프로그래밍언어

**Mini Project**

12조

2015312305 박원호

2014313366 홍기원

2014311675 현민지

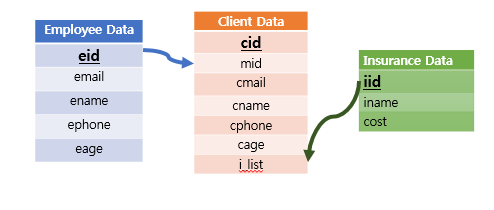
1. **프로젝트 목표**

대표적인 Functional Language인 Lisp을 이용하여 보험 회사 Database 관리 Library를 구현함으로써 Functional Language에 대한 이해를 높이고 Programming language의 폭을 넓힐 수 있다.

1. **프로그램 설명**
   1. 보험 회사 Database 관리 프로그램

보험 회사의 고객(Client), 직원(Employee), 보험(Insurance)의 데이터를 입력, 수정, 추출, 삭제할 수 있는 Database 관리 프로그램이다.

* 1. Database 구조
     1. 전체 구조



* + 1. 직원(Employee)

-eid : Employee ID 약자로 사원번호이다. Employee Data의 Primary Key이다. String type으로 E01, E02와 같은 형식을 가지고 있다.

-email : Employee Mail의 약자로 사원 이메일 주소이다. String을 Type으로 한다.

-ename : Employee Name으로 사원 이름이다. String을 Type으로 한다.

-ephone : Employee Phone으로 사원 전화번호이다. String을 Type으로 한다.

-eage : Employee Age로 사원의 나이이다. Integer를 Type으로 한다.

* + 1. 보험(Insurance)

-iid : Insurance ID 약자로 보험상품 번호이다. Insurance Data의 Primary Key이다. String type으로 I01, I02와 같은 형식을 가지고 있다.

-iname : Insurance Name으로 보험상품의 이름이다. String을 Type으로 갖는다.

-icost : Insurance Cost로 보험 상품의 한달 요금이다. Integer를 Type으로 한다.

* + 1. 고객(Client)

-cid : Client ID 약자로 고객 번호이다. Client Data의 Primary Key이다. String type으로 C01, C02와 같은 형식을 가지고 있다.

-mid : Manager ID 약자로 담당 사원 번호이다. Employee Data의 eid를 Foreign Key로 가진다.

-cmail : Client Mail의 약자로 고객 이메일 주소이다. String을 Type으로 한다.

-cname : Client Name으로 고객 이름이다. String을 Type으로 한다.

-cphone : Client Phone으로 고객 전화번호이다. String을 Type으로 한다.

-cage : Client Age로 고객의 나이이다. Integer를 Type으로 한다.

-i\_list : Insurance List로 고객이 가입한 상품으로 이루어진 list이다. Insurance Data의 iid를 Foreign Key로 갖는다.

* 1. 기능 및 Library

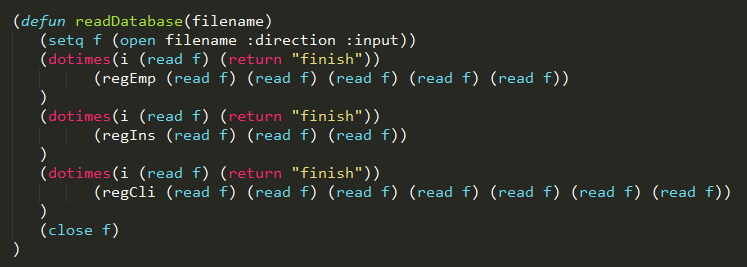
기능은 크게 Database로부터 데이터 가져오기 및 내보내기, 데이터 추가 및 삭제, 검색으로 나눌 수 있다.

* + 1. 데이터 가져오기 및 내보내기
       1. 데이터 구성



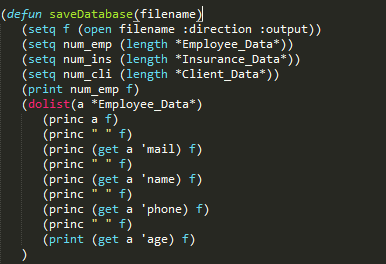
데이터는 크게 Employee, Insurance, Client 세부분으로 나눠진다. 각 부분의 맨 첫 줄은 데이터tuple의 개수로 이루어져 있고 그 다음줄부터 차례로 tuple이 이어진다. Tuple은 Enter로 구분되며 한 tuple의 attribute들은 space로 구분된다.

* + - 1. 데이터 가져오기



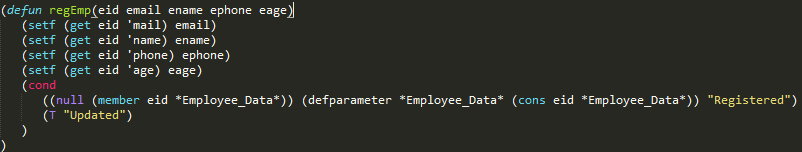
데이터를 가지고 있는 파일인 filename을 인자로 받아 실행한다. 데이터의 tuple개수를 읽어와 그에 해당하는 만큼 iteration을 이용하여 data를 가져오게 된다. 그리고 각각 regEmp, regIns, regCli 함수를 호출하여 Employee\_Data, Insurance\_Data, Client\_Data List를 구성한다.

* + - 1. 데이터 내보내기



먼저 Employee\_Data의 길이를 계산하여 Data 맨 위에 출력해주고, list 길이만큼 iteration을 사용하여 tuple의 data를 추출한다. 이를 Insurance, Client\_Data에서도 반복한다.

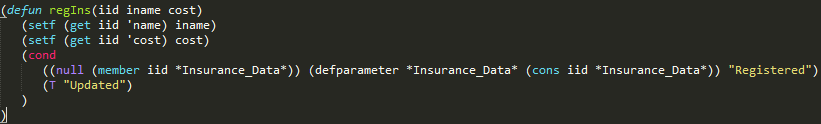
* + 1. 데이터 추가 및 삭제
       1. regEmp : 직원 데이터 추가



함수 regEmp를 사용하여 직원의 데이터를 추가할 수 있다. regEmp는 함수 인자로 직원의 id, 메일주소, 이름, 전화번호, 나이를 입력 받아 \*Employee\_Data\*에 저장한다. 각각의 직원은 id를 통해 구별하므로, 만약 인자로 받은 id가 \*Employee\_Data\*에 존재하지 않는다면 새로 추가하고 registered 되었다는 메시지를 출력한다. 만약 이미 존재하는 id를 인자로 받았을 경우, 새롭게 추가하지 않고 기존에 존재하는 직원의 정보를 갱신한다. 이후 updated 되었다는 메시지를 출력한다.

id가 \*Employee\_Data\*에 존재하는지 하지 않는지의 여부는 member를 사용하여 판단한다. \*Employee\_Data\*라는 list에 id가 포함되어 있지 않다면 nil을 리턴하고, 포함되어 있다면 nil이 아닌 값을 리턴하므로 이를 통해 존재 여부를 판단하게 된다. 따라서 nil일 경우 cons를 사용하여 id를 list에 넣고, nil이 아닐 경우 넣지 않는다.

