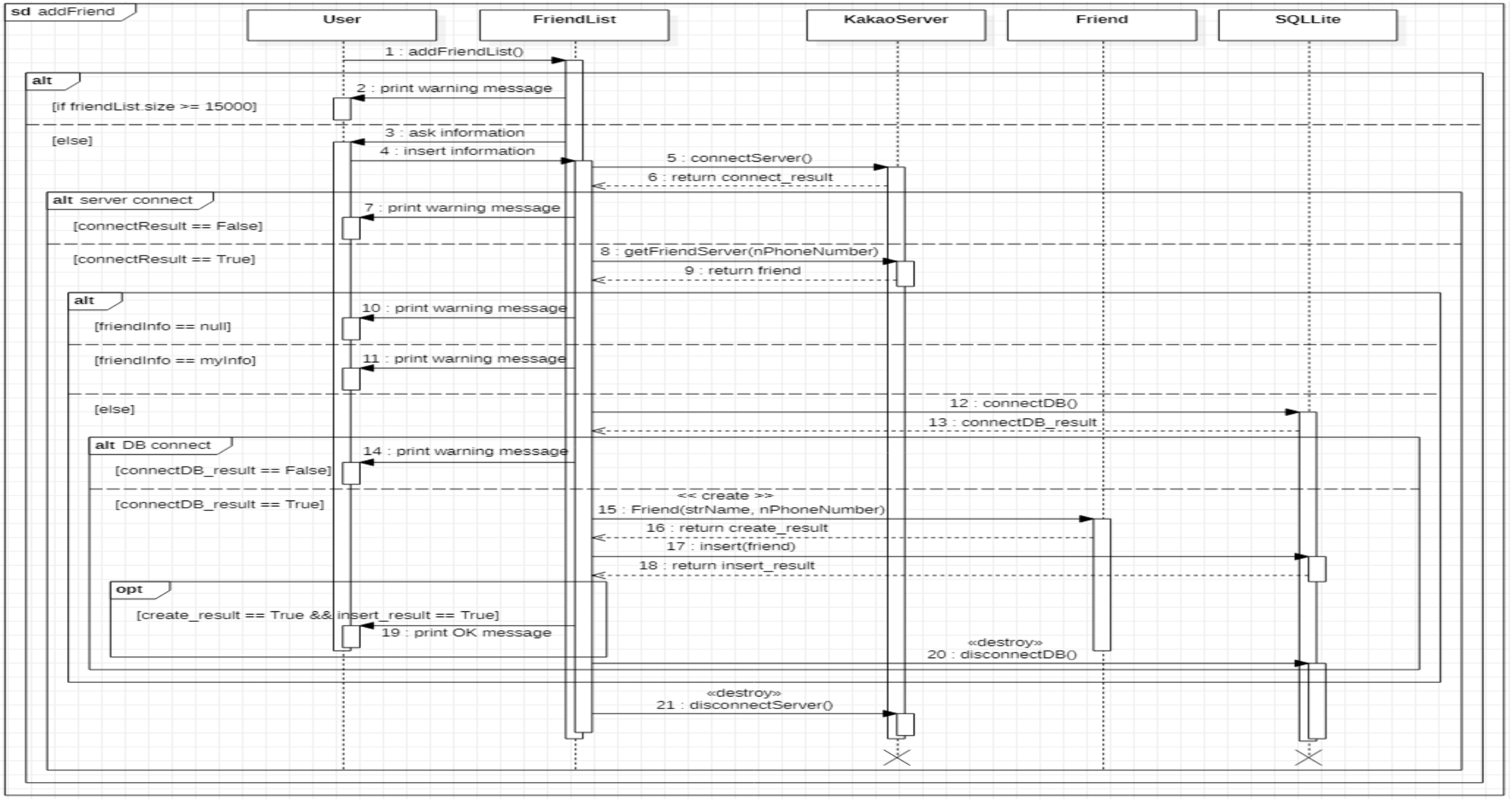
- 서버와 연결을 하기 위한 메소드
- 데이터를 주고 받기 위한 메소드를 가짐

(2) SQLLite

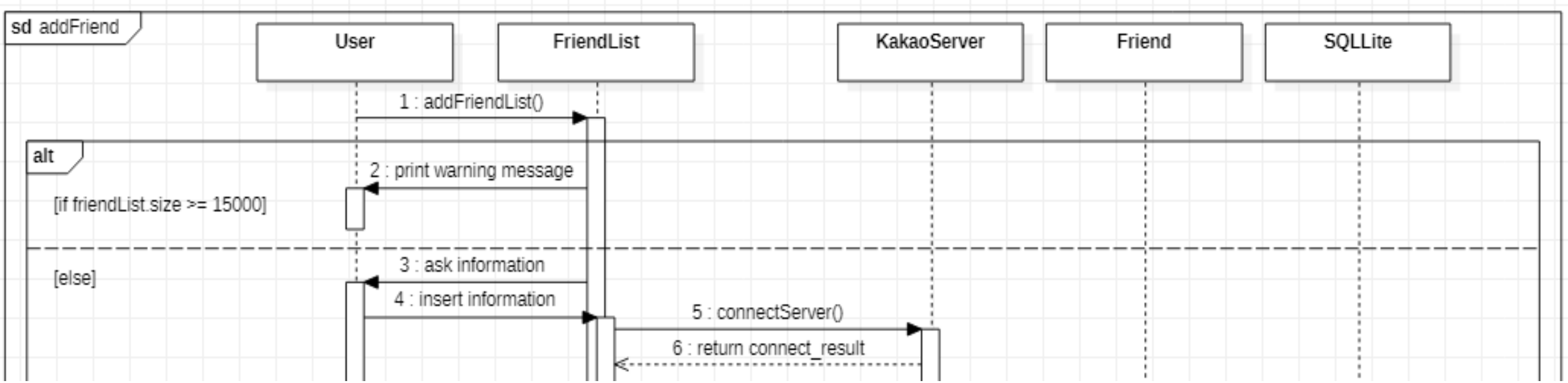
- 연락처 및 카카오톡 친구목록정보를 저장하는 모바일 전용 데이터베이스

시퀀스 다이어그램

U1. 친구추가

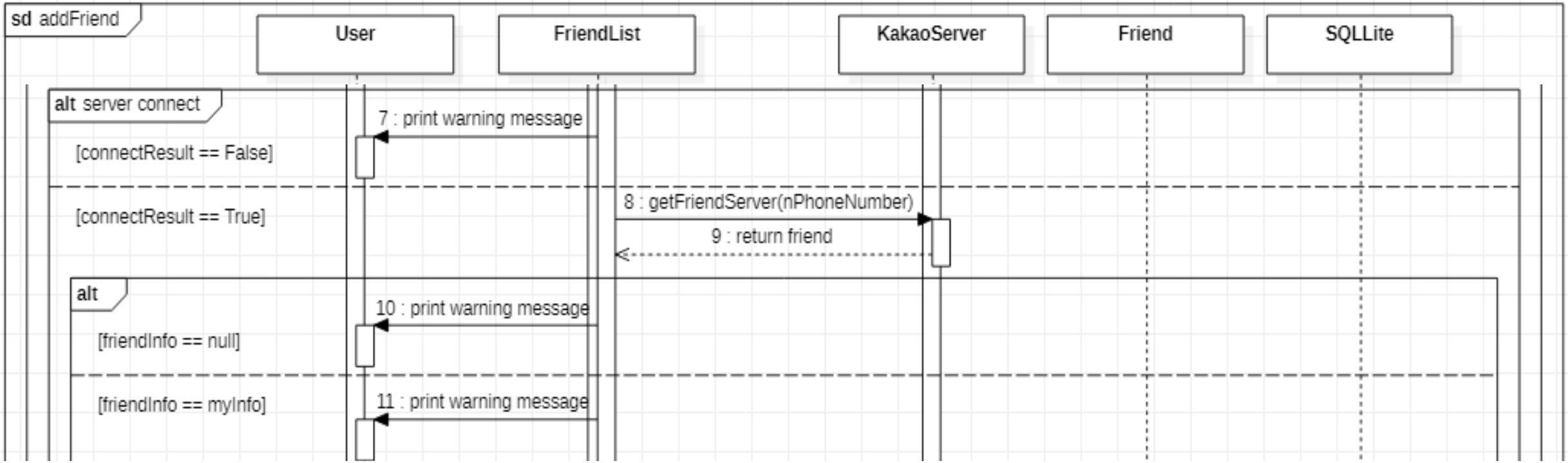

sd addFriend
User
FriendList
KakaoServer
Friend
SQLLite
1 : addFriendList()
alt
[if friendList.size >= 15000]
2 : print warning message
[else]
3 : ask information
4 : insert information
5 : connectServer()
6 : return connect_result
alt server connect
[connectResult == False]
7 : print warning message
[connectResult == True]
8 : getFriendServer(nPhoneNumber)
9 : return friend
alt
[friendInfo == null]
10 : print warning message
[friendInfo == myInfo]
11 : print warning message
[else]
12 : connectDB()
13 : connectDB_result
alt DB connect
[connectDB_result == False]
14 : print warning message
[connectDB_result == True]
<< create >>
15 : Friend(strName, nPhoneNumber)
16 : return create_result
17 : insert(friend)
18 : return insert_result
opt
[create_result == True && insert_result == True]
19 : print OK message
«destroy»
20 : disconnectDB()
«destroy»
21 : disconnectServer()

U1. 친구추가



1. 사용자(User)가 친구 추가 버튼을 클릭하여 **addFriendList()** 메소드를 호출한다.
2. 만약 저장되어 있는 친구의 수가 15000명을 초과했다면, **경고 메시지를 출력**한다.
3. 그렇지 않다면, 사용자에게 정보(친구 이름, 전화번호) **입력을 요구**한다.
4. 사용자가 정보(친구 이름, 전화번호)를 **입력**한다.
5. FriendList에서 KakaoServer에 **연결을 요청**한다.
6. KakaoServer에서 FriendList에게 **연결 결과(boolean)를 반환**한다.

U1. 친구추가



7. 만약 서버 연결에 실패했다면, **경고 메시지를 출력**한다.

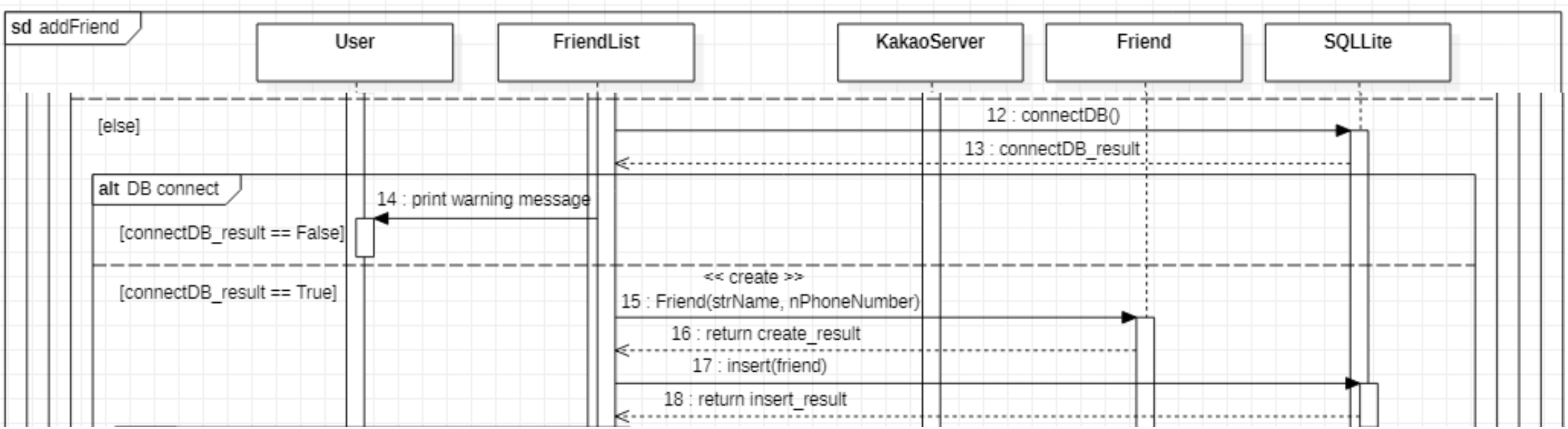
8. 서버 연결에 성공했다면, FriendList에서 KakaoServer로부터 입력한 전화번호에 대한 **정보를 요청**한다.

9. KakaoServer에서 FriendList에 **친구 정보(Friend)를 반환**한다.

10. 만약 반환 받은 친구 정보가 없다면(=전화번호를 잘못 입력했다면), 경고 메시지를 출력한다.

11. 만약 반환 받은 친구 정보가 내 전화번호라면(=내 전화번호를 입력했다면), 경고 메시지를 출력한다.

U1. 친구추가



12. 전화번호를 제대로 입력했다면, FriendList에서 SQLLite로 **DB 연결을 요청**한다.

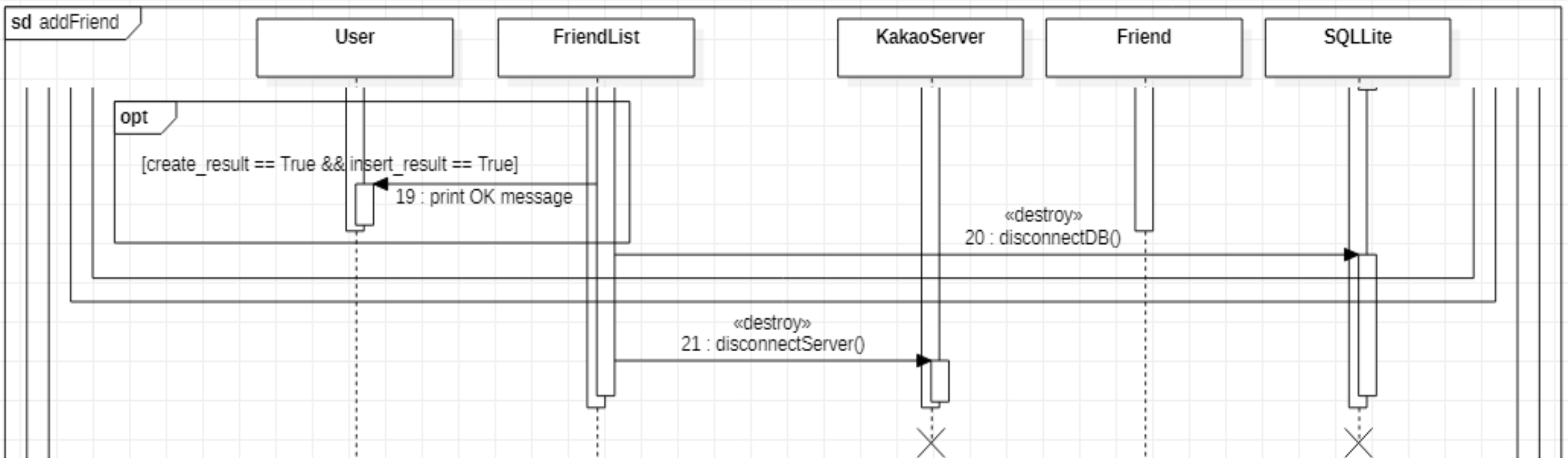
13. DB **연결에 대한 결과(boolean)를 반환**한다.

14. 만약 DB에 연결이 되지 않았다면 경고 메시지를 출력한다.