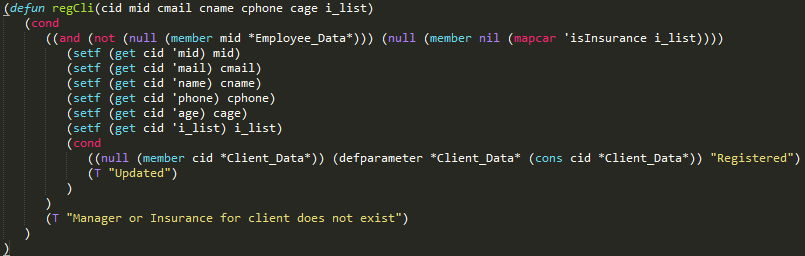
* + - 1. regIns : 보험 데이터 추가



함수 regIns를 사용하여 보험의 데이터를 추가할 수 있다. 인자로 보험의 id, 보험의 이름, 보험의 비용을 입력 받아 \*Insurance\_Data\*에 저장한다. regEmp와 마찬가지로, 인자로 받은 id가 \*Insurance\_Data\*에 존재하지 않는다면 새로 추가하고 registered 되었다는 메시지를 출력한다. id가 \*Insurance\_Data\*에 존재할 경우, 기존에 존재하는 보험의 정보를 갱신하고 updated 되었다는 메시지를 출력한다.

id가 \*Insurance\_Data \*에 존재하는지 여부는 regEmp와 마찬가지로member를 사용하여 판단한다. \* Insurance \_Data\* 라는 list에 id가 포함되어 있지 않다면 nil을 리턴하고, 포함되어 있다면 nil이 아닌 값을 리턴하므로 이를 통해 존재 여부를 판단하게 된다. 따라서 nil일 경우 cons를 사용하여 id를 list에 넣고, nil이 아닐 경우 넣지 않는다.

* + - 1. 고객 데이터 추가



함수 regCli를 사용하여 고객의 데이터를 추가할 수 있다. 인자로는 고객의 id, 고객을 담당하는 직원의 id, 메일주소, 이름, 전화번호, 나이, 가입한 보험의 목록을 받는다. regEmp나 regIns와 마찬가지로 인자로 받은 id가 \*Client\_Data\*에 존재하지 않는다면 새로 추가하고 registered 되었다는 메시지를 출력한다. id가 \* Client\_Data \*에 존재할 경우, 기존에 존재하는 보험의 정보를 갱신하고 updated 되었다는 메시지를 출력한다.

직원이나 보험과 다른 점은 고객의 경우, 고객을 담당하는 직원이나 고객이 가입한 보험의 정보 (DB의 foreign key)가 둘 중 하나라도 존재하지 않을 경우 \*Client\_Data\*에 데이터를 추가하거나 갱신할 수 없다는 것이다. 이 경우 매니저나 보험이 존재하지 않는다는 메시지를 출력하게 한다.

고객을 담당하는 직원이나 고객이 가입한 보험이 존재하지 않는 지에 대한 여부 역시 member를 사용하여 판단한다. member의 리턴 값이 nil값이 아니어야 존재한다는 의미이므로 and를 사용하여 둘 중 하나라도 nil값이 나온다면 조건을 충족하지 못하도록 한다. 고객을 담당하는 직원이 존재하는 지 여부는 regEmp에서의 반대로, not을 사용하여 판단할 수 있다. 다만 고객이 가입한 보험의 경우 list이기 때문에 list의 모든 요소들이 \*Insurance\_Data\*에 존재하는지 판단해야 한다. 이를 위해서 ‘isInsurance라는 새로운 함수를 만들었다.



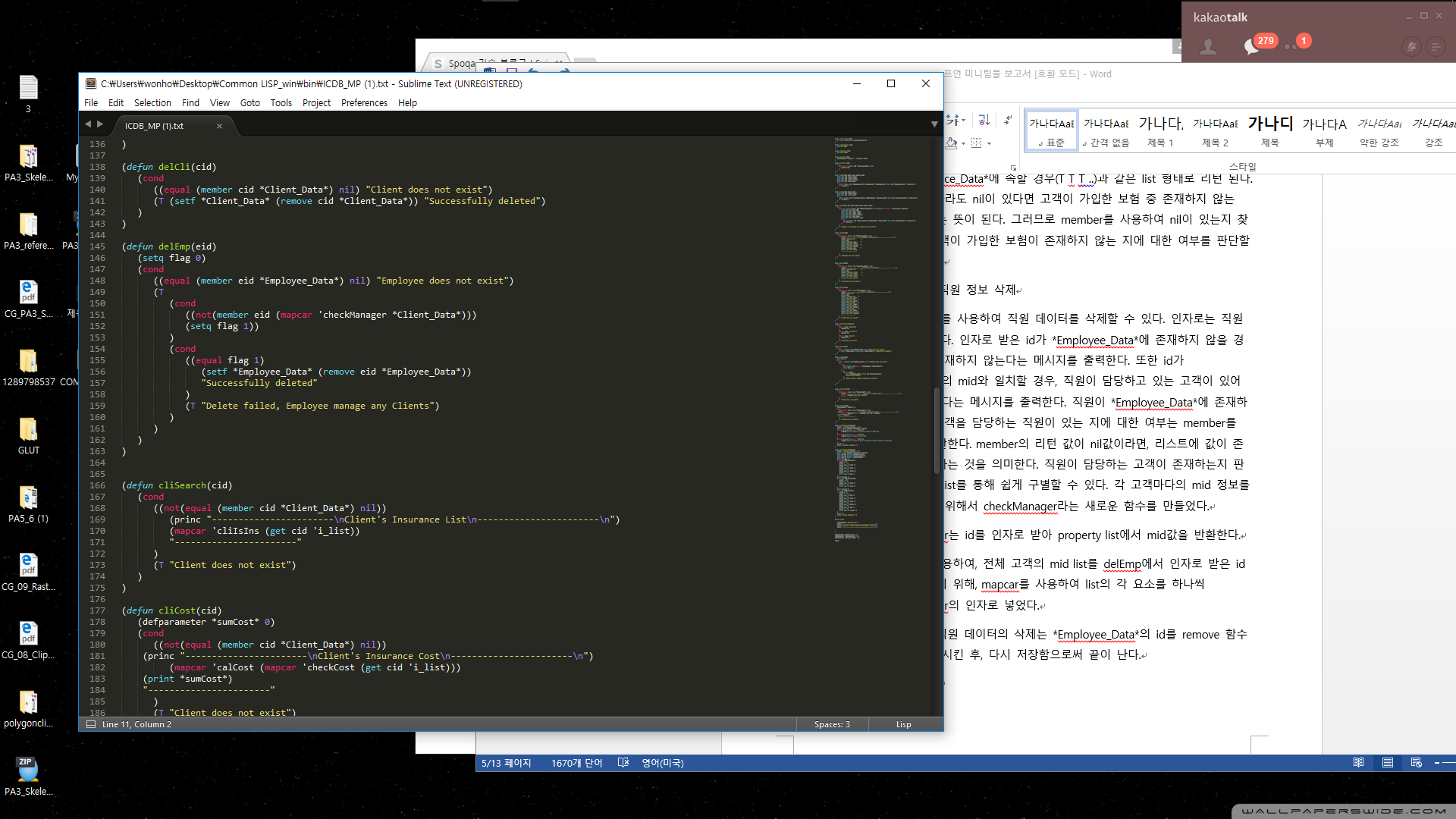
isInsurance는 id를 인자로 받아 그 id가 \*Insurance\_Data\*에 존재하는지를 판단한다. 존재할 경우 T를, 존재하지 않을 경우 nil을 리턴한다.

이 함수를 사용하여, 고객이 가입한 보험 list(i\_list)의 모든 요소들이 \*Insurance\_Data\*에 속하는지 판단하기 위해, mapcar를 사용하여 list의 각 요소를 하나씩 isInsurance의 인자로 넣었다.