15. FriendList에서 Friend객체를 생성하여 **이름과 전화번호를 추가**해주고, 16. 결과(boolean)를 반환한다.

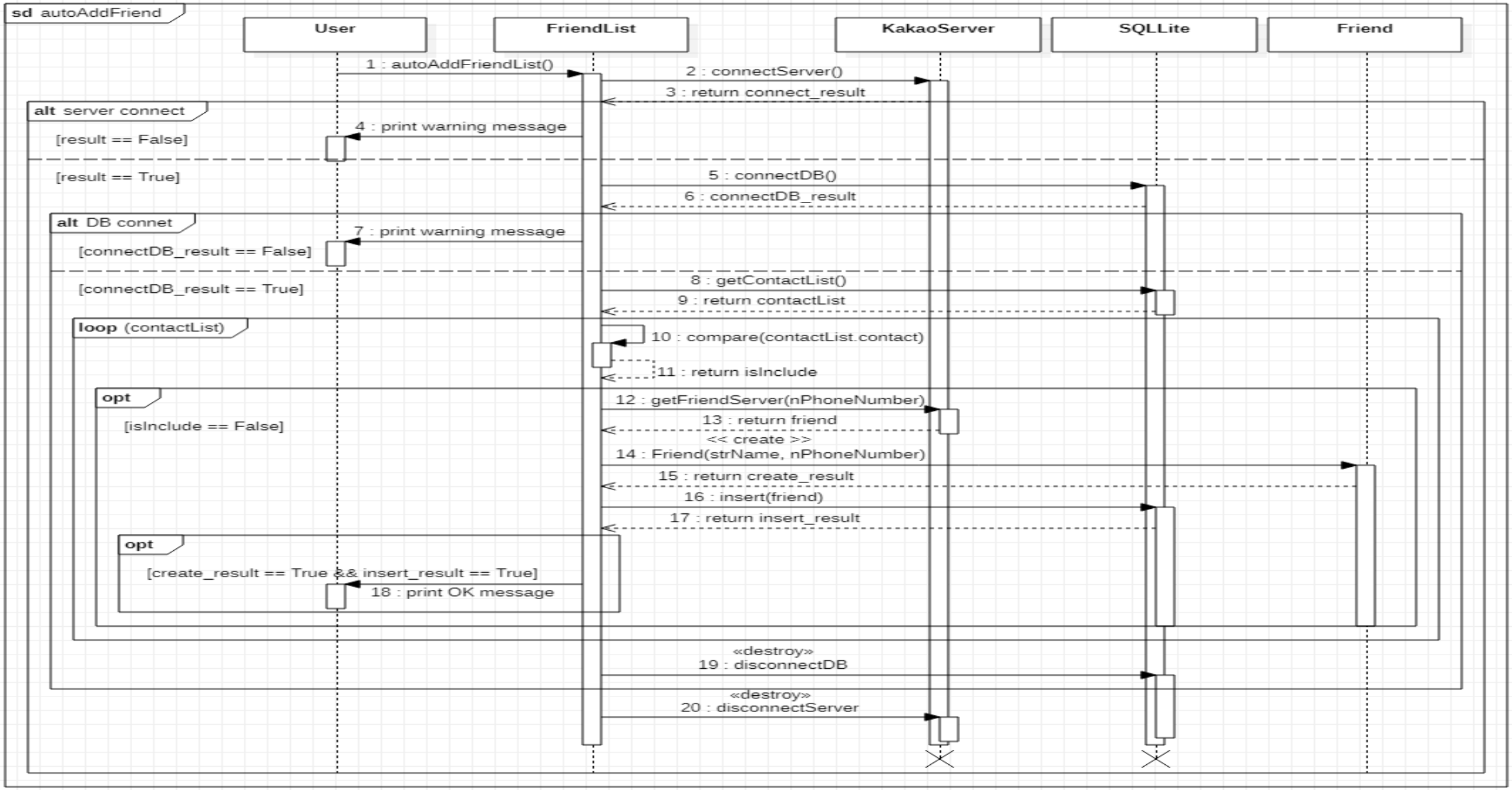
17. FriendList에서 **SQLLite에 추가된 친구를 입력**해주고, 18. 그 결과(boolean)를 반환한다.

U1. 친구추가

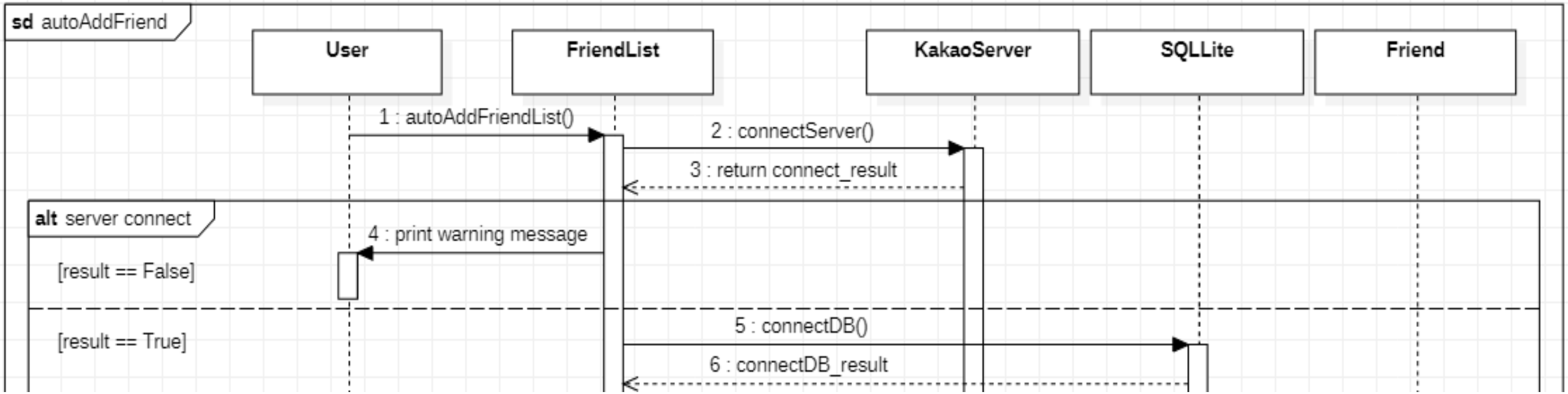


19. 만약 Friend 객체가 생성되었고, 카카오 서버에 친구 정보가 저장이 잘 되었다면 **OK 메시지를 출력**한다.
20. DB 연결을 **destroy** 한다.
21. 카카오 서버 연결을 **destroy** 한다.

U2. 자동친구추가

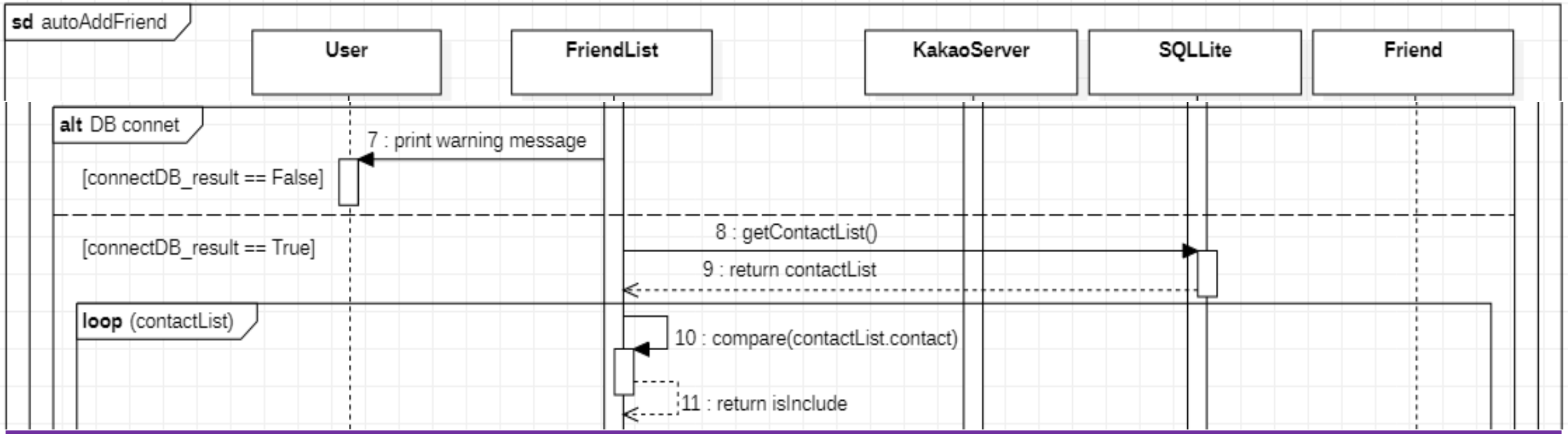

sd autoAddFriend
User
FriendList
KakaoServer
SQLLite
Friend
1 : autoAddFriendList()
2 : connectServer()
3 : return connect_result
alt server connect
[result == False]
4 : print warning message
[result == True]
5 : connectDB()
6 : connectDB_result
alt DB connet
[connectDB_result == False]
7 : print warning message
[connectDB_result == True]
8 : getContactList()
9 : return contactList
loop (contactList)
10 : compare(contactList.contact)
11 : return isInclude
opt
[isInclude == False]
12 : getFriendServer(nPhoneNumber)
13 : return friend
<< create >>
14 : Friend(strName, nPhoneNumber)
15 : return create_result
16 : insert(friend)
17 : return insert_result
opt
[create_result == True && insert_result == True]
18 : print OK message
«destroy»
19 : disconnectDB
«destroy»
20 : disconnectServer

U2. 자동친구추가



1. 사용자(User)가 자동 친구 추가 버튼을 클릭하여 **autoAddFriendList()** 메소드를 호출한다.
2. FriendList에서 KakaoServer에 **연결을 요청**한다.
3. KakaoServer에서 FriendList에게 **연결 결과(boolean)를 반환**한다.
4. 만약 서버 연결에 실패했다면, **경고 메시지를 출력**한다.
5. 서버 연결에 성공했다면, FriendList에서 SQLLite에 **연결을 요청**한다.
6. SQLLite에서 FriendList로 **연결 결과(boolean)를 반환**한다.

U2. 자동친구추가



7. 만약 DB 연결에 실패했다면, **경고 메시지를 출력**한다.

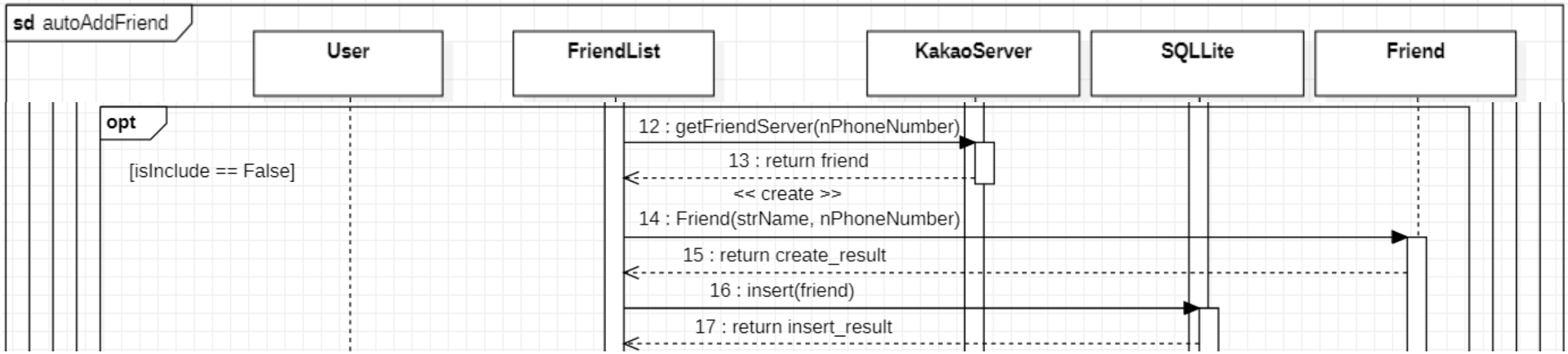
8. DB 연결에 성공했다면, FriendList에서 SQLLite로 **휴대폰 연락처 목록을 요청**한다.

9. SQLLite에서 **휴대폰 연락처 목록(ArrayList)을 반환**한다.

10. 휴대폰 연락처 목록(contactList)에 들어있는 연락처의 개수만큼 loop문을 반복하며, 카카오톡 친구 목록과 휴대폰 연락처 목록(contactList.contact)을 **비교**한다.

11. 휴대폰 연락처 목록의 연락처가 카카오톡 친구 목록에 있는지에 대한 **여부(boolean)를 반환**한다.

U2. 자동친구추가



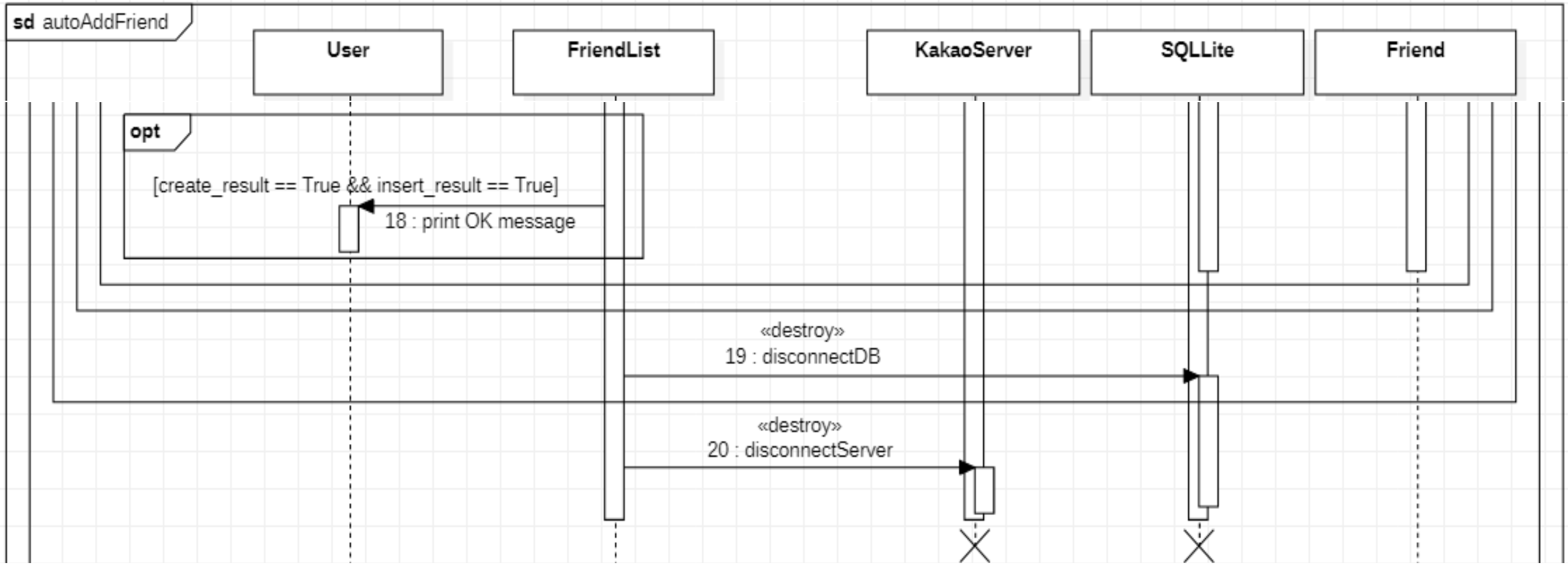
12. 만약 휴대폰 연락처 목록의 연락처가 카카오톡 친구 목록에 포함되어 있지 않다면, FriendList에서 KakaoServer로부터 해당 전화번호에 대한 **정보를 요청**한다.

13. KakaoServer에서 FriendList에 **친구 정보(Friend)를 반환**한다.

14. FriendList에서 Friend객체를 생성하여 **이름과 전화번호를 추가**해주고, 15. 결과(boolean)를 반환한다.

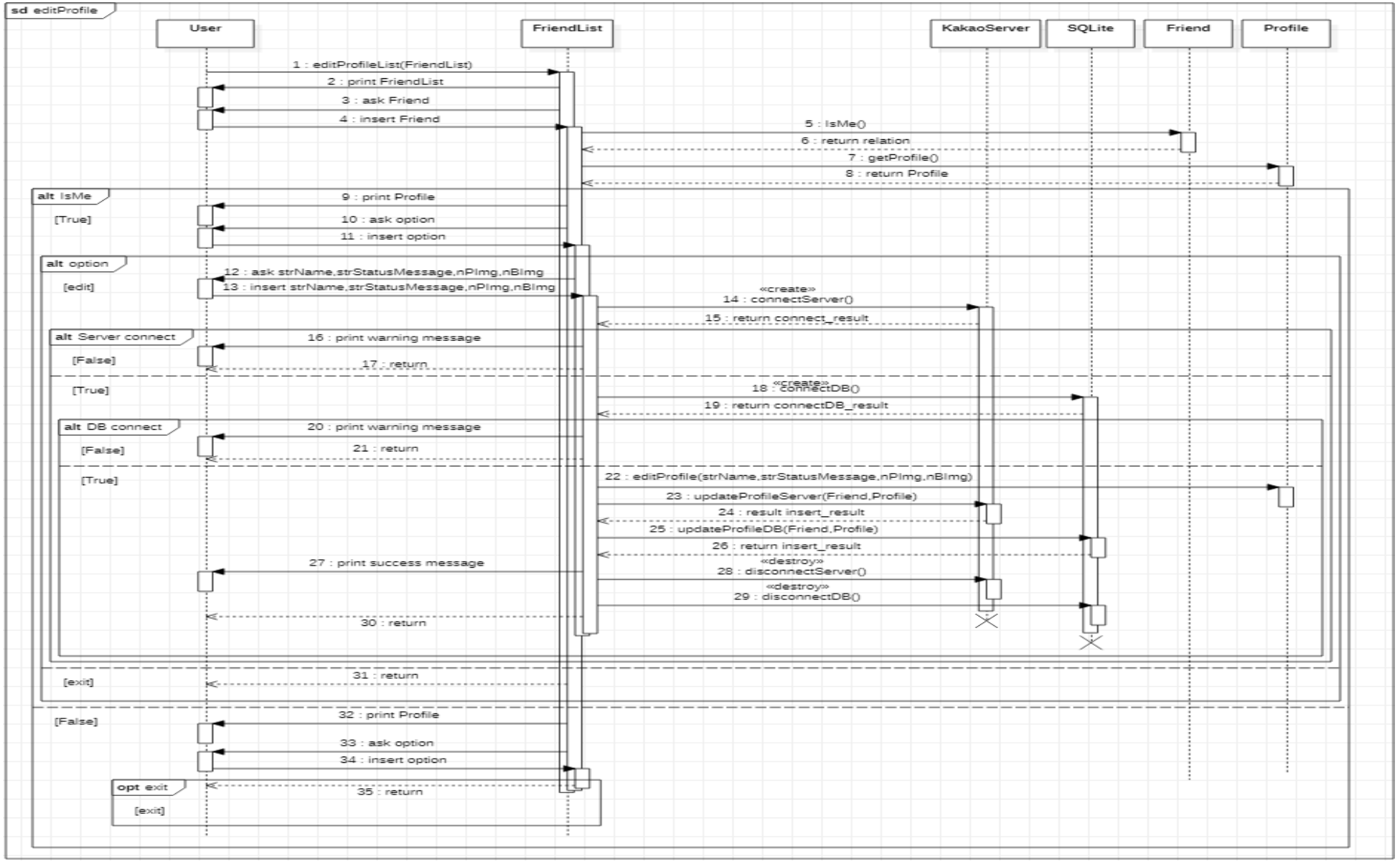
16. FriendList에서 **SQLLite에 추가된 친구를 입력**해주고, 17. 그 결과(boolean)를 반환한다.