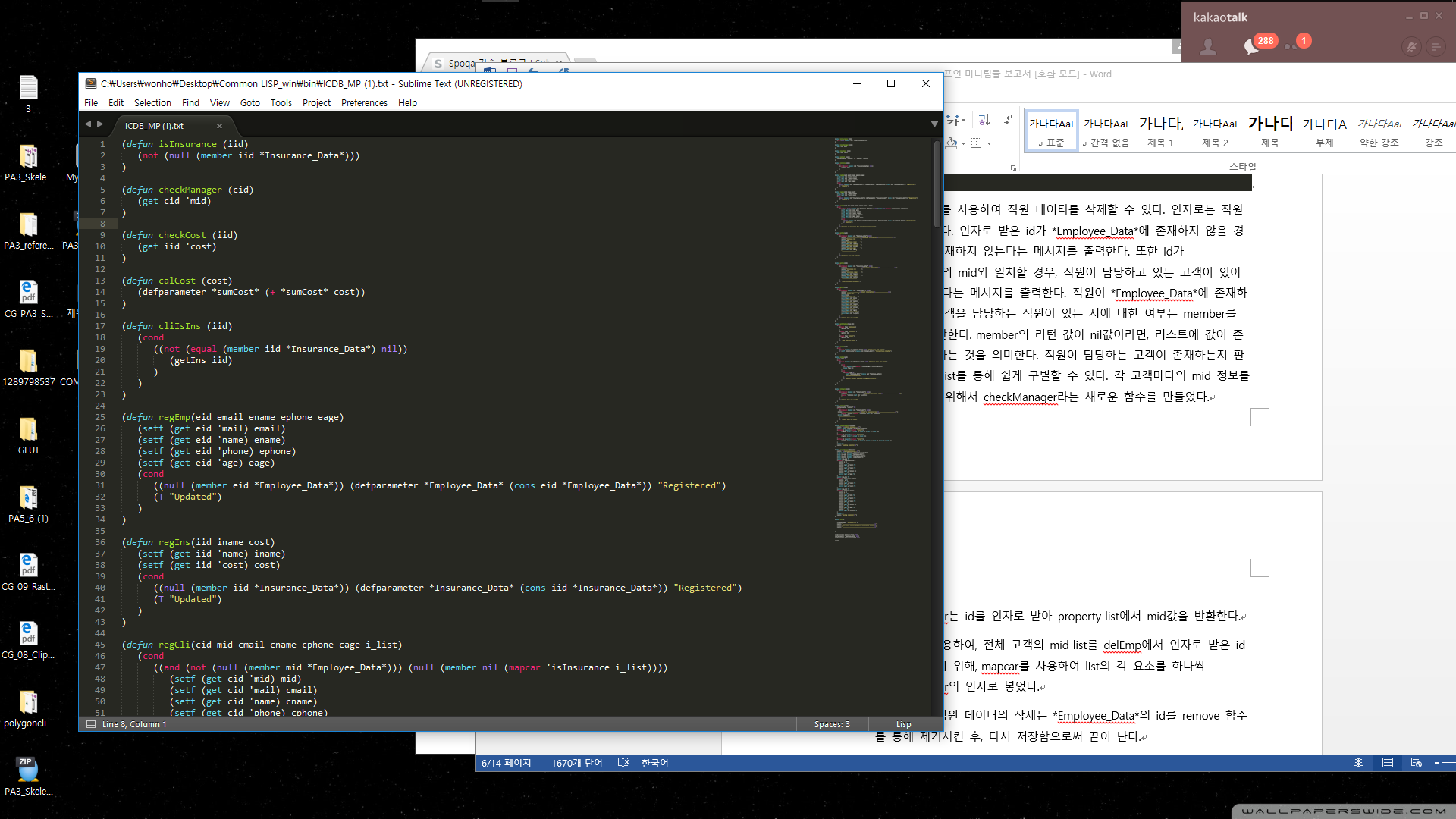


따라서 위 코드에서 (mapcar ‘isInsurance i\_list)는 만약 i\_list의 모든 요소들이 \*Insurance\_Data\*에 속할 경우(T T T ..)과 같은 list 형태로 리턴 된다. 그 list중 하나라도 nil이 있다면 고객이 가입한 보험 중 존재하지 않는 보험이 있다는 뜻이 된다. 그러므로 member를 사용하여 nil이 있는지 찾아본다면, 고객이 가입한 보험이 존재하지 않는 지에 대한 여부를 판단할 수 있게 된다.

* + - 1. delEmp : 직원 정보 삭제



함수 delEmp를 사용하여 직원 데이터를 삭제할 수 있다. 인자로는 직원의 id를 받는다. 인자로 받은 id가 \*Employee\_Data\*에 존재하지 않을 경우, 직원이 존재하지 않는다는 메시지를 출력한다. 또한 id가 \*Client\_Data\*의 mid와 일치할 경우, 직원이 담당하고 있는 고객이 있어 삭제할 수 없다는 메시지를 출력한다. 직원이 \*Employee\_Data\*에 존재하는지, 또는 고객을 담당하는 직원이 있는 지에 대한 여부는 member를 사용하여 판단한다. member의 리턴 값이 nil값이라면, 리스트에 값이 존재하지 않는다는 것을 의미한다. 직원이 담당하는 고객이 존재하는지 판단할 경우에 list를 통해 쉽게 구별할 수 있다. 각 고객마다의 mid 정보를 list로 만들기 위해서 checkManager라는 새로운 함수를 만들었다.

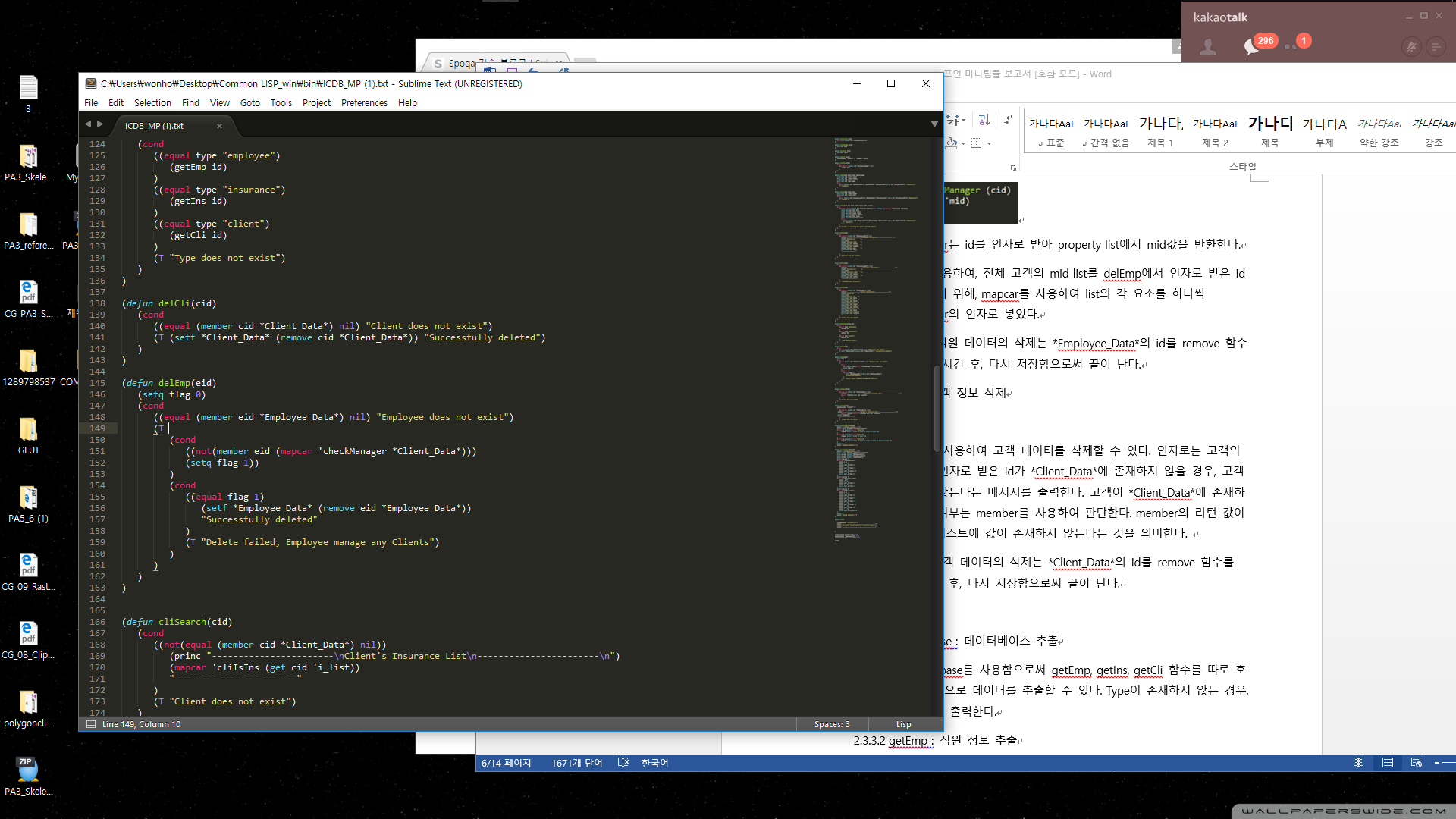


checkManager는 id를 인자로 받아 property list에서 mid값을 반환한다.