U2. 자동친구추가

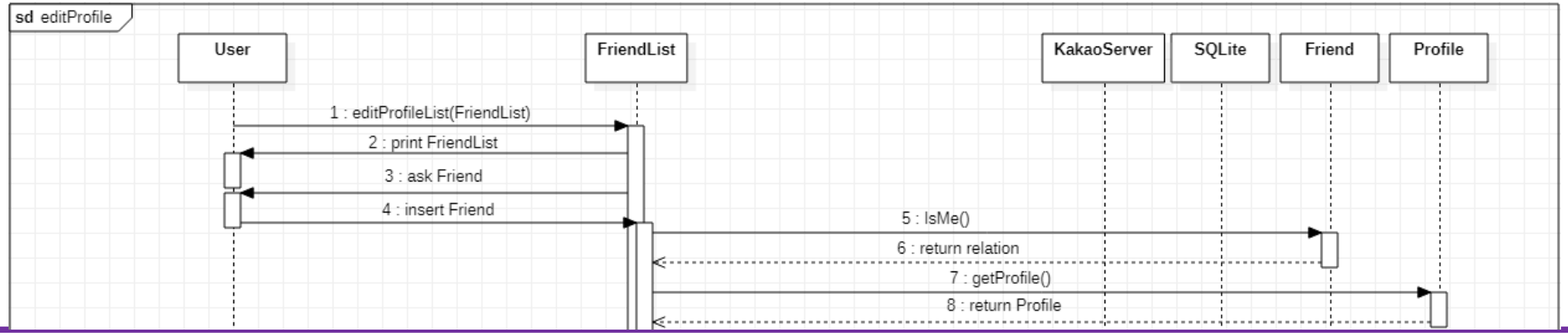


18. 만약 Friend 객체가 생성되었고, 카카오 서버에 친구 정보가 저장이 잘 되었다면 **OK 메시지를 출력**한다.
19. DB 연결을 **destroy** 한다.
20. 카카오 서버 연결을 **destroy** 한다.

U3. 프로필 편집

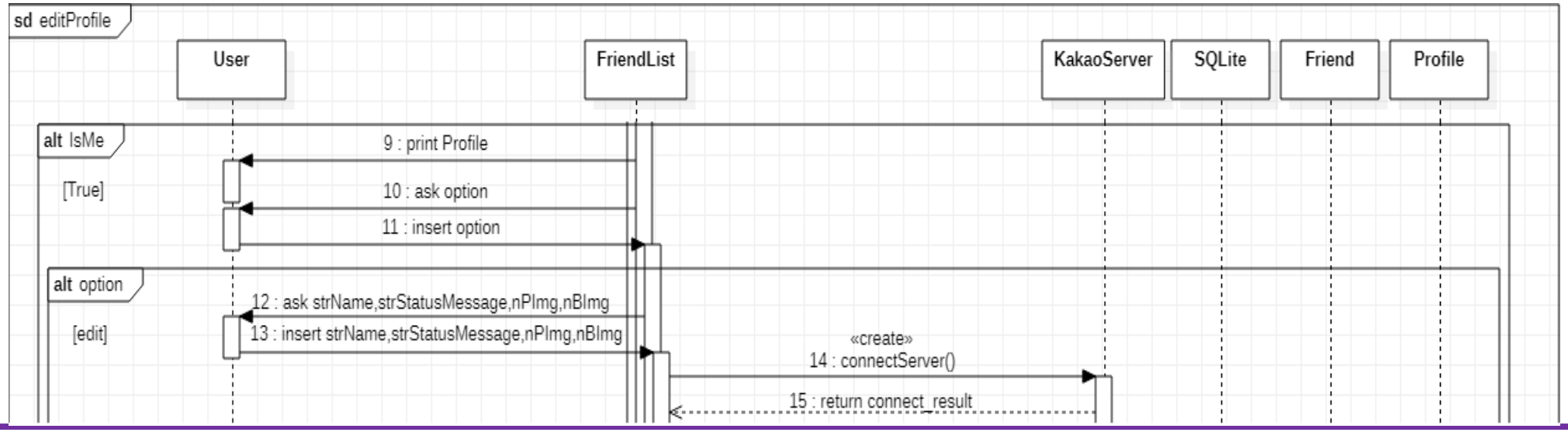

sd editProfile
User
FriendList
KakaoServer
SQLite
Friend
Profile
1 : editProfileList(FriendList)
2 : print FriendList
3 : ask Friend
4 : insert Friend
5 : IsMe()
6 : return relation
7 : getProfile()
8 : return Profile
alt IsMe
[True]
9 : print Profile
10 : ask option
11 : insert option
alt option
[edit]
12 : ask strName,strStatusMessage,nPImg,nBImg
13 : insert strName,strStatusMessage,nPImg,nBImg
«create»
14 : connectServer()
15 : return connect_result
alt Server connect
[False]
16 : print warning message
17 : return
[True]
«create»
18 : connectDB()
19 : return connectDB_result
alt DB connect
[False]
20 : print warning message
21 : return
[True]
22 : editProfile(strName,strStatusMessage,nPImg,nBImg)
23 : updateProfileServer(Friend,Profile)
24 : result insert_result
25 : updateProfileDB(Friend,Profile)
26 : return insert_result
27 : print success message
«destroy»
28 : disconnectServer()
«destroy»
29 : disconnectDB()
30 : return
[exit]
31 : return
[False]
32 : print Profile
33 : ask option
34 : insert option
opt exit
[exit]
35 : return

U3. 프로필 편집



1. 메소드 **editProfileList(FriendList)**를 호출하여 프로필 편집 기능을 실행한다.
2. 전달받은 파라미터인 **FriendList(ArrayList<Friend>)**에 포함된 친구 **Friend(Friend)**들을 모두 출력한다.
3. 친구 **Friend(Friend)**를 선택하도록 요청받는다.
4. 친구 **Friend(Friend)**를 선택한다.
5. 선택한 친구가 나인지 친구인지 판단하는 **IsMe()** 메소드를 호출한다.
6. 나인지 친구인지 나타내는 **Friend 객체의 멤버변수**인 **relation(boolean)**을 반환받는다.
7. 선택한 친구의 **프로필(Profile)** 정보를 받기 위해 **getProfile()** 메소드를 호출하여 한다.
8. 선택한 친구의 **프로필(Profile)** 정보인 **Profile(Profile)**을 반환받는다.

U3. 프로필 편집



9. 선택한 친구가 자신 즉, **IsMe()** 메소드를 통해 받은 값이 **True**라면 프로필 **Profile(Profile)**정보를 출력한다.

10. 프로필 보기를 종료할지 프로필 편집을 실행할지에 대한 **option**을 입력 요청한다.

11. 프로필 보기, 프로필 편집 중 하나의 **option**을 입력한다.

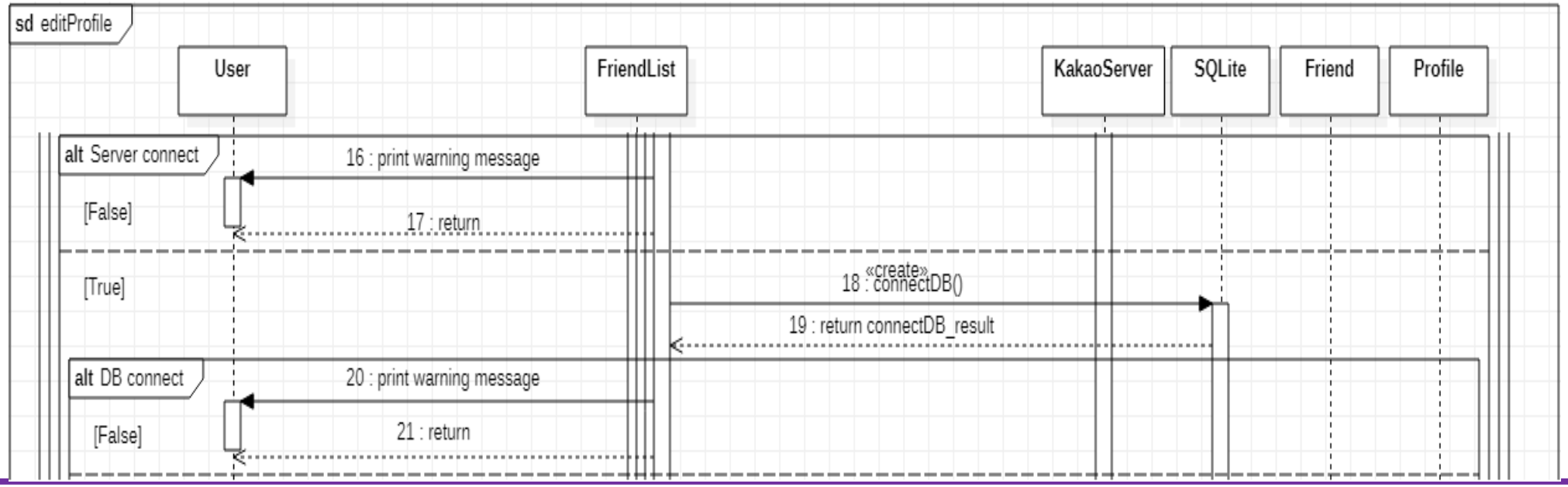
12. 편집을 선택하면 프로필에 들어갈 정보 입력 **(strName, strStatusMessage, nPImg, nBImg)**을 요청한다.

13. 프로필 수정에 필요한 정보**(strName, strStatusMessage, nPImg, nBImg)**를 입력한다.

14. **connectServer()** 메소드를 호출하여 서버에 연결을 시도한다.

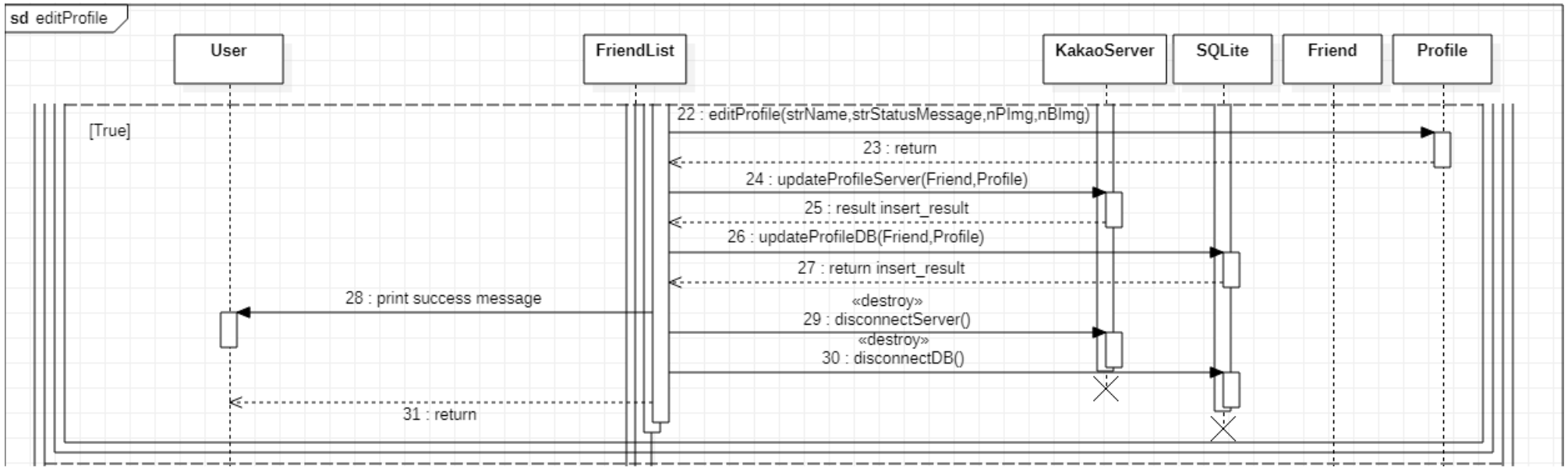
15. 서버 연결 결과 **connect_result(boolean)**을 반환받는다.

U3. 프로필 편집



16. 서버 연결 결과가 **False**라면, 네트워크가 불안하다는 메시지를 출력한다.
17. 그리고 **return(void)**으로 종료한다.
18. 서버 연결 결과가 **True**라면, **connectDB()** 메소드를 호출하여 SQLite에도 연결을 시도한다.
19. SQLite 연결 결과 **connectDB_result(boolean)**을 반환받는다.
20. SQLite 연결 결과가 **False**라면, 데이터를 로드하는데 실패하였다는 메시지를 출력한다.
21. 그리고 **return(void)**으로 종료한다.

U3. 프로필 편집



22~23. SQLite와의 연결도 True라면, 이전에 입력받은 정보를 바탕으로 **editProfile(strName, strStatusMessage, nPImg, nBImg)** 메소드를 호출하여 프로필을 수정한 후 결과 **return(void)**를 반환한다.

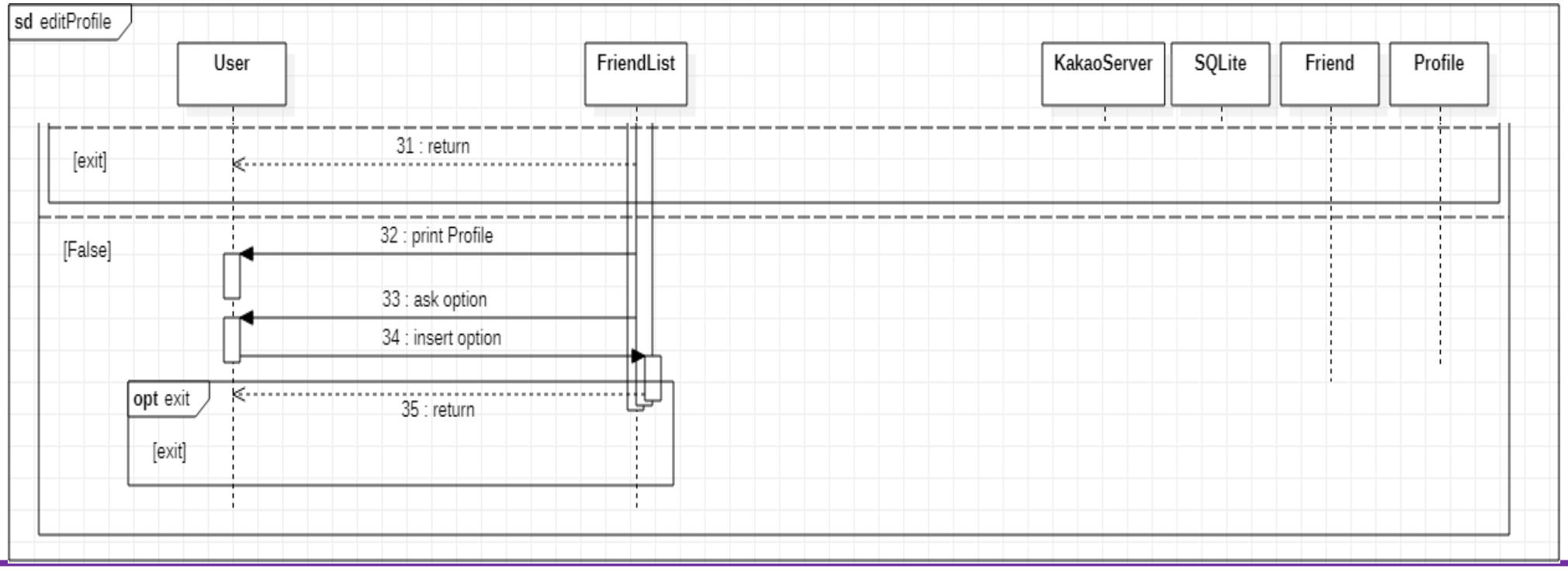
23~26. 수정된프로필을 **updateProfileServer(Friend(Friend), Profile(Profile))** 메소드와 **updateProfileDB(Friend(Friend), Profile(Profile))** 메소드를 호출하여 서버와 SQLite에 저장한다.

27. 수정 완료 메시지를 출력한다.

28~29. 서버와 SQLite와의 연결을 **disconnectServer()** 메소드와 **disconnectDB()** 메소드를 호출하여 종료한다.

30. **return(void)**을 하여 함수를 종료한다.

U3. 프로필 편집



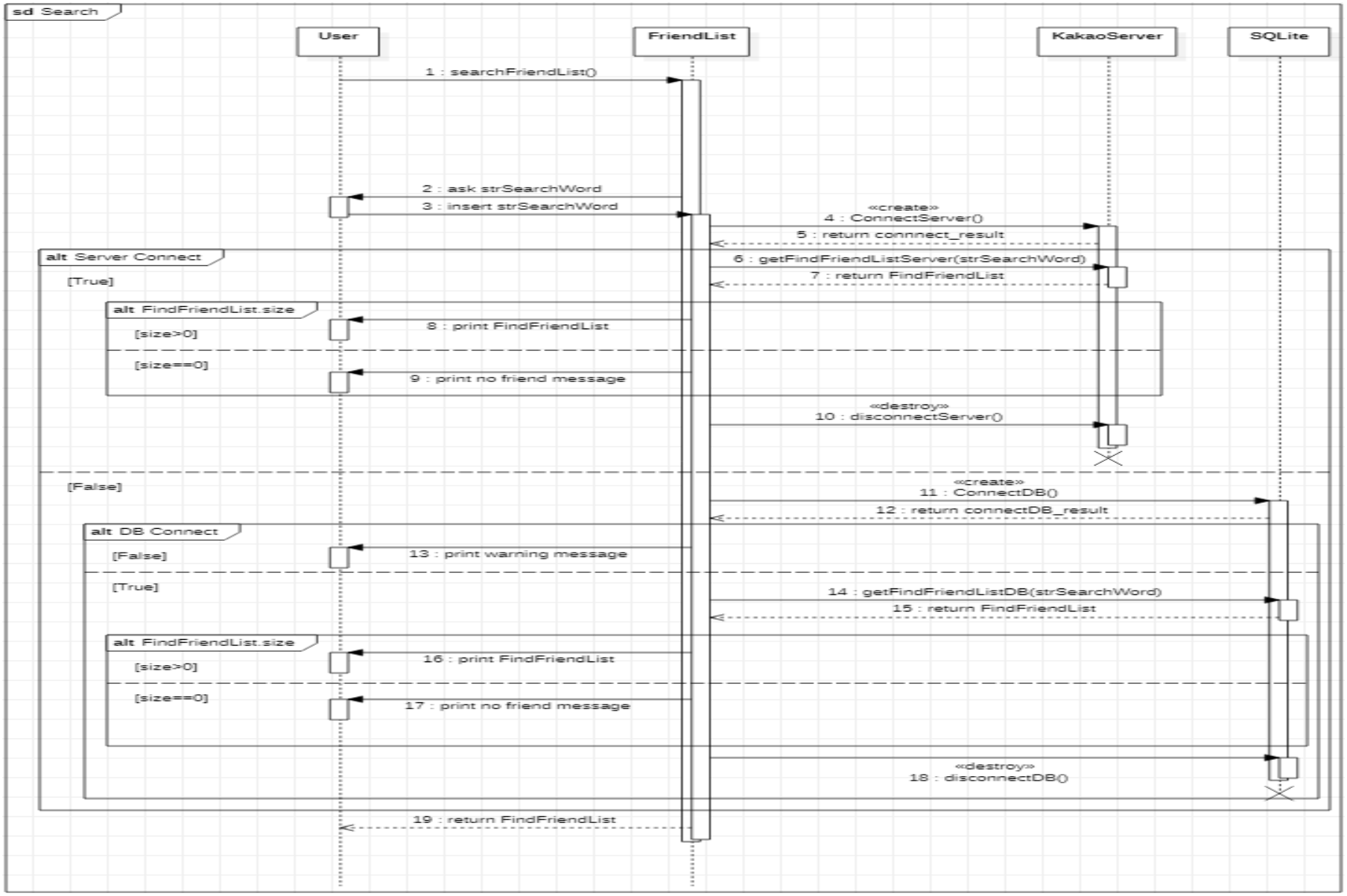
31. 선택한 친구가 나인 상태에서 option을 exit을 입력하였다면 **return(void)**로 종료한다.

32. 선택한 친구가 자신 즉, **IsMe()** 메소드를 통해 받은 값이 **False**라면 프로필 **Profile(Profile)** 정보를 출력한다

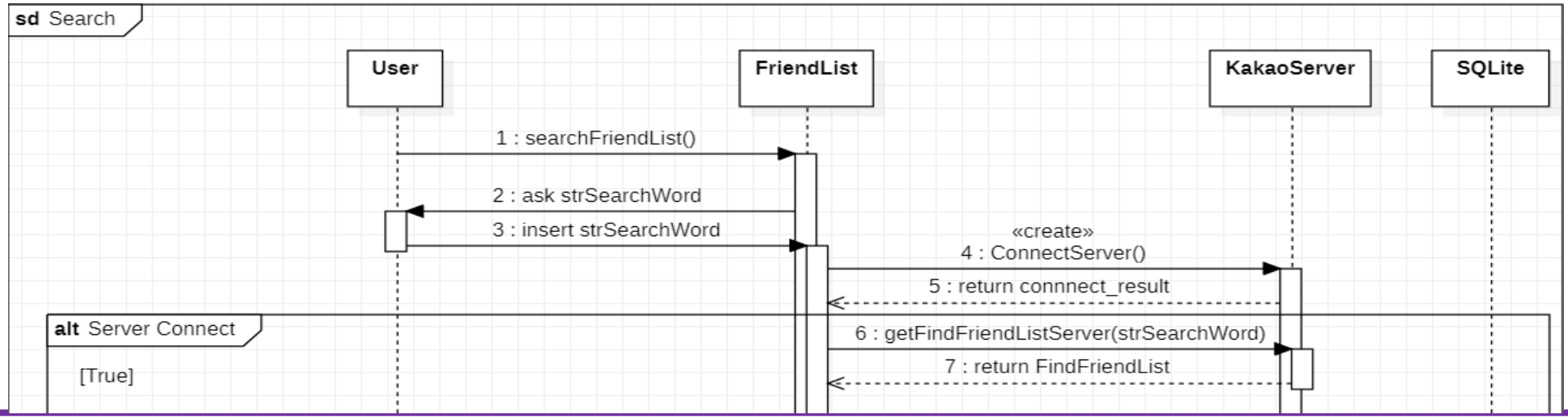
33. 프로필 보기를 종료할 건지에 대한 **option**을 입력 요청한다.

34. **exit**을 입력하면 **return(void)**를 통해 기능을 종료한다.

U4. 친구 검색

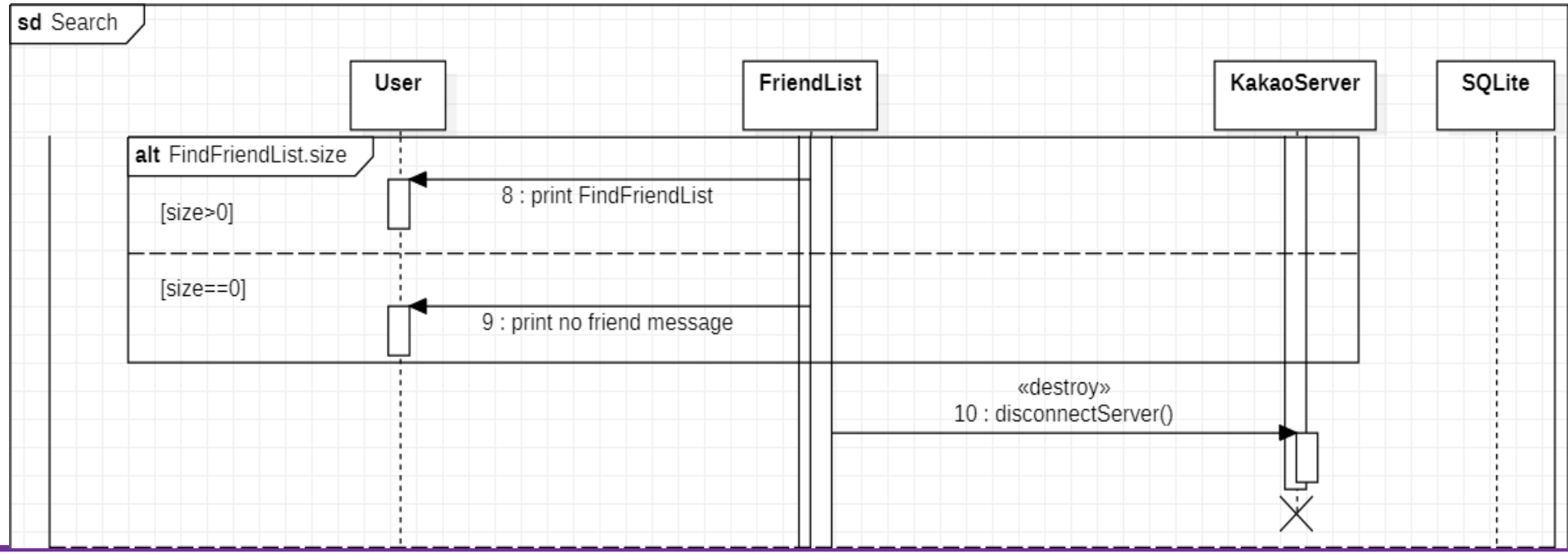

sd Search
User
FriendList
KakaoServer
SQLite
1 : searchFriendList()
2 : ask strSearchWord
3 : insert strSearchWord
«create»
4 : ConnectServer()
5 : return connnect_result
alt Server Connect
[True]
6 : getFindFriendListServer(strSearchWord)
7 : return FindFriendList
alt FindFriendList.size
[size>0]
8 : print FindFriendList
[size==0]
9 : print no friend message
«destroy»
10 : disconnectServer()
[False]
«create»
11 : ConnectDB()
12 : return connectDB_result
alt DB Connect
[False]
13 : print warning message
[True]
14 : getFindFriendListDB(strSearchWord)
15 : return FindFriendList
alt FindFriendList.size
[size>0]
16 : print FindFriendList
[size==0]
17 : print no friend message
«destroy»
18 : disconnectDB()
19 : return FindFriendList

U4. 친구 검색



1. 사용자가 **searchFriendList()** 메소드를 호출하여 검색 기능을 시작한다.
2. 연락처나 이름에 해당하는 검색어 **strSearchWord(String)**을 사용자에게 요청한다.
3. 사용자는 검색어 **strSearchWord(String)**을 입력한다.
4. 검색어를 입력하면 **connectServer()** 메소드를 호출하여 서버 연결을 시도한다.
5. 서버 연결 결과 **connect_result(Boolean)**을 반환한다.
6. 서버 연결 결과가 True라면, **getFindFriendListServer(strSearchWord)** 메소드를 호출하여 검색어로 해당 정보의 친구를 서버를 통해 찾는다.
7. 찾은 친구 목록 **FindFriendList(ArrayList<Friend>)**을 반환한다.

U4. 친구 검색

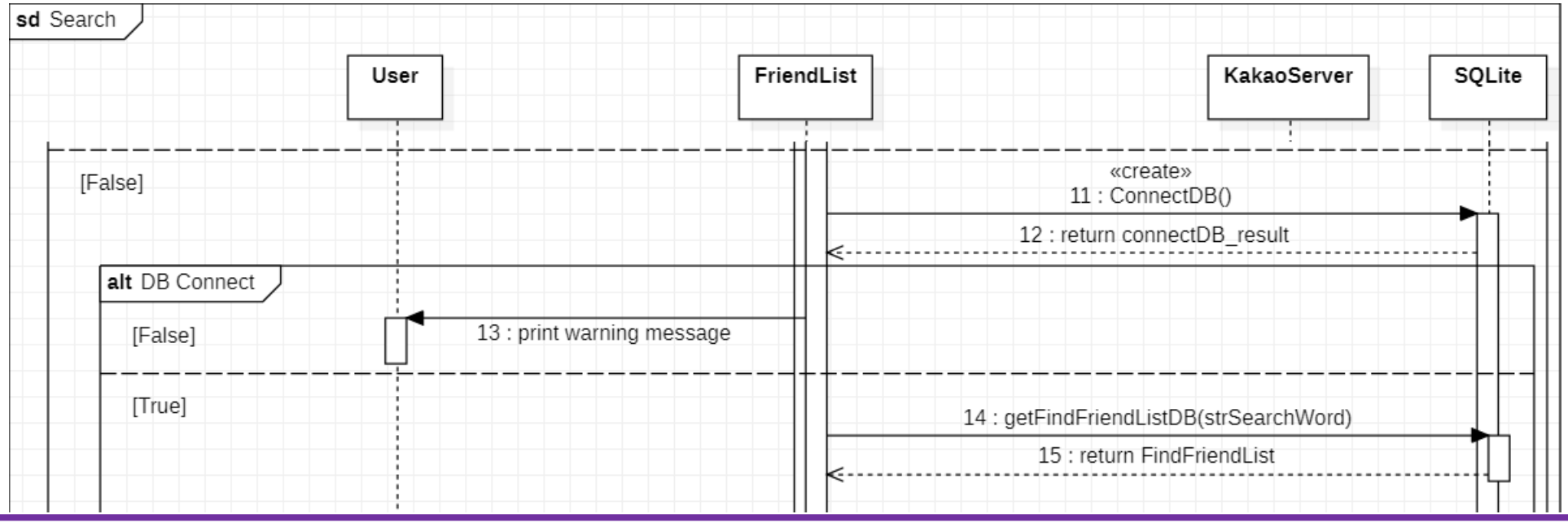


8. 반환받은 **FindFriendList(ArrayList<Friend>)**의 사이즈(원소 수)가 0보다 크다면 찾은 친구 목록의 친구 정보(이름, 상태메시지)를 모두 출력한다.

9. 반환받은 **FindFriendList(ArrayList<Friend>)**의 사이즈(원소 수)가 0이라면(찾은 친구가 없다면), 해당 정보의 친구가 없음을 의미하는 메시지를 출력한다.

10. 서버를 통해 친구를 찾은 후, **disconnectServer()**로 서버 연결을 종료한다.

U4. 친구 검색



11. 서버 연결 결과가 **False**라면, **ConnectDB()** 메소드를 호출하여 SQLite와 연결을 시도한다.

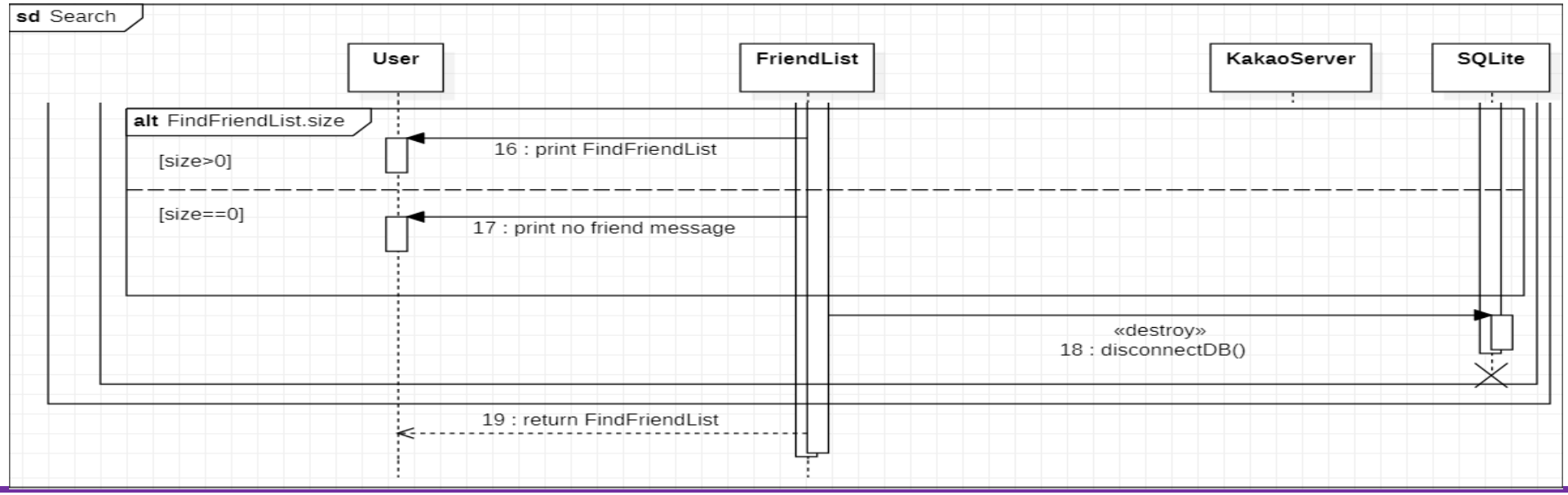
12. SQLite 연결 결과 **connectDB_result(Boolean)**을 반환한다.