이 함수를 사용하여, 전체 고객의 mid list를 delEmp에서 인자로 받은 id와 비교해주기 위해, mapcar를 사용하여 list의 각 요소를 하나씩 checkManager의 인자로 넣었다.

최종적으로 직원 데이터의 삭제는 \*Employee\_Data\*의 id를 remove 함수를 통해 제거시킨 후, 다시 저장함으로써 끝이 난다.

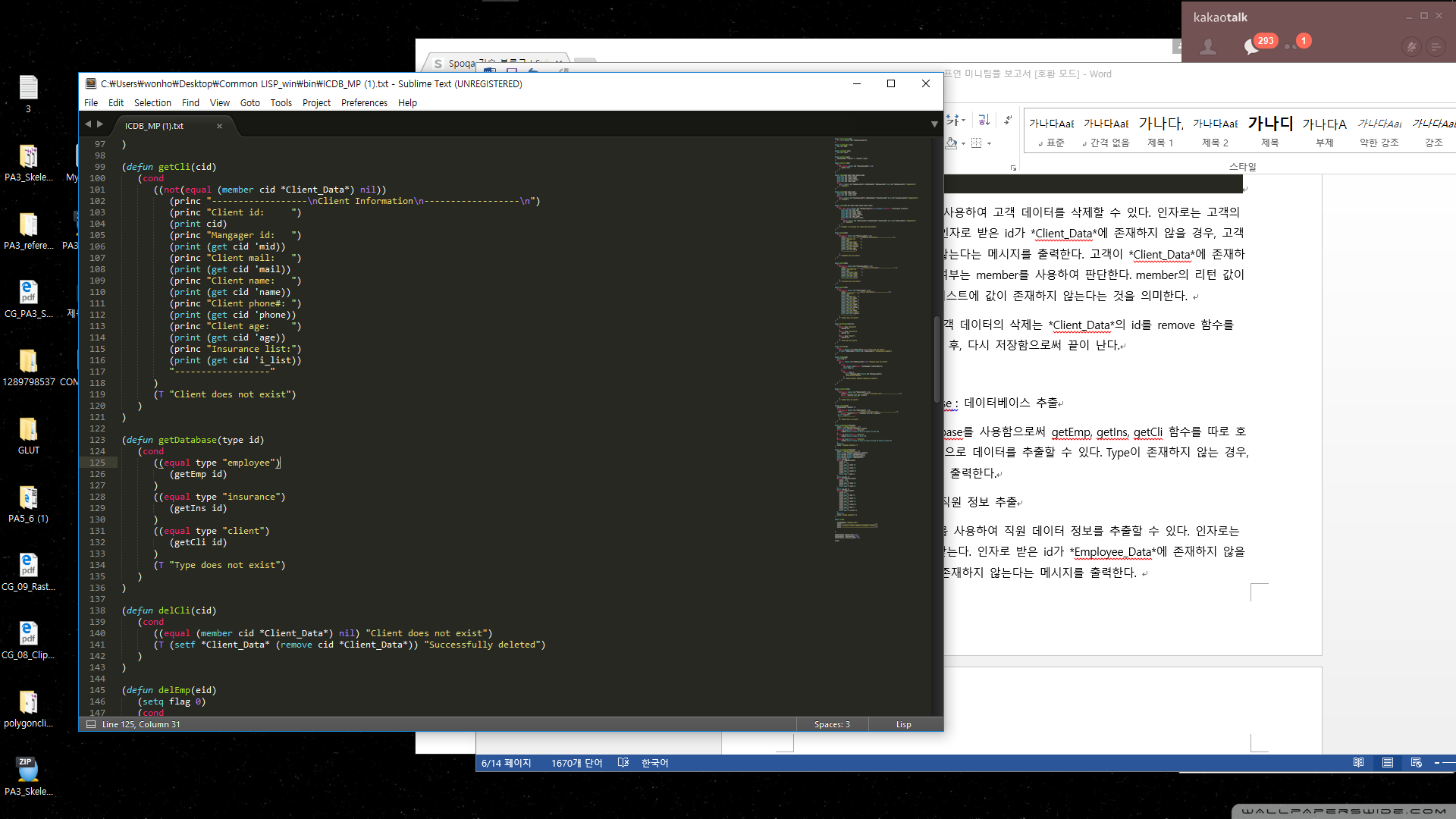
* + - 1. delCli : 고객 정보 삭제



함수 delCli를 사용하여 고객 데이터를 삭제할 수 있다. 인자로는 고객의 id를 받는다. 인자로 받은 id가 \*Client\_Data\*에 존재하지 않을 경우, 고객이 존재하지 않는다는 메시지를 출력한다. 고객이 \*Client\_Data\*에 존재하는지에 대한 여부는 member를 사용하여 판단한다. member의 리턴 값이 nil값이라면, 리스트에 값이 존재하지 않는다는 것을 의미한다.

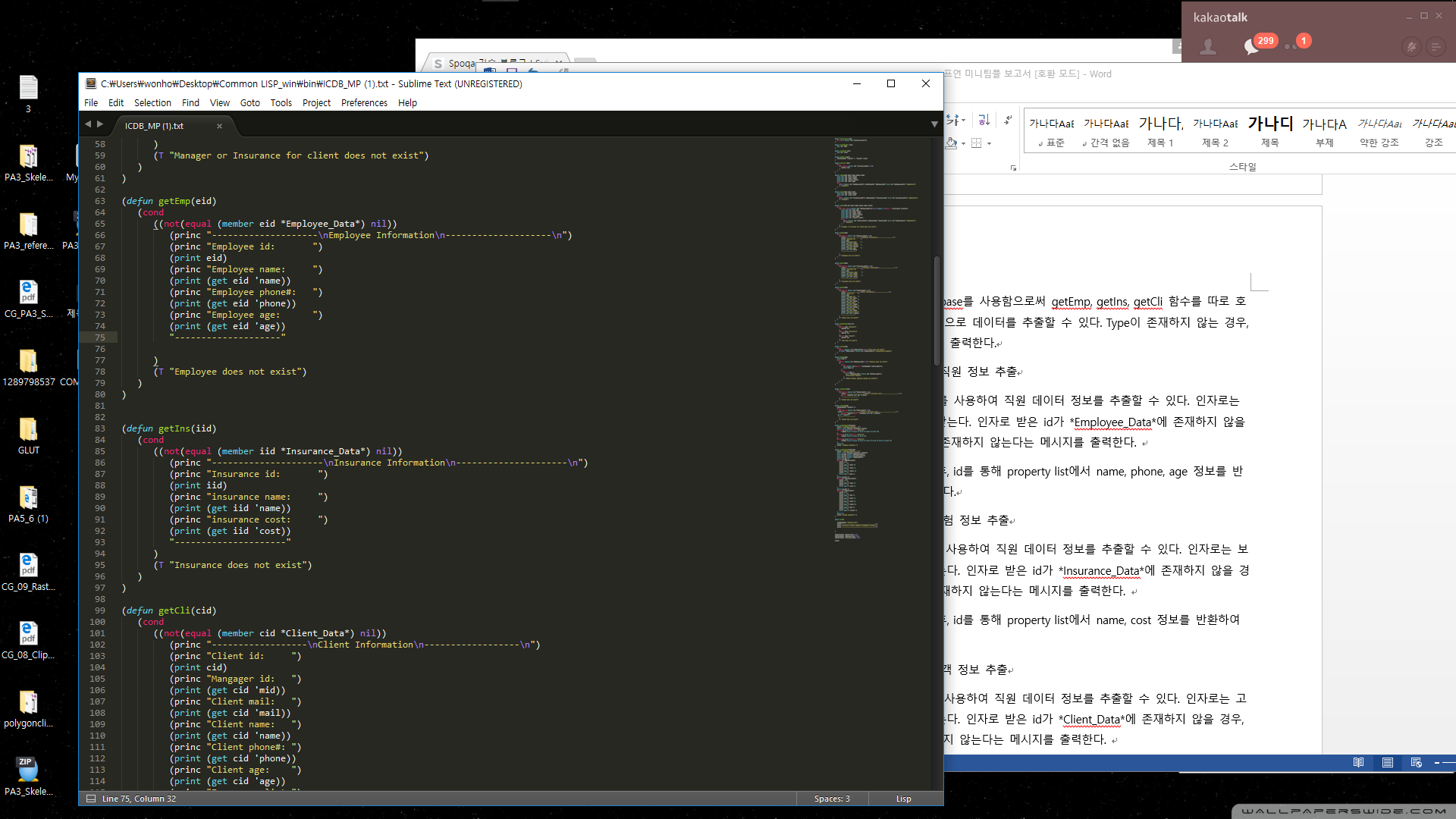
최종적으로 고객 데이터의 삭제는 \*Client\_Data\*의 id를 remove 함수를 통해 제거시킨 후, 다시 저장함으로써 끝이 난다.

* + 1. 검색
       1. getDatabase : 데이터베이스 추출



함수 getDatabase를 사용함으로써 getEmp, getIns, getCli 함수를 따로 호출하여 유동적으로 데이터를 추출할 수 있다. Type이 존재하지 않는 경우, 오류 메시지를 출력한다.

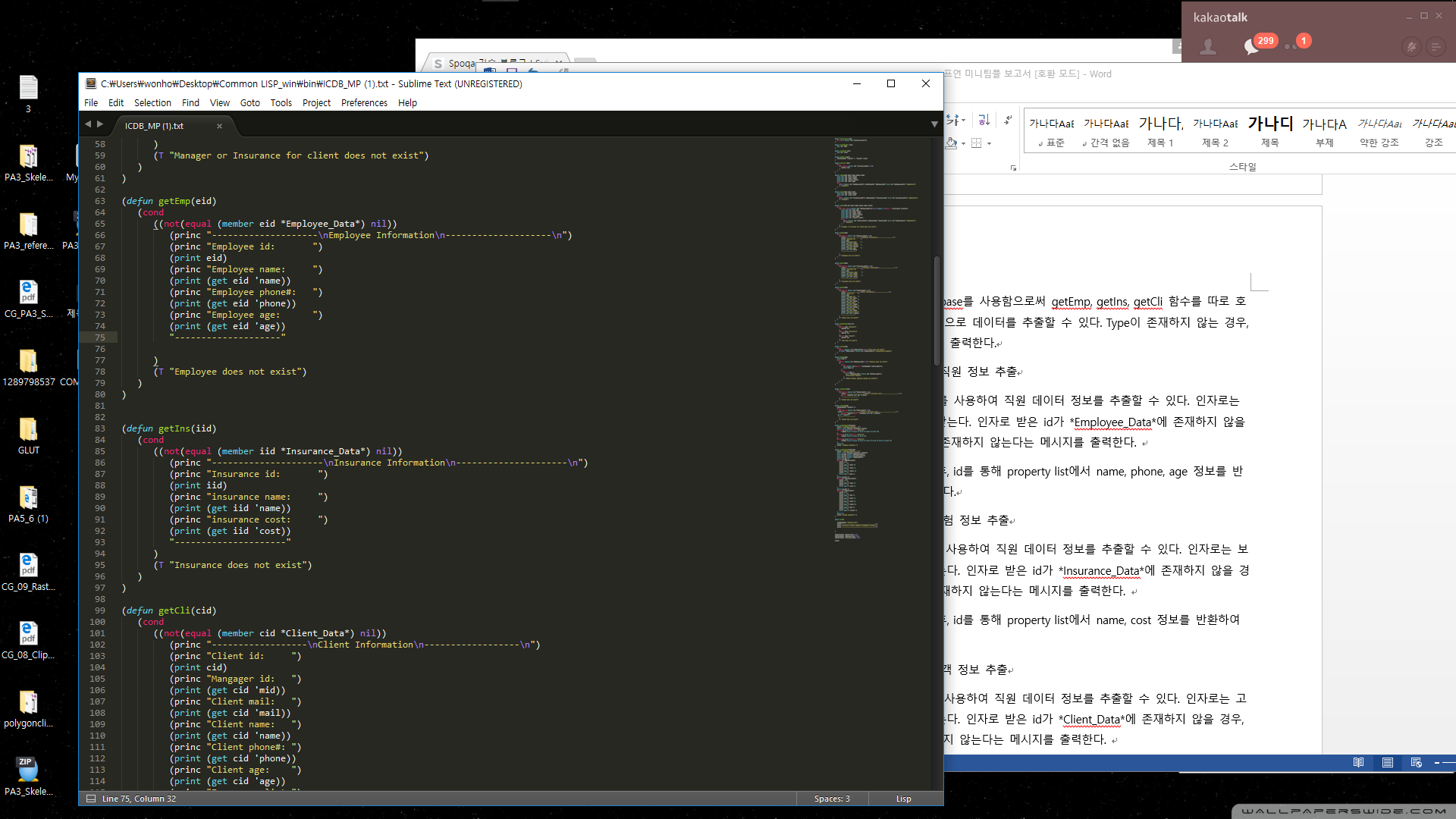
* + - 1. getEmp : 직원 정보 추출



함수 getEmp를 사용하여 직원 데이터 정보를 추출할 수 있다. 인자로는 직원의 id를 받는다. 인자로 받은 id가 \*Employee\_Data\*에 존재하지 않을 경우, 직원이 존재하지 않는다는 메시지를 출력한다.

Id를 출력한 후, id를 통해 property list에서 name, phone, age 정보를 반환하여 출력한다.