13. SQLite 연결 결과 **False**라면, 연결이 불안정하다는 의미의 경고 메시지를 출력한다.

14. SQLite 연결 결과가 **True**라면, **getFindFriendListDB(strSearchWord)** 메소드를 호출하여 검색어로 해당 정보의 친구를 서버를 통해 찾는다.

15. 찾은 친구 목록 **FindFriendList(ArrayList<Friend>)**을 반환한다.

U4. 친구 검색



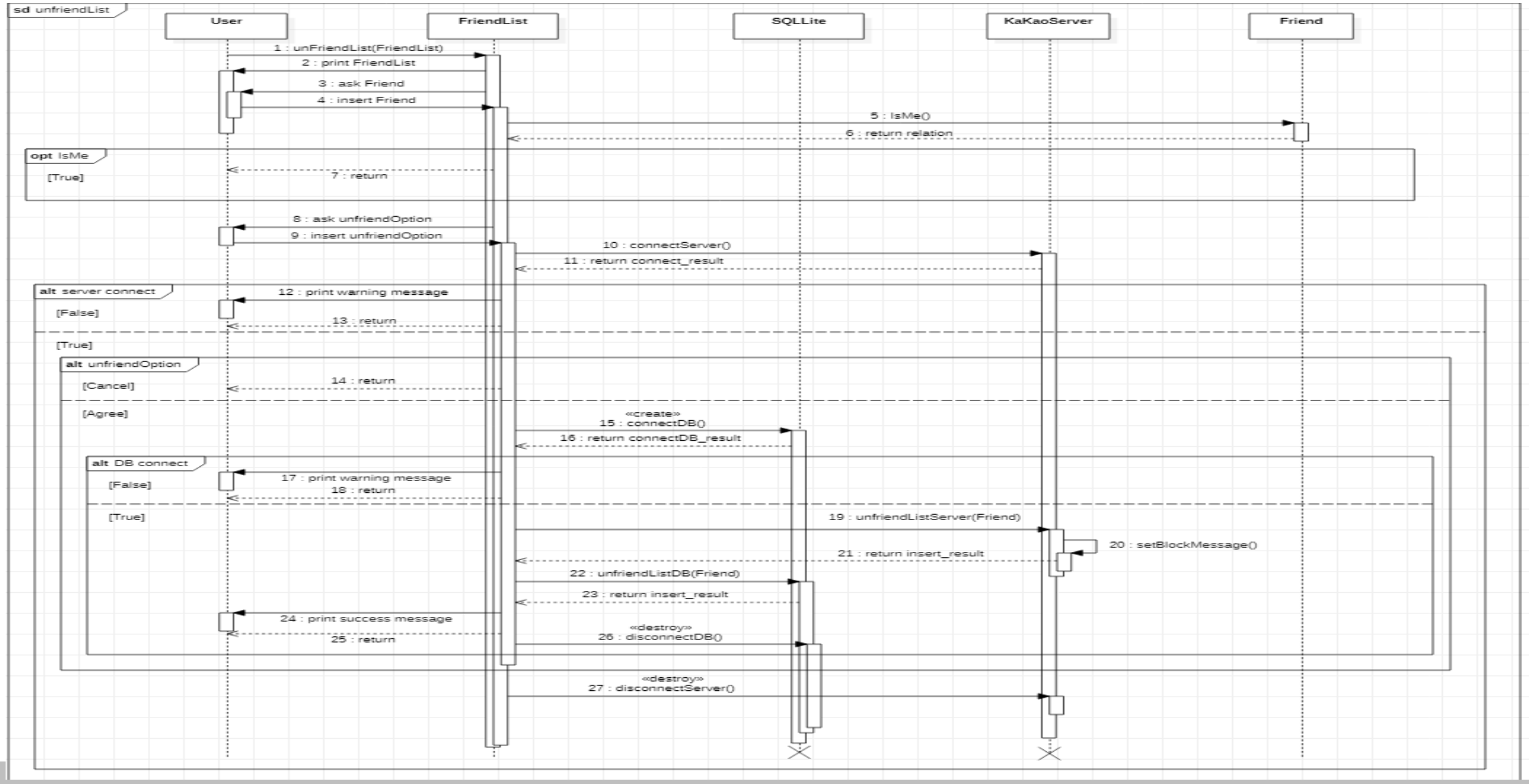
16. 반환받은 **FindFriendList(ArrayList<Friend>)**의 사이즈(원소 수)가 0보다 크다면 찾은 친구 목록의 친구 정보(이름, 상태메시지)를 모두 출력한다.

17. 반환받은 **FindFriendList(ArrayList<Friend>)**의 사이즈(원소 수)가 0이라면(찾은 친구가 없다면), 해당 정보의 친구가 없음을 의미하는 메시지를 출력한다.

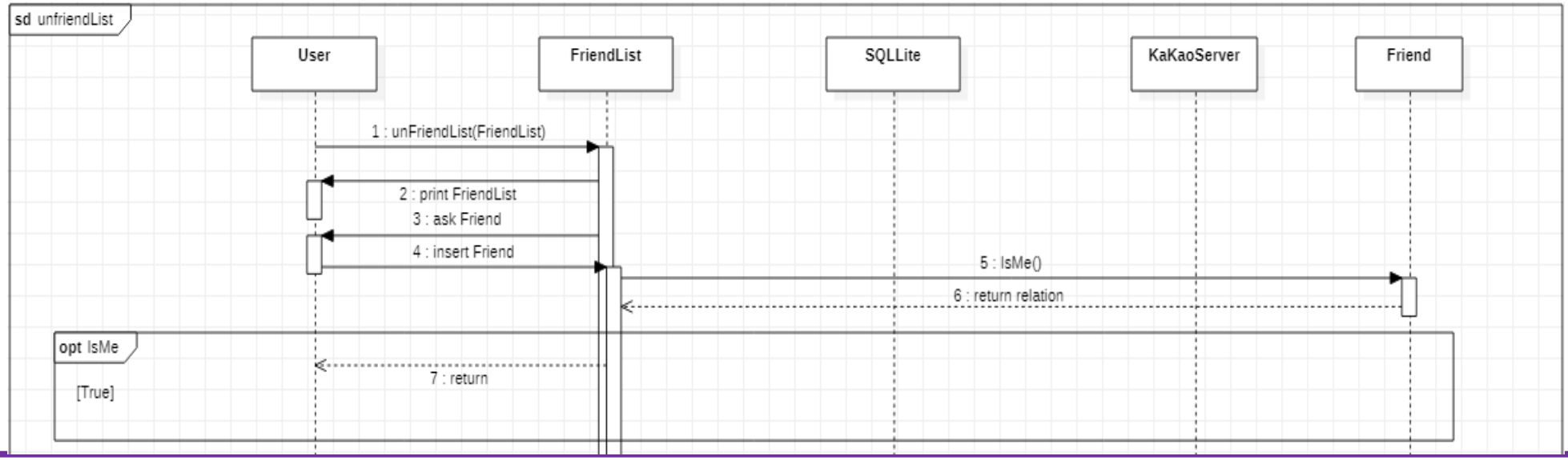
18. SQLite를 통해 친구를 찾은 후, **disconnectDB()**로 SQLite 연결을 종료한다.

19. 마지막으로 찾은 친구 목록 **FindFriendList(ArrayList<Friend>)**를 반환한다.

U5. 친구 차단

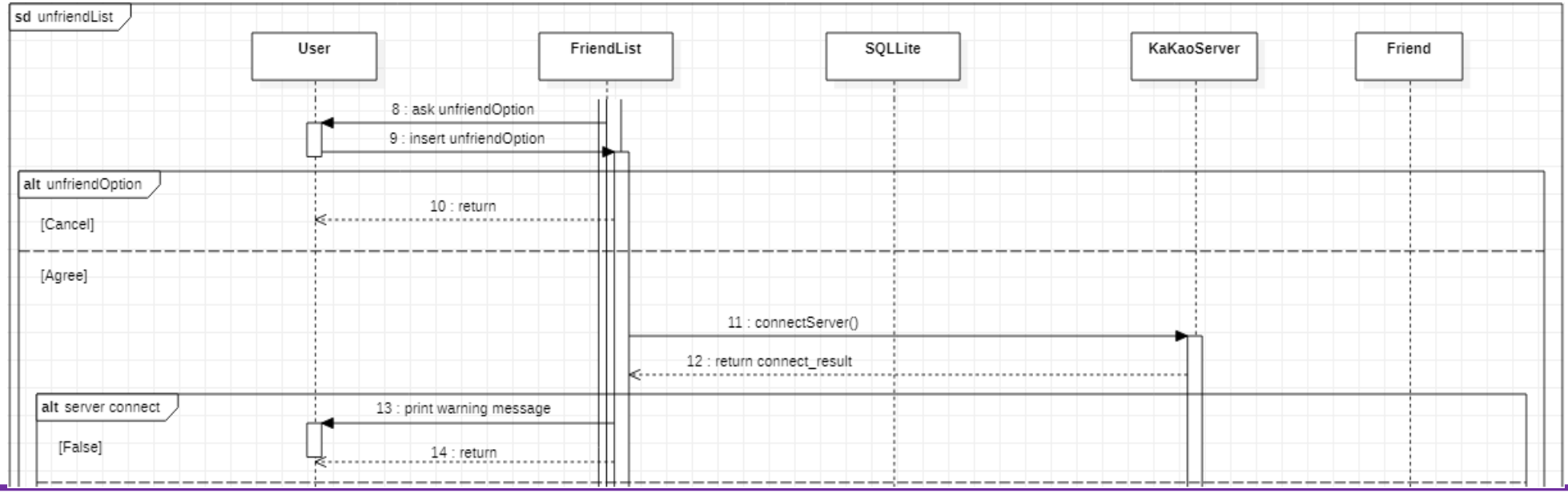

sd unfriendList
User
FriendList
SQLLite
KaKaoServer
Friend
1 : unFriendList(FriendList)
2 : print FriendList
3 : ask Friend
4 : insert Friend
5 : IsMe()
6 : return relation
opt IsMe
[True]
7 : return
8 : ask unfriendOption
9 : insert unfriendOption
10 : connectServer()
11 : return connect_result
alt server connect
12 : print warning message
[False]
13 : return
[True]
alt unfriendOption
14 : return
[Cancel]
[Agree]
«create»
15 : connectDB()
16 : return connectDB_result
alt DB connect
17 : print warning message
[False]
18 : return
[True]
19 : unfriendListServer(Friend)
20 : setBlockMessage()
21 : return insert_result
22 : unfriendListDB(Friend)
23 : return insert_result
24 : print success message
25 : return
«destroy»
26 : disconnectDB()
«destroy»
27 : disconnectServer()

U5. 친구 차단



1. unFriendList(FriendList) 메소드를 호출하여 차단 기능을 실행한다.
2. 친구목록 FriendList(ArrayList<Friend>)을 통해 친구들을 출력한다.
3. 출력된 친구를 바탕으로 친구 Friend 선택을 요청한다.
4. 친구(Friend)를 선택하여 입력한다
5. 선택한 친구가 나인지 아닌지 판단하는 IsMe() 메소드를 호출하고 결과 relation(boolean)을 반환받는다.
6~7. 선택한 친구(Friend)가 나인 경우, 즉 True라면 return(void)로 종료한다.

U5. 친구 차단



8. 선택한 친구(Friend)가 자신이 아니라면 진짜 차단할 것인지 취소할 것인지 unfriendOption을 요청한다.

9. unfriendOption을 입력한다.

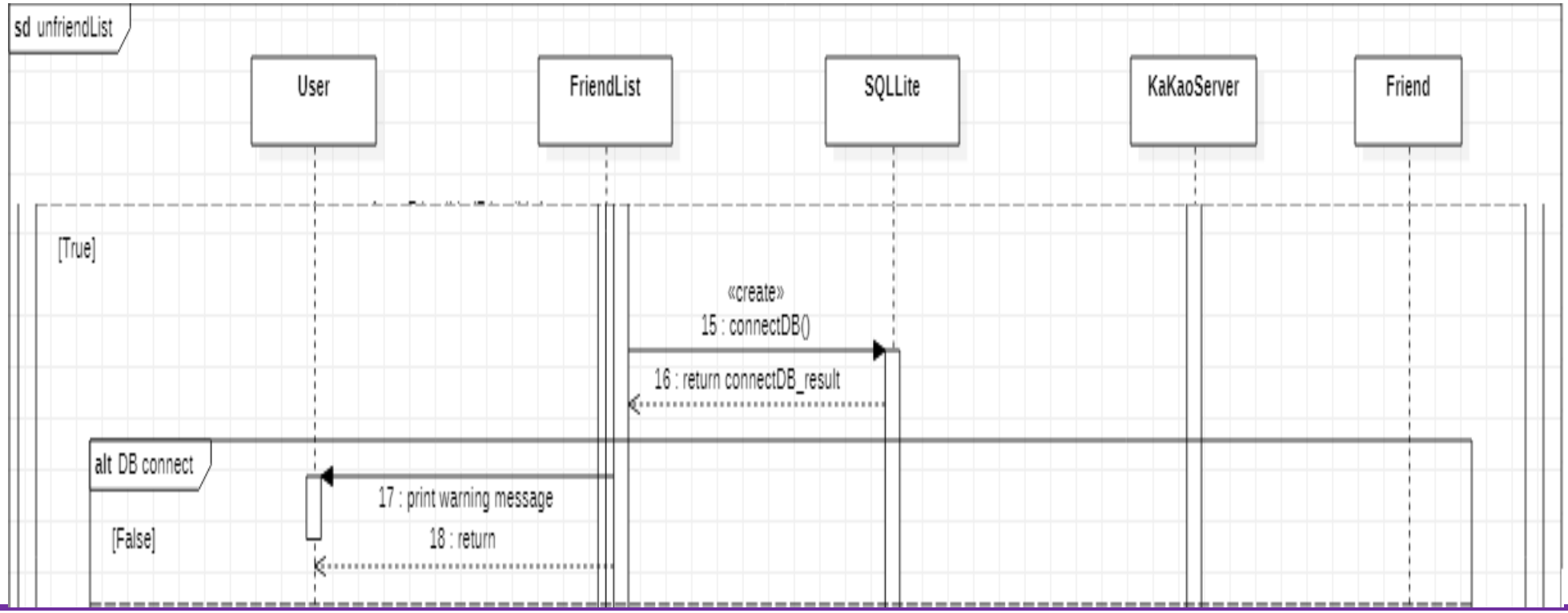
10. unfriendOption이 Cancel이라면 return(void)를 통해 종료한다.

11. unfriendOption이 Agree이라면 connectServer() 메소드를 통해 서버 연결을 시도한다.

12. 서버 연결이 안된다면 경고 메시지를 출력한다.

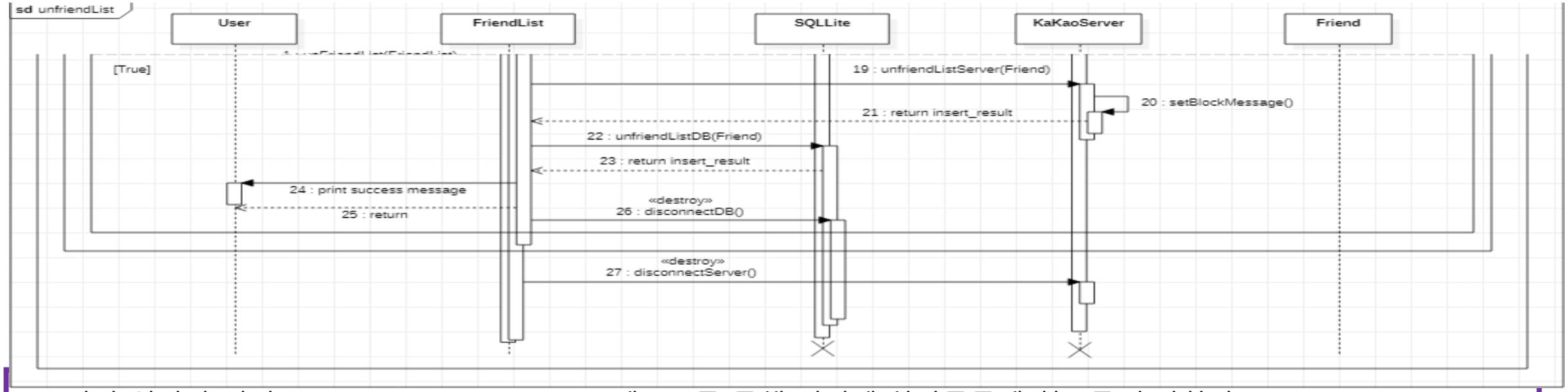
13. 경고 메시지를 출력 후에 종료한다.

U5. 친구 차단



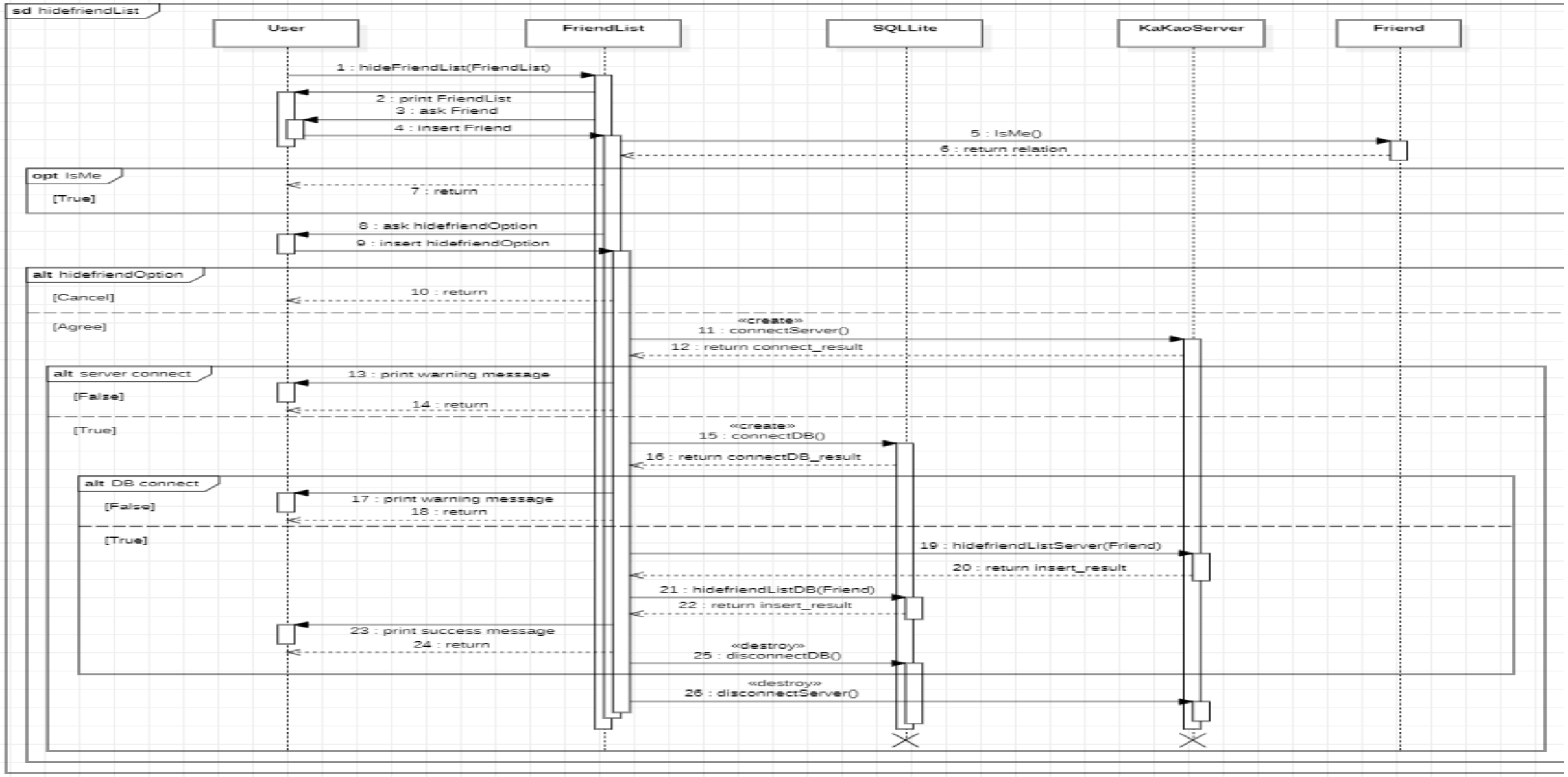
15. 서버 연결이 되었다면 connectDB()를 호출하여 SQLite에 연결을 시도한다.
16. SQLLite 연결결과인 connectDB_result(Boolean)을 반환받는다.
17. SQLLite 연결이 안된다면 경고메시지를 출력한다.
18. 그리고 종료한다.

U5. 친구 차단

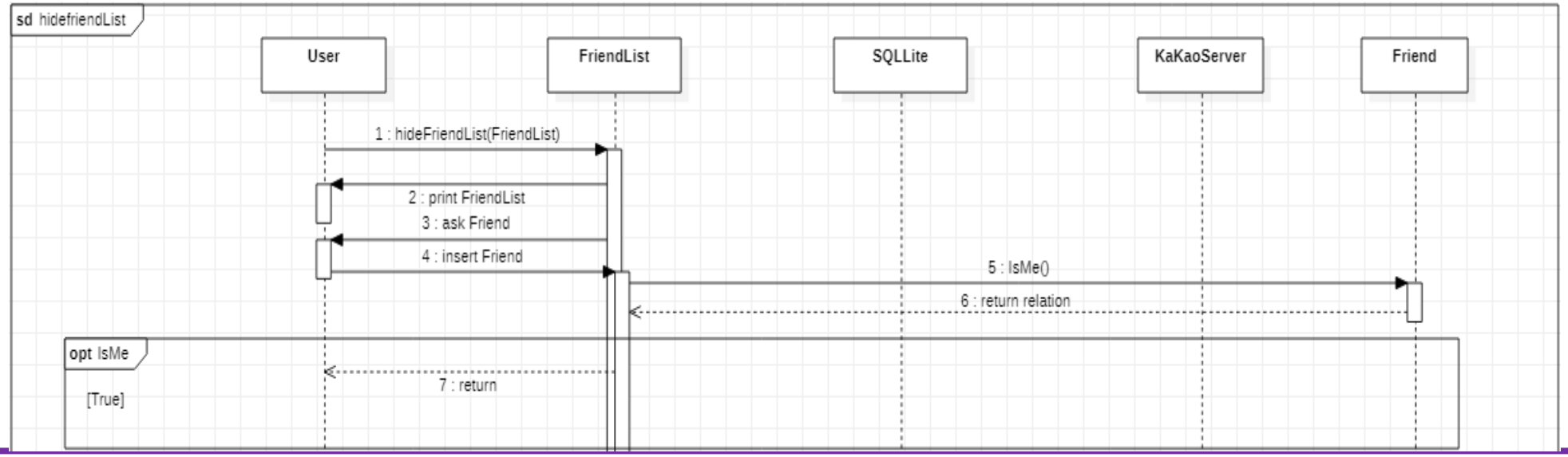


19. 서버 연결이 되면 unfriendListServer(Friend) 메소드를 통해 서버에 차단목록에 친구를 추가한다.
20. setBlockMessage()를 내부 호출하여 해당 친구가 사용자에게 메시지를 못보내도록 서버에서 설정한다.
21. 차단결과 insert_result(boolean)를 반환한다.
22. unfriendListDB(Friend) 메소드를 호출하여 SQLite에도 차단목록에 해당 친구를 추가한다.
23. 차단결과 insert_result(boolean)를 반환한다
24. SQLLite 연결을 disconnectDB() 메소드를 호출하여 서버 연결을 종료한다.
25. 서버 연결을 disconnectServer() 메소드를 호출하여 서버 연결을 종료한다.

U6. 친구 숨김

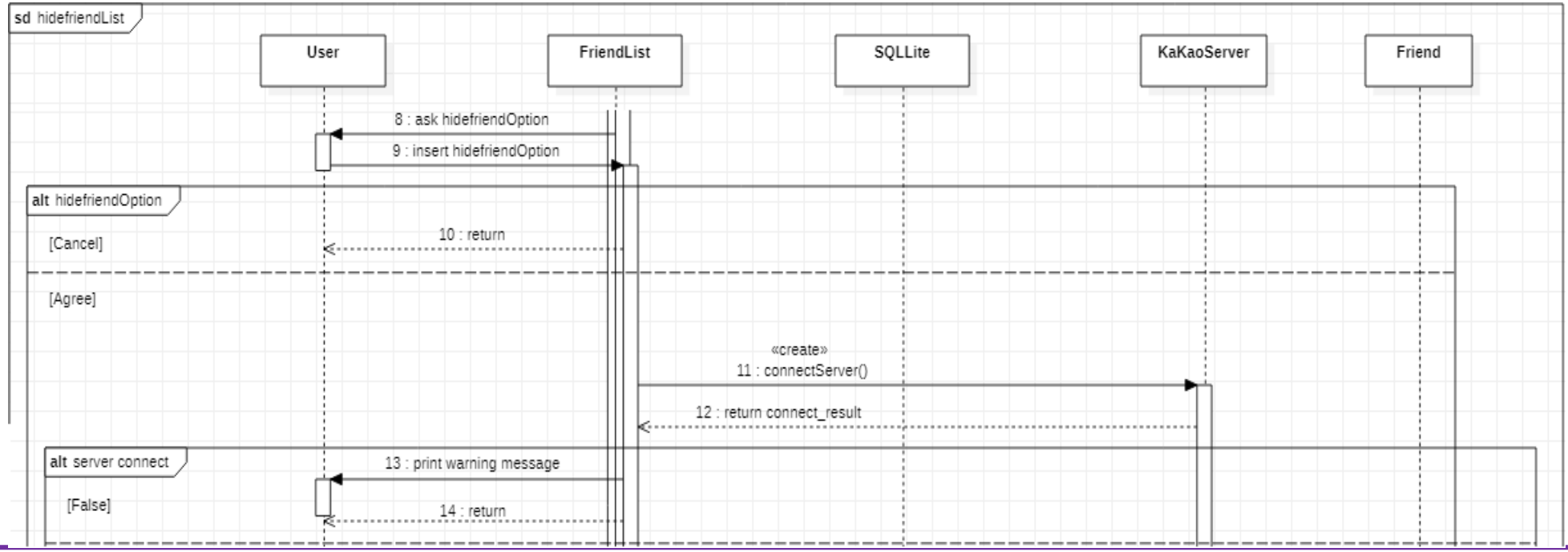

sd hidefriendList
User
FriendList
SQLLite
KaKaoServer
Friend
1 : hideFriendList(FriendList)
2 : print FriendList
3 : ask Friend
4 : insert Friend
5 : IsMe()
6 : return relation
opt IsMe
[True]
7 : return
8 : ask hidefriendOption
9 : insert hidefriendOption
alt hidefriendOption
[Cancel]
10 : return
[Agree]
«create»
11 : connectServer()
12 : return connect_result
alt server connect
[False]
13 : print warning message
14 : return
[True]
«create»
15 : connectDB()
16 : return connectDB_result
alt DB connect
[False]
17 : print warning message
18 : return
[True]
19 : hidefriendListServer(Friend)
20 : return insert_result
21 : hidefriendListDB(Friend)
22 : return insert_result
23 : print success message
24 : return
«destroy»
25 : disconnectDB()
«destroy»
26 : disconnectServer()

U6. 친구 숨김



1. hideFriendList(FriendList) 메소드를 호출하여 차단 기능을 실행한다.
2. 친구목록 FriendList(ArrayList<Friend>)을 통해 친구들을 출력한다.
3. 출력된 친구를 바탕으로 친구 Friend 선택을 요청한다.
4. 친구(Friend)를 선택하여 입력한다
5. 선택한 친구가 나인지 아닌지 판단하는 IsMe() 메소드를 호출하고 결과 relation(boolean)을 반환받는다.
6~7. 선택한 친구(Friend)가 나인 경우, 즉 True라면 return(void)로 종료한다.

U6. 친구 숨김



8. 선택한 친구(Friend)가 자신이 아니라면 진짜 차단할 것인지 취소할 것인지 hidefriendOption을 요청한다.

9. hidefriendOption을 입력한다.

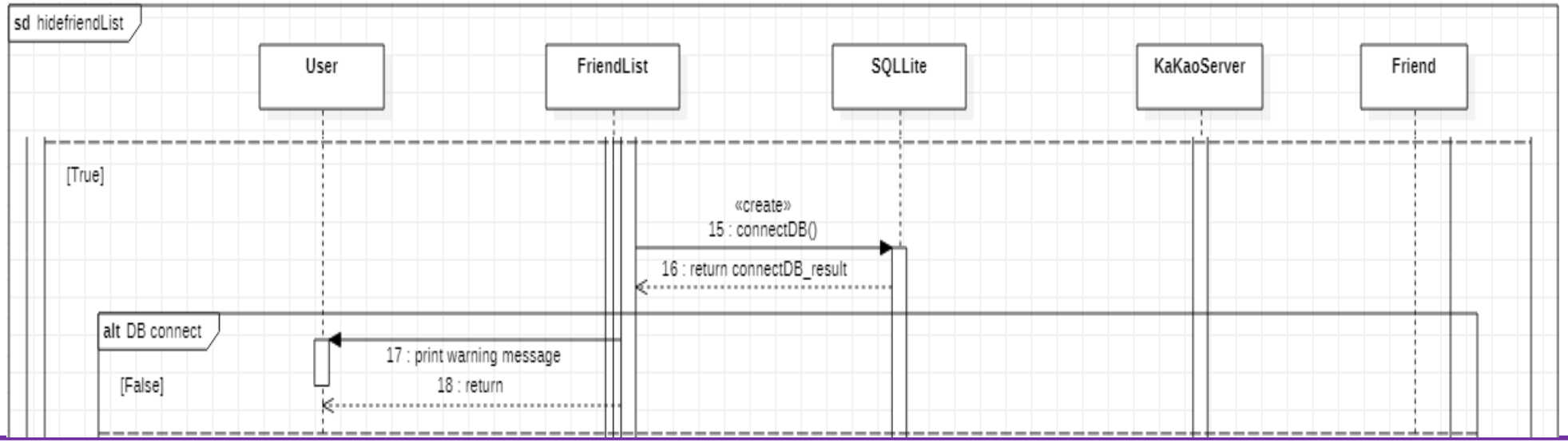
10. hidefriendOption이 Cancel이라면 return(void)를 통해 종료한다.

11. hidefriendOption이 Agree이라면 connectServer() 메소드를 통해 서버 연결을 시도한다.

12. 서버 연결이 안된다면 경고 메시지를 출력한다.

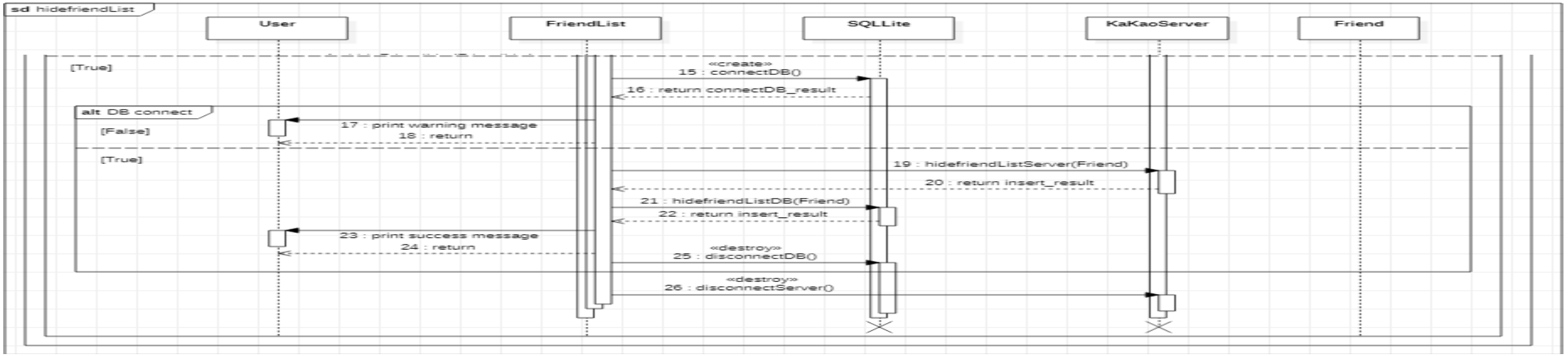
13. 경고 메시지를 출력 후에 종료한다.