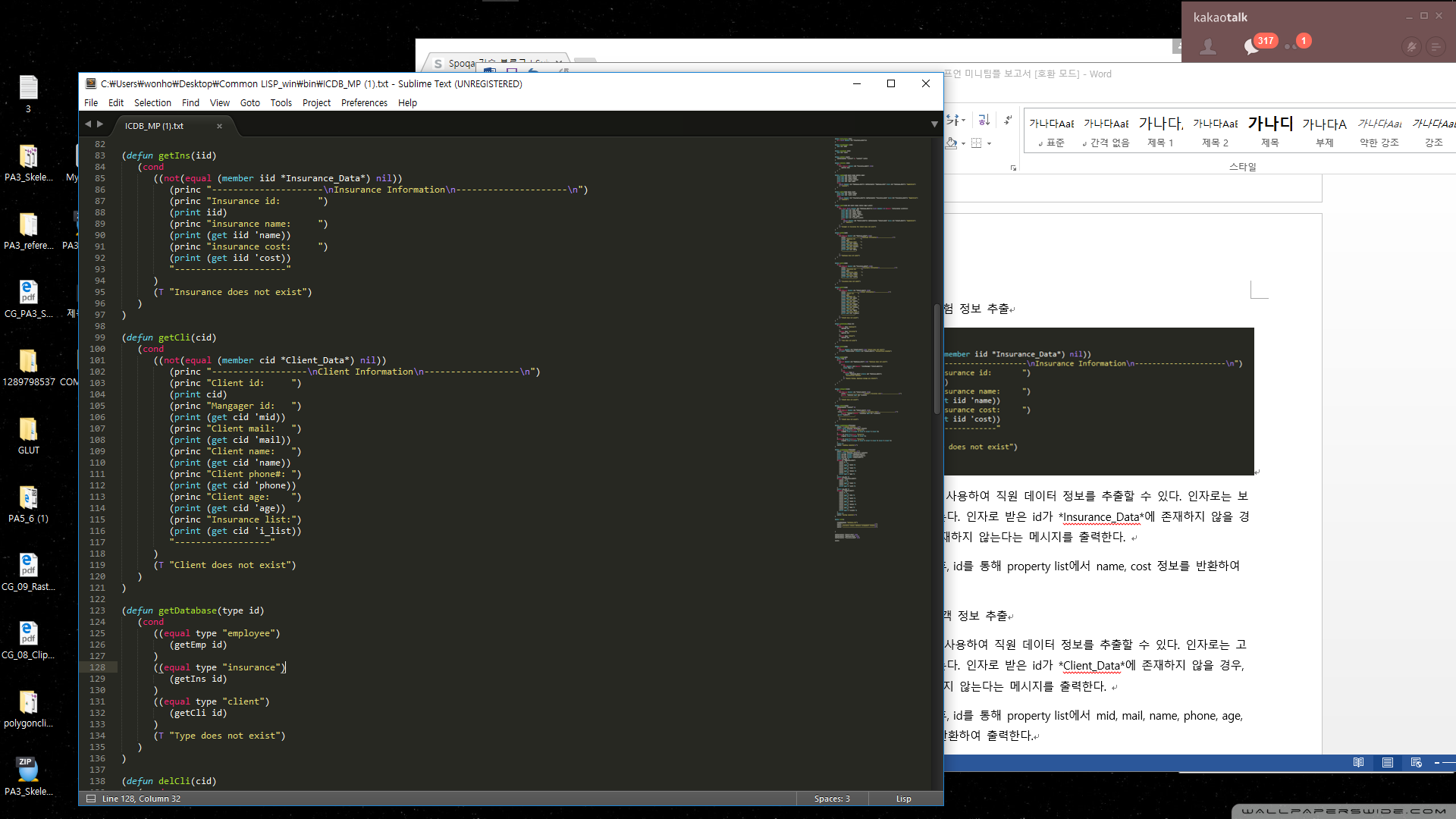
* + - 1. getIns : 보험 정보 추출



함수 getIns를 사용하여 직원 데이터 정보를 추출할 수 있다. 인자로는 보험의 id를 받는다. 인자로 받은 id가 \*Insurance\_Data\*에 존재하지 않을 경우, 보험이 존재하지 않는다는 메시지를 출력한다.

Id를 출력한 후, id를 통해 property list에서 name, cost 정보를 반환하여 출력한다.

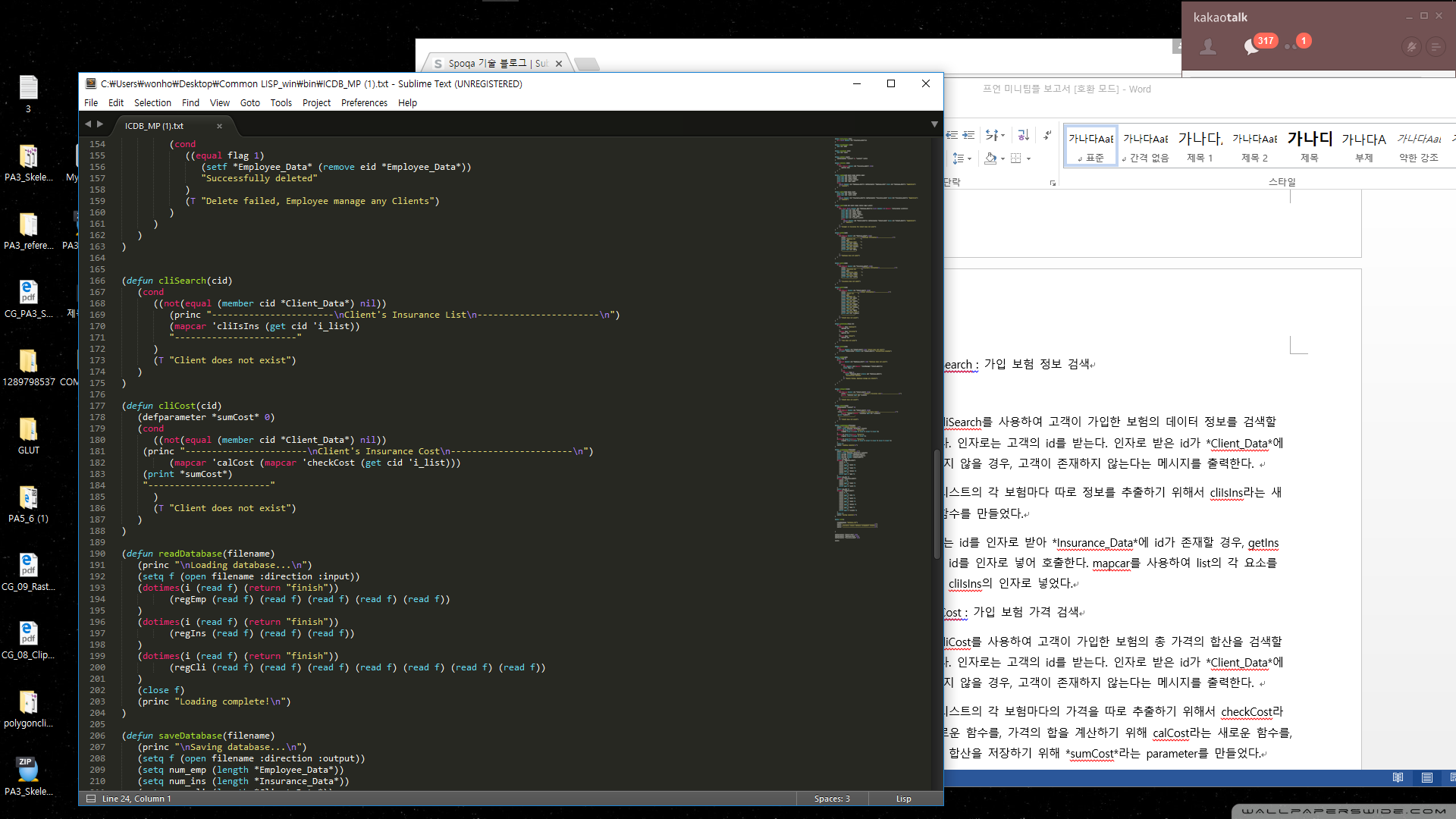
* + - 1. getCli : 고객 정보 추출



함수 getCli를 사용하여 직원 데이터 정보를 추출할 수 있다. 인자로는 고객의 id를 받는다. 인자로 받은 id가 \*Client\_Data\*에 존재하지 않을 경우, 고객이 존재하지 않는다는 메시지를 출력한다.