U6. 친구 숨김



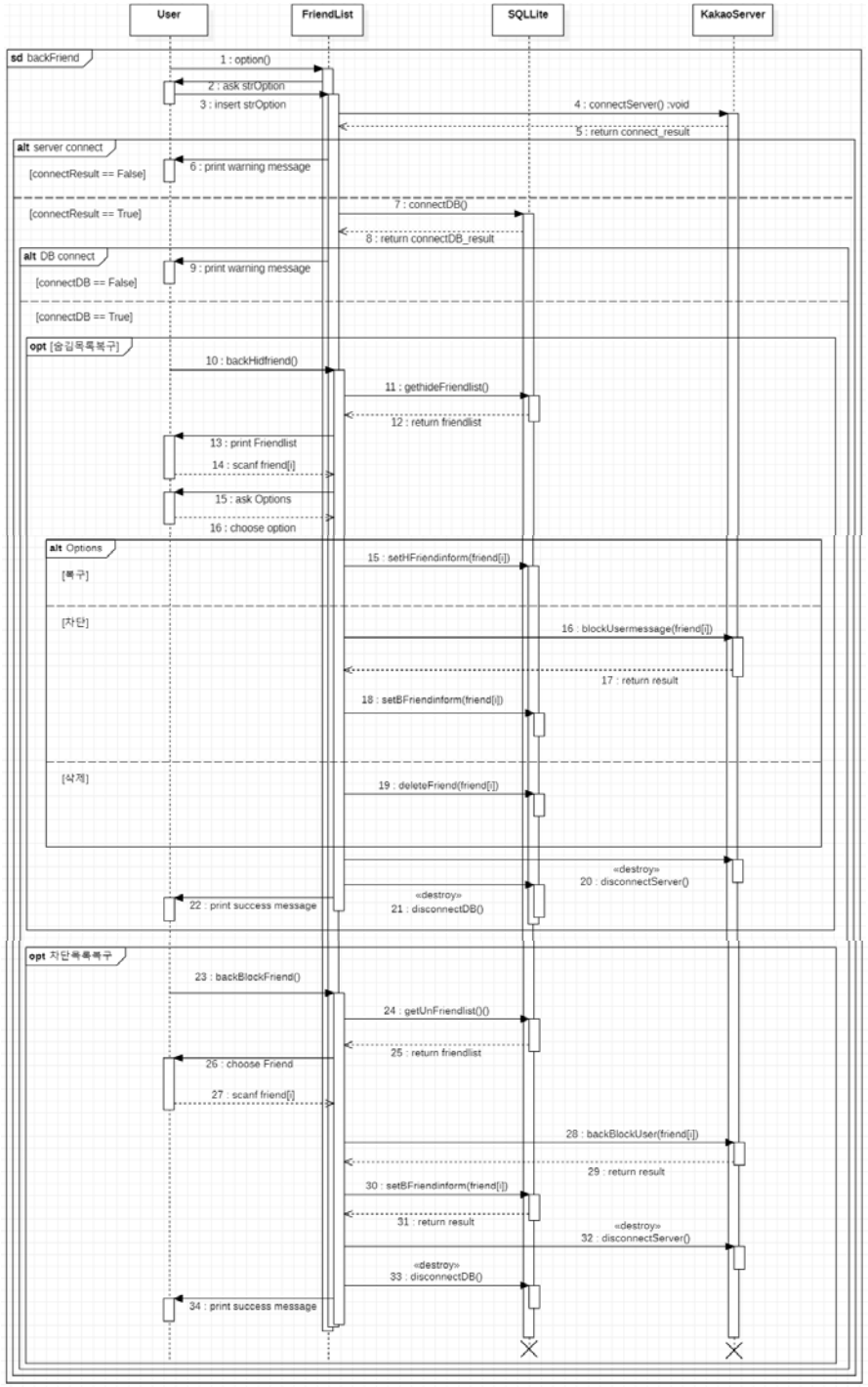
15. 서버 연결이 되었다면 connectDB()를 호출하여 SQLite에 연결을 시도한다.
16. SQLLite 연결결과인 connectDB_result(Boolean)을 반환받는다.
17. SQLLite 연결이 안된다면 경고메시지를 출력한다.
18. 그리고 종료한다.

U6. 친구 숨김



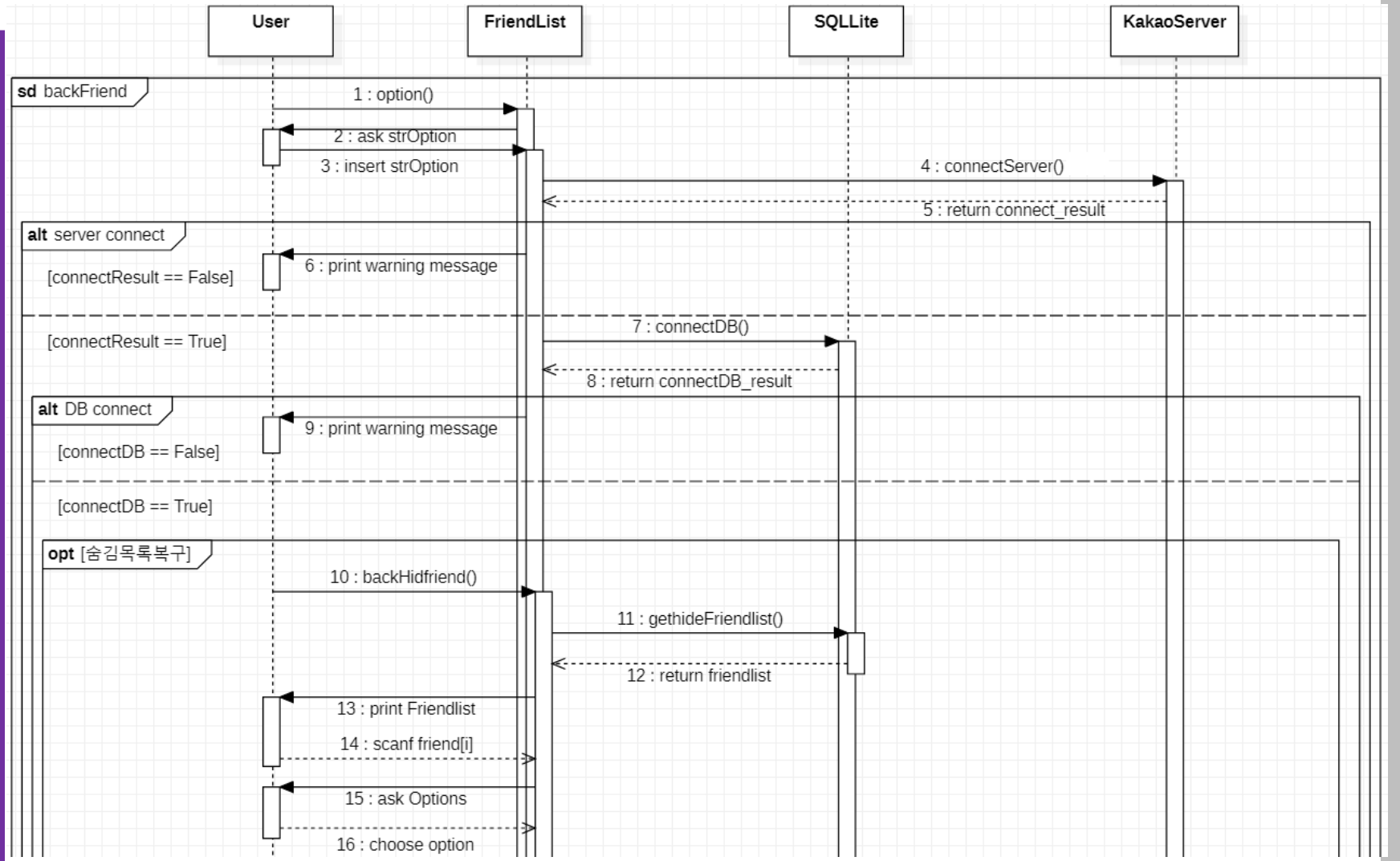
19. 서버 연결이 되면 hidefriendListServer(Friend) 메소드를 통해 서버에 숨김목록에 친구를 추가한다.
20. 차단결과 insert_result(boolean)를 반환한다.
21. hidefriendListDB(Friend) 메소드를 호출하여 SQLite에도 숨김목록에 해당 친구를 추가한다.
22. 차단결과 insert_result(boolean)를 반환한다
23. SQLLite 연결을 disconnectDB() 메소드를 호출하여 서버 연결을 종료한다.
24. 서버 연결을 disconnectServer() 메소드를 호출하여 서버 연결을 종료한다.

U7. 친구목록복구


User
FriendList
SQLLite
KakaoServer
sd backFriend
1 : option()
2 : ask strOption
3 : insert strOption
4 : connectServer() :void
5 : return connect_result
alt server connect
[connectResult == False]
6 : print warning message
[connectResult == True]
7 : connectDB()
8 : return connectDB_result
alt DB connect
[connectDB == False]
9 : print warning message
[connectDB == True]
opt [숨김목록복구]
10 : backHidfriend()
11 : gethideFriendlist()
12 : return friendlist
13 : print Friendlist
14 : scanf friend[i]
15 : ask Options
16 : choose option
alt Options
[복구]
15 : setHFriendinform(friend[i])
[차단]
16 : blockUsermessage(friend[i])
17 : return result
18 : setBFriendinform(friend[i])
[삭제]
19 : deleteFriend(friend[i])
«destroy»
20 : disconnectServer()
«destroy»
21 : disconnectDB()
22 : print success message
opt 차단목록복구
23 : backBlockFriend()
24 : getUnFriendlist()()
25 : return friendlist
26 : choose Friend
27 : scanf friend[i]
28 : backBlockUser(friend[i])
29 : return result
30 : setBFriendinform(friend[i])
31 : return result
«destroy»
32 : disconnectServer()
«destroy»
33 : disconnectDB()
34 : print success message

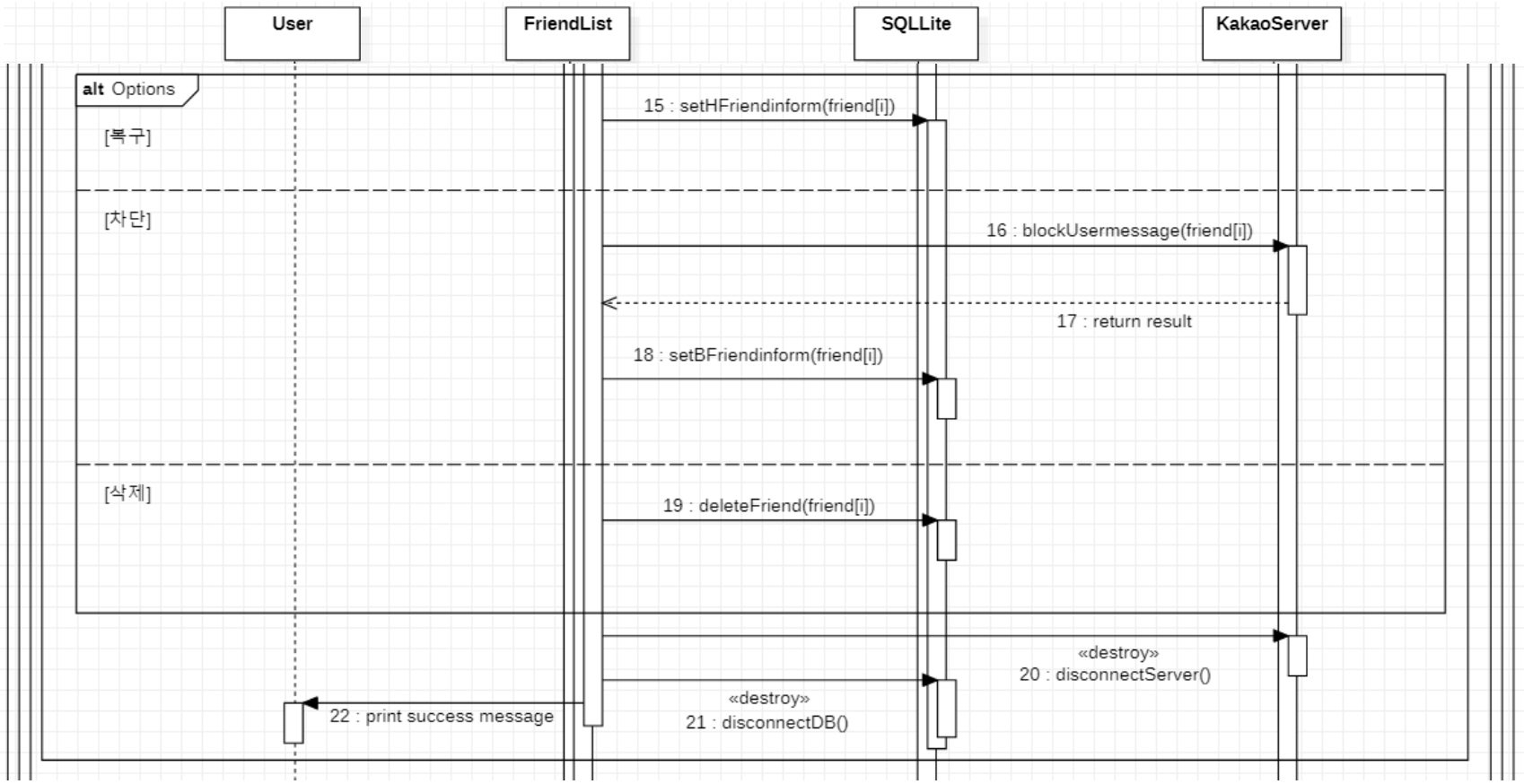
U7. 친구목록복구

1. Option()함수 호출(void), option 메소드 내부에서 strOption 변수를 입력받음(scanf strOption). 차단친구복구 인지 숨김친구 복구인지 확인(option메소드 내에서). 카카오서버에 연결시도, DB에도 연결을 시도한다. 리턴값은(boolean).
2. 각각 False인 경우 오류메세지를 반환하고 True인 경우에는 strOption 값을 통해 숨김목록복구 혹은 차단목록 복구중 하나를 선택한다.
3. 숨김목록 복구는 backHidefriend()를 사용하고(void) DB에서 숨김목록을 받아온 후(ArrayList)에 친구를 출력하고 친구한 명을 선택한다. Ui로 세가지 옵션중 하나를 선택하고 (Options변수 int) 1번이면 복구, 2번이면 차단, 3번이면 삭제를 실행한다.



U7. 친구목록복구

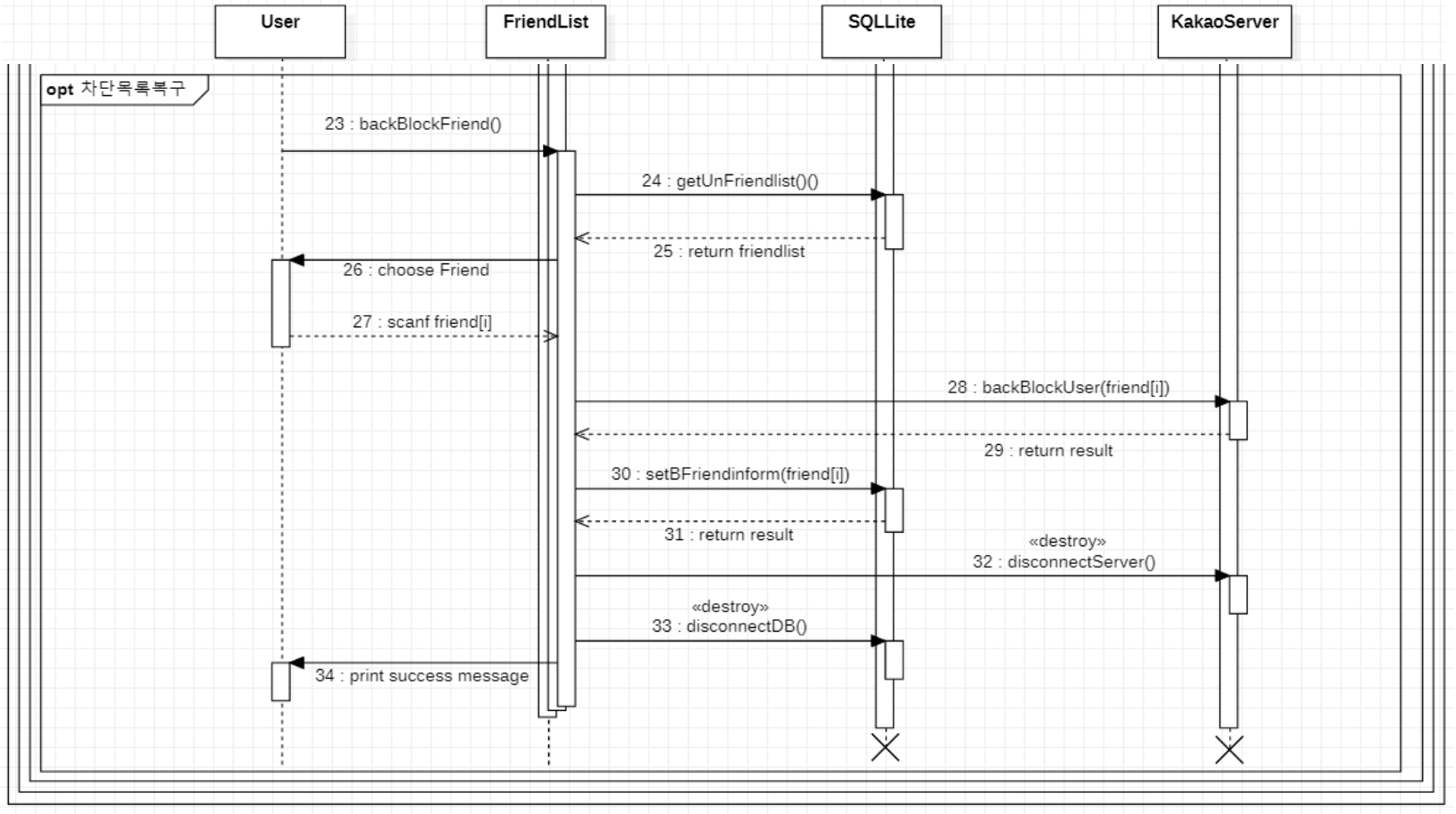
4. 선택한 친구를 파라미터로 DB에 내장되어있는 setHFriendinform(friend[i])함수를 호출하면 해당 친구정보가 업데이트된다.(로직제외, frien의경우 ArrayList형태로 파라미터넘김 , void형)
5.차단이 선택된 경우 blockUsermessage()함수를 통해 호출하고 (boolean) 파라미터는 ArrayList 형태로 넣어준다. DB에도 같은 파라미터 형태로 setBFriendinform()으로 업데이트 해준다(로직제외).
6. 삭제의 경우도 같은 파라미터 형태로 DB에 삭제 친구를 넘겨준다.
7. 작업이 마무리되면 void형태로 카카오서버, DB 를 연결해제후 성공메세지를 출력한다.



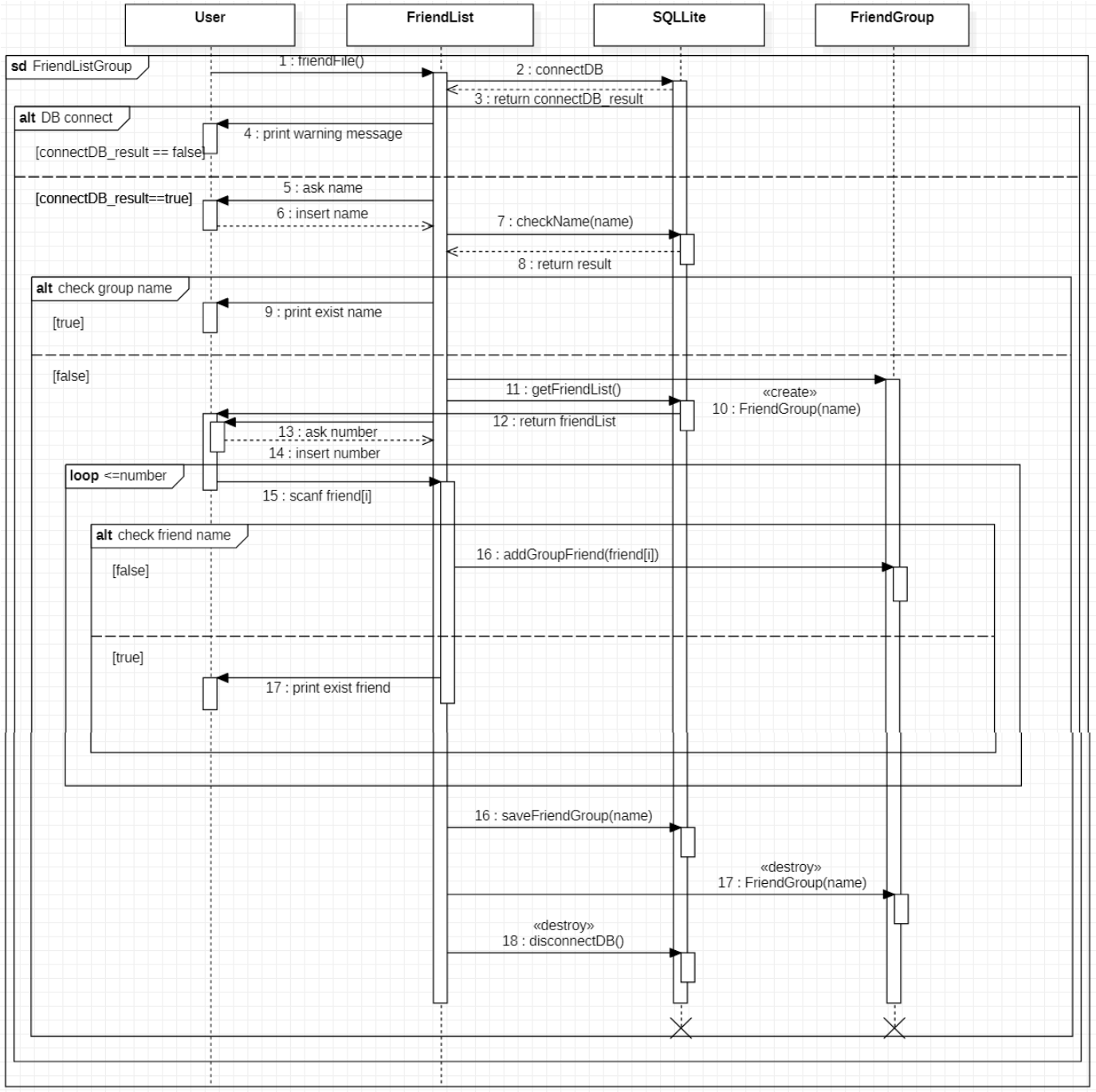
U7. 친구목록복구

8. 차단을 선택하면 backBlockFriend()를 실행한다. 리턴값은 void이다. 해당 메소드 내에 getUnFriendlist()를 실행하고 ArrayList형태로 친구리스트를 반환한다. 반환한 리스트중 한명의 친구를 입력받아 카카오서버에 내장되어있는 backBlockUser() 를 호출한다. 파라미터는 ArrayList형태의 선택한 친구이다. Boolean값을 리턴해준다.

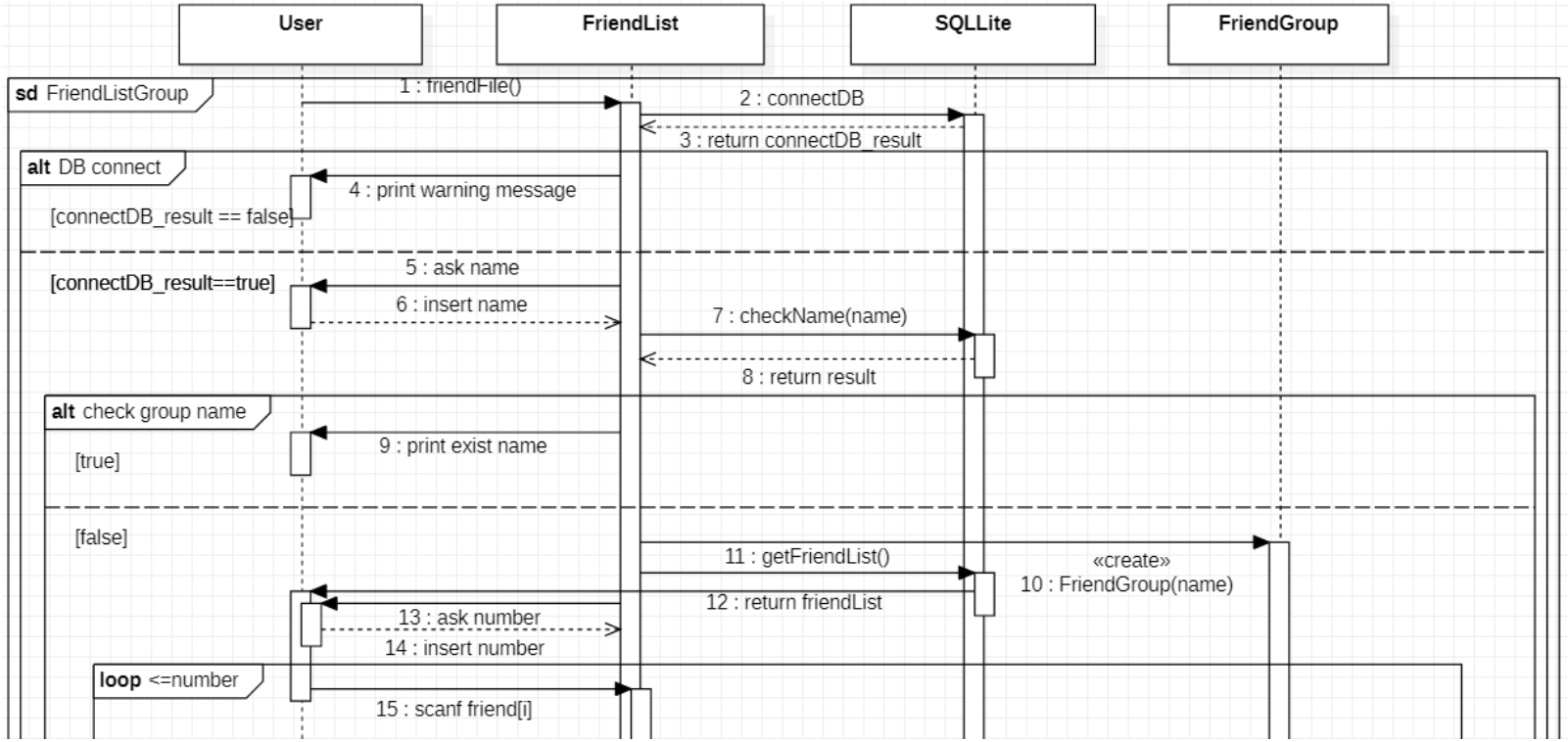
9. 선택된 ArrayList 친구를 매개변수로 setBFriendinform()를 호출한다. 해당 함수는 DB에 내장되어있다. boolean값을 리턴받고 서버와 DB연결을 해제한후 성공메세지를 출력한다. (리턴값이 FALSE인경우는 일단 제외)



U8. 친구목록그룹화

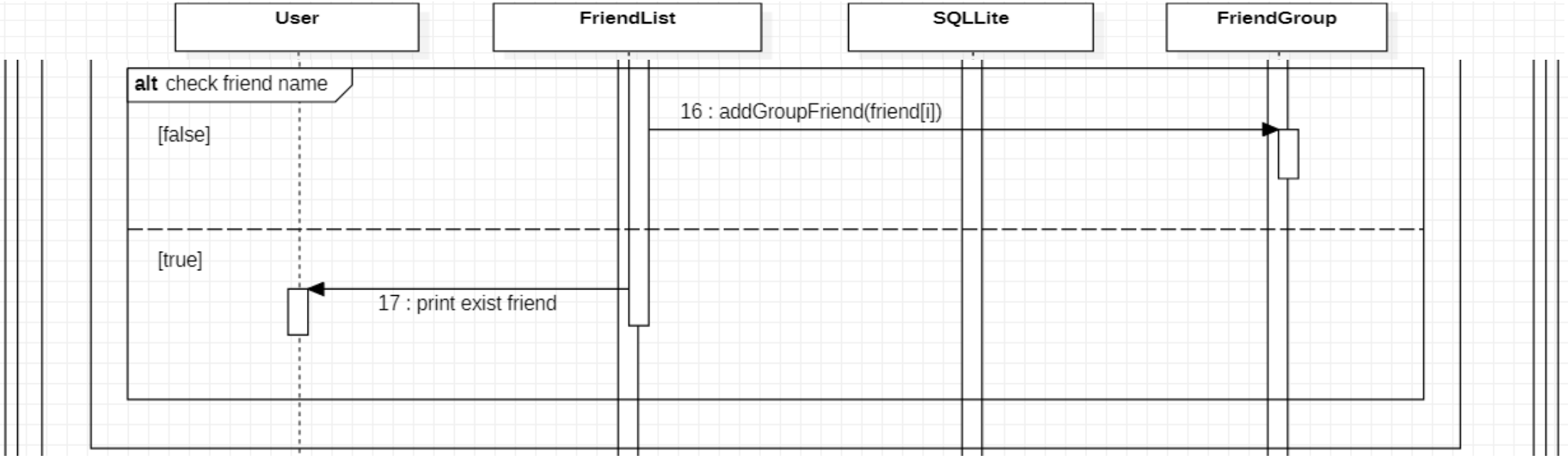

User
FriendList
SQLLite
FriendGroup
sd FriendListGroup
1 : friendFile()
2 : connectDB
3 : return connectDB_result
alt DB connect
4 : print warning message
[connectDB_result == false]
5 : ask name
[connectDB_result==true]
6 : insert name
7 : checkName(name)
8 : return result
alt check group name
9 : print exist name
[true]
[false]
11 : getFriendList()
«create»
10 : FriendGroup(name)
12 : return friendList
13 : ask number
14 : insert number
loop <=number
15 : scanf friend[i]
alt check friend name
16 : addGroupFriend(friend[i])
[false]
[true]
17 : print exist friend
16 : saveFriendGroup(name)
«destroy»
17 : FriendGroup(name)
«destroy»
18 : disconnectDB()

U8. 친구목록 그룹화



1. 사용자가 friendFile()함수를 호출한다(void).
2. friendFile()내에 DB연결 함수가 내장되어있다. boolean타입을 리턴한다.
3. 실패하면 경고메세지를 성공한다면 사용자에게 그룹이름(스트링변수 name)을 요청한다.
4. DB에 내장되어있는 checkName()함수를 이용하여(리턴값 boolean) 이름을 체크하고 그룹이름이 중복된다면 이미존재하는 메시지를 출력한다.
5. 존재하지 않는다면 FriendGroup(name)객체를 생성하고, DB에 친구목록을 요청한다.(getFriendList(), ArrayList 반환)
6. number변수를 생성하고 몇 명을 입력받을지 선택한다.
7. number 숫자만큼 for루프를 돌며 친구목록을 한명씩 입력받는다.

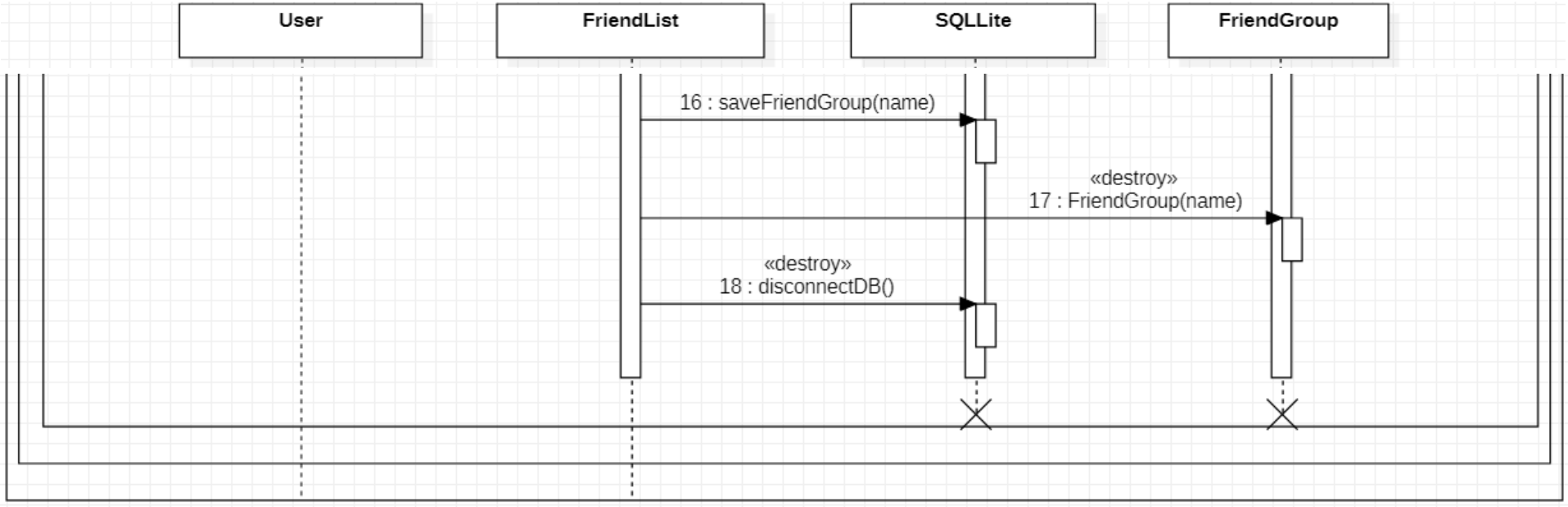
U8. 친구목록그룹화



8. 만약이름이 존재한다면 exist friend 를 출력하고 number를 채울때까지 루프를 돌게된다.

9. 중복되는 친구가 없다면 addGroupFriend(friend[i])를 통해 객체에 친구를 추가한다. void형태고 파라미터는 ArrayList 형태로 전달한다.

U8. 친구목록그룹화



10. 생성된 FriendGroup(name)객체를 saveFriendGroup(객체이름)으로 저장한다. void형태
11. FriendGroup(name)객체를 파괴한다.
12. DB연결을 끊는다.

Q&A