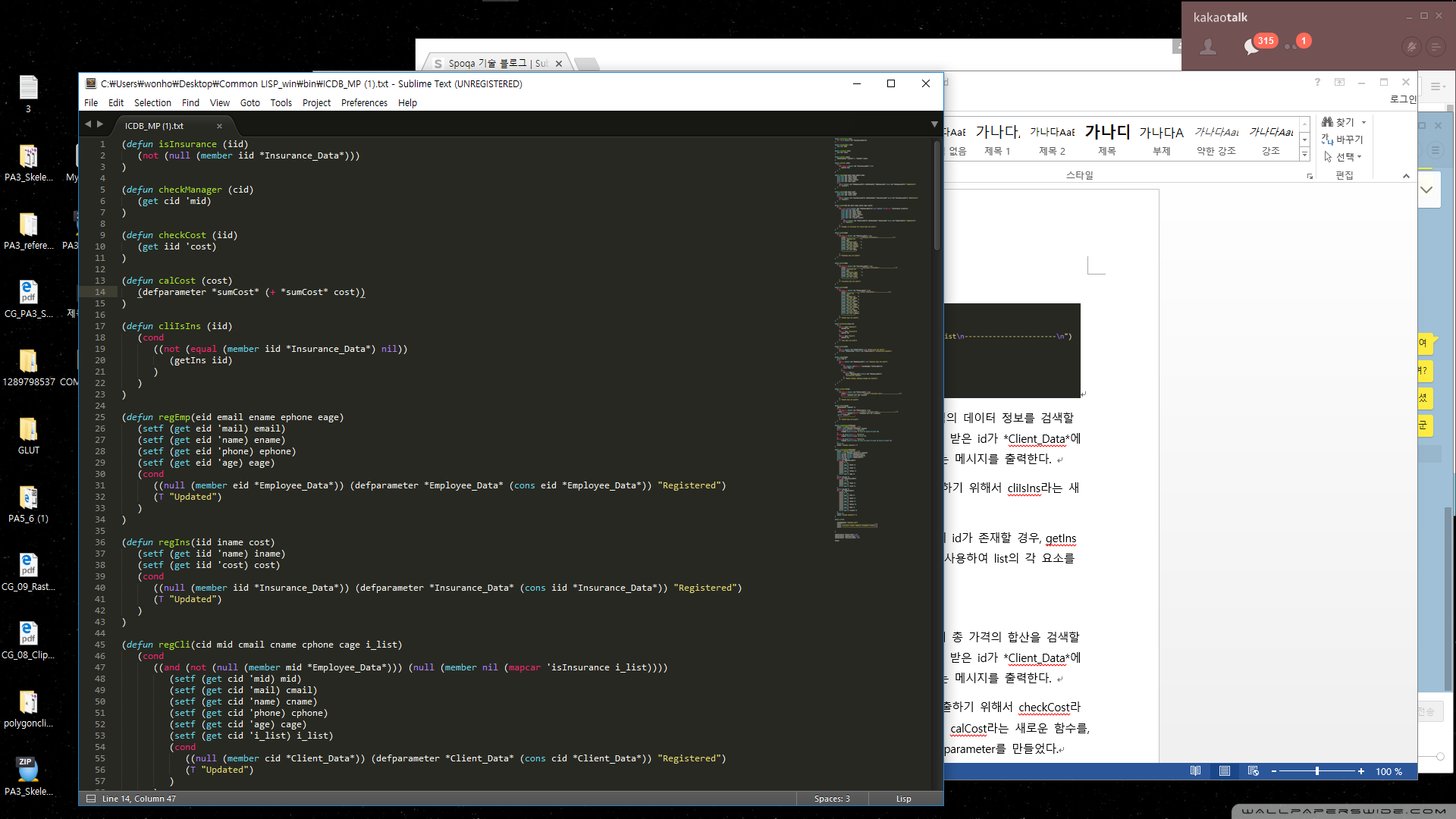
Id를 출력한 후, id를 통해 property list에서 mid, mail, name, phone, age, i\_list 정보를 반환하여 출력한다.

* + - 1. cliSearch : 가입 보험 정보 검색



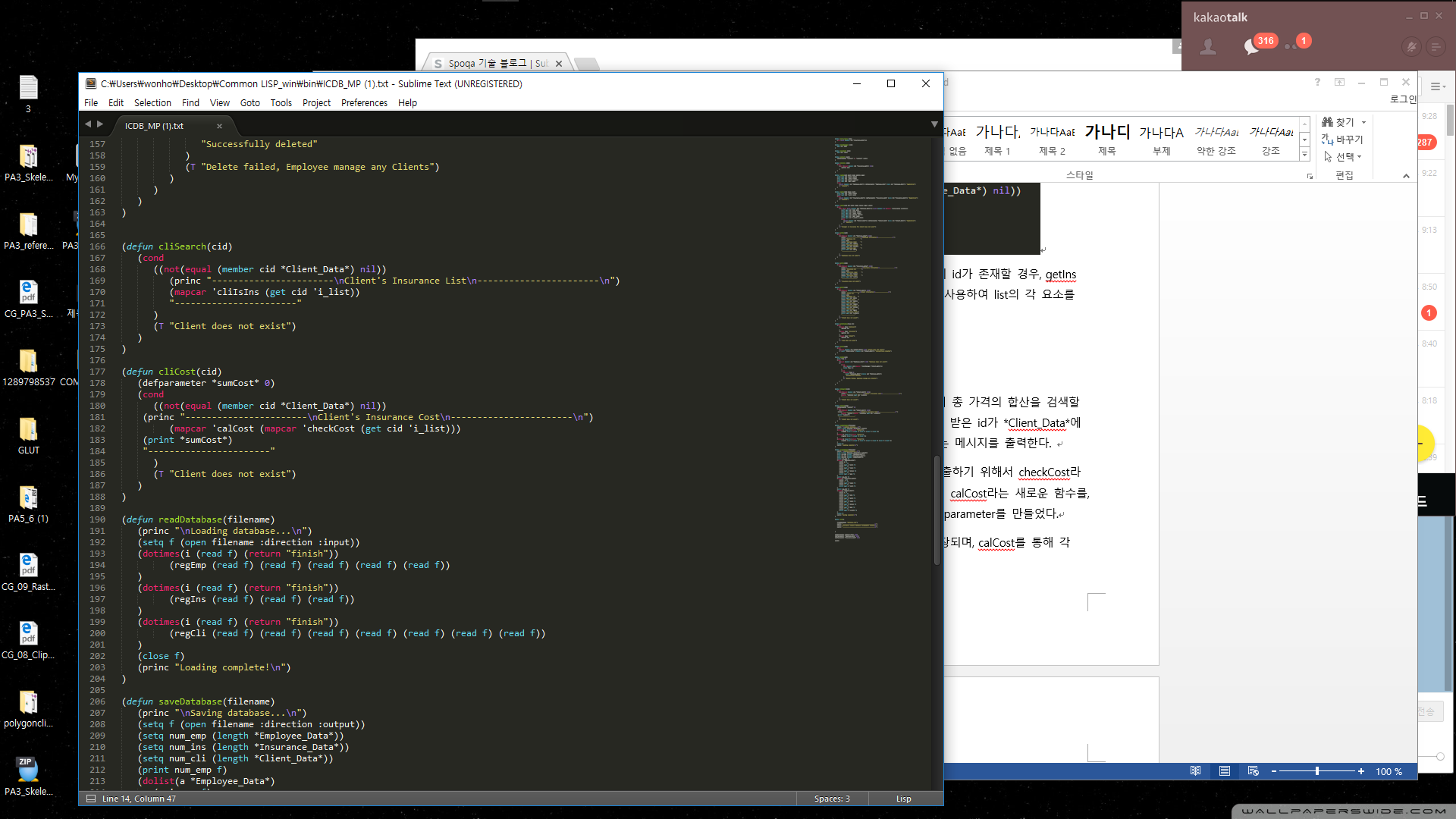
함수 cliSearch를 사용하여 고객이 가입한 보험의 데이터 정보를 검색할 수 있다. 인자로는 고객의 id를 받는다. 인자로 받은 id가 \*Client\_Data\*에 존재하지 않을 경우, 고객이 존재하지 않는다는 메시지를 출력한다.

보험 리스트의 각 보험마다 따로 정보를 추출하기 위해서 cliIsIns라는 새로운 함수를 만들었다.



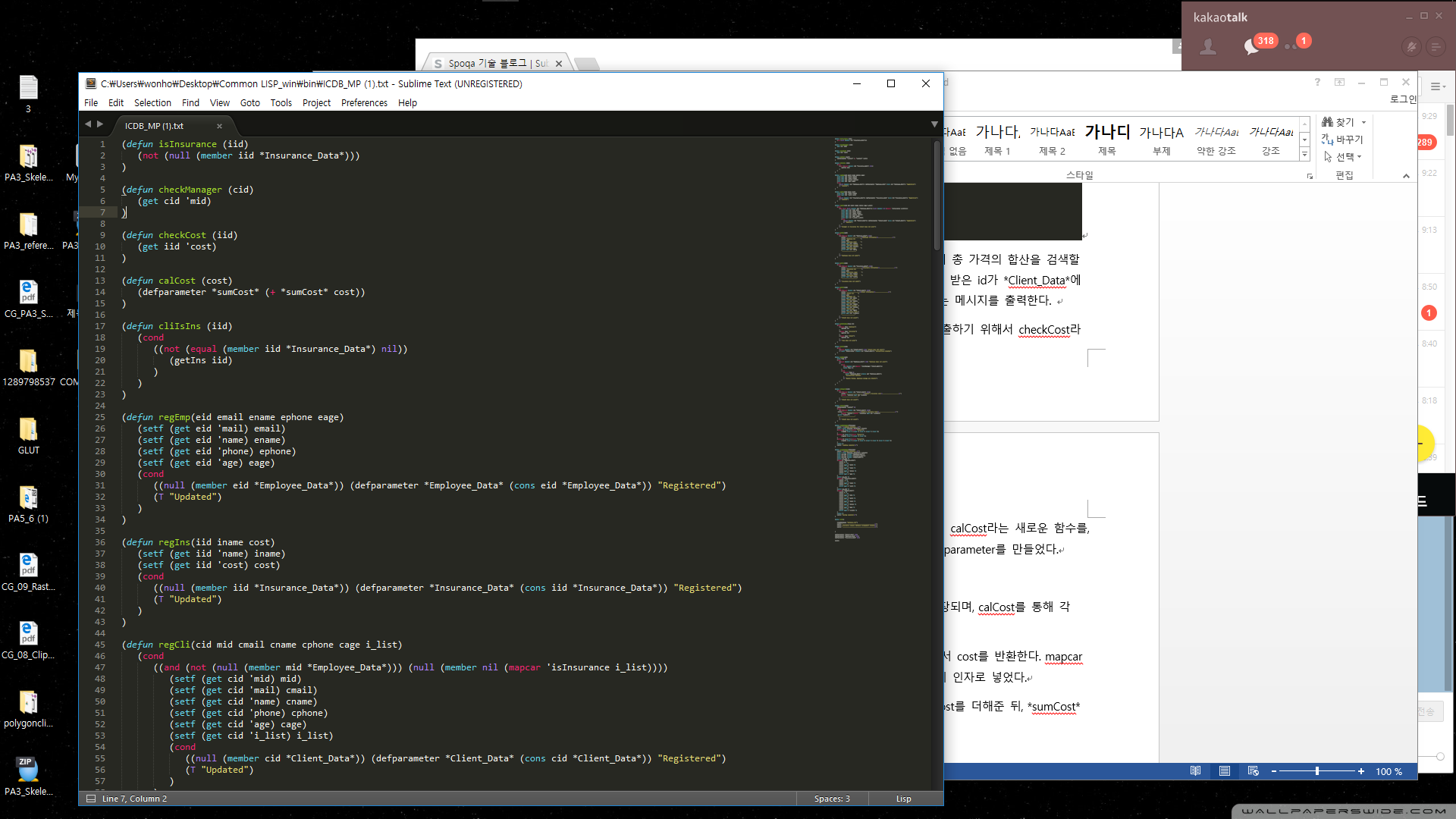
cliIsIns는 id를 인자로 받아 \*Insurance\_Data\*에 id가 존재할 경우, getIns 함수에 id를 인자로 넣어 호출한다. mapcar를 사용하여 list의 각 요소를 하나씩 cliIsIns의 인자로 넣었다.

* + - 1. cliCost : 가입 보험 가격 검색



함수 cliCost를 사용하여 고객이 가입한 보험의 총 가격의 합산을 검색할 수 있다. 인자로는 고객의 id를 받는다. 인자로 받은 id가 \*Client\_Data\*에 존재하지 않을 경우, 고객이 존재하지 않는다는 메시지를 출력한다.

보험 리스트의 각 보험마다의 가격을 따로 추출하기 위해서 checkCost라는 새로운 함수를, 가격의 합을 계산하기 위해 calCost라는 새로운 함수를, 가격의 합산을 저장하기 위해 \*sumCost\*라는 parameter를 만들었다.



\*sumCost\*는 함수가 호출되면 값이 0으로 저장되며, calCost를 통해 각 cost를 합산할 때에 쓰인다.

checkCost는 id를 인자로 받아 property list에서 cost를 반환한다. mapcar를 사용하여 list의 각 요소를 하나씩 calCost의 인자로 넣었다.

calCost는 cost를 인자로 받아 \*sumCost\*에 cost를 더해준 뒤, \*sumCost\* 값을 재정의해준다.

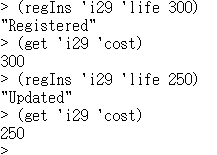
1. **실행 예시**
   1. regEmp

****

새로운 직원을 e41이라는 id로 regEmp함수를 통해 \*Employee\_Data\*에 넣으면 “Registered”라는 메시지가 떠서 등록이 되었다는 것을 알려주고, 실제 get을 통해 값을 가져오면 데이터가 잘 들어가 있다는 것을 알 수 있다.

또한 다시 같은 id인 e41로 age값만 바꿔서 regEmp함수를 통해 \*Employee\_Data\*에 넣으면 “Updated”라는 메시지가 떠서 이미 등록된 id이며, 데이터가 갱신되었다는 것을 알려준다. 또한 get을 통해 값을 가져오면 데이터가 갱신이 되었다는 것을 알 수 있다.

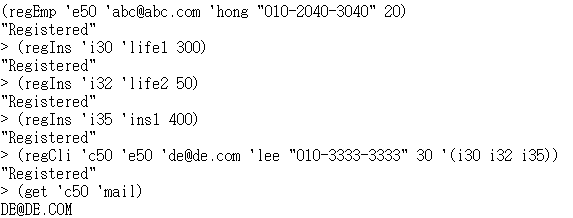
* 1. regIns

****

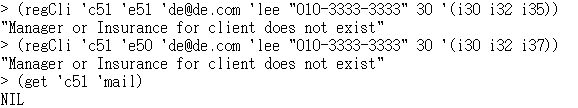
새로운 보험을 regIns를 통해 i29라는 id로 \*Insurance\_Data\*에 넣으면, “Registered”라는 메시지가 떠서 등록이 되었다는 것을 알려주고, 실제 get을 통해 그 보험의 cost값을 가져오면 데이터가 잘 들어가 있다는 것을 알 수 있다.

또한 다시 같은 id인 i29로 cost값을 바꿔서 regIns함수를 통해 \*Insurance\_Data\*에 넣으면 “Updated”라는 메시지가 떠서 이미 등록된 id이며, 데이터가 갱신되었다는 것을 알려준다. 또한 get을 통해 값을 가져오면 데이터가 갱신이 되었다는 것을 알 수 있다.

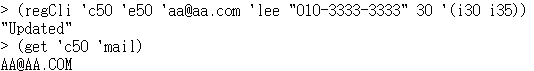
* 1. regCli

****

새로운 고객을 regCli를 통해 e50이라는 id로 \*Client\_Data\*에 넣으면 (그 전에 그 고객을 담당할 직원과 그 고객이 가입한 보험 목록이 미리 regEmp와 regIns를 통해 등록이 되었다) Registered 메시지가 떠서 등록이 되었다는 것을 알려주고, 실제 get을 통해 그 고객의 mail값을 가져오면 데이터가 잘 들어가 있다는 것을 알 수 있다.



만약 존재하지 않는 i37이라는 보험을 가입한 고객을 등록하려 하거나, e51처럼 존재하지 않는 직원이 담당하는 고객을 등록하려 하면 Manager or Insurance for client does not exist라는 메시지를 띄워서 등록하지 못했다는 것을 알려준다. 이 경우 get을 통해 mail값을 가져오려고 해도 nil값이 떠서 데이터가 없다는 것을 알 수 있다.



또한 이미 존재하는 id인 c50으로 고객을 등록하면, Registered 대신 Updated라는 메시지가 떠서 이미 등록된 id이며, 데이터가 갱신되었다는 것을 알려준다. 실제로 mail값을 확인해보면 값이 바뀌었다는 것을 알 수 있다.

* 1. delEmp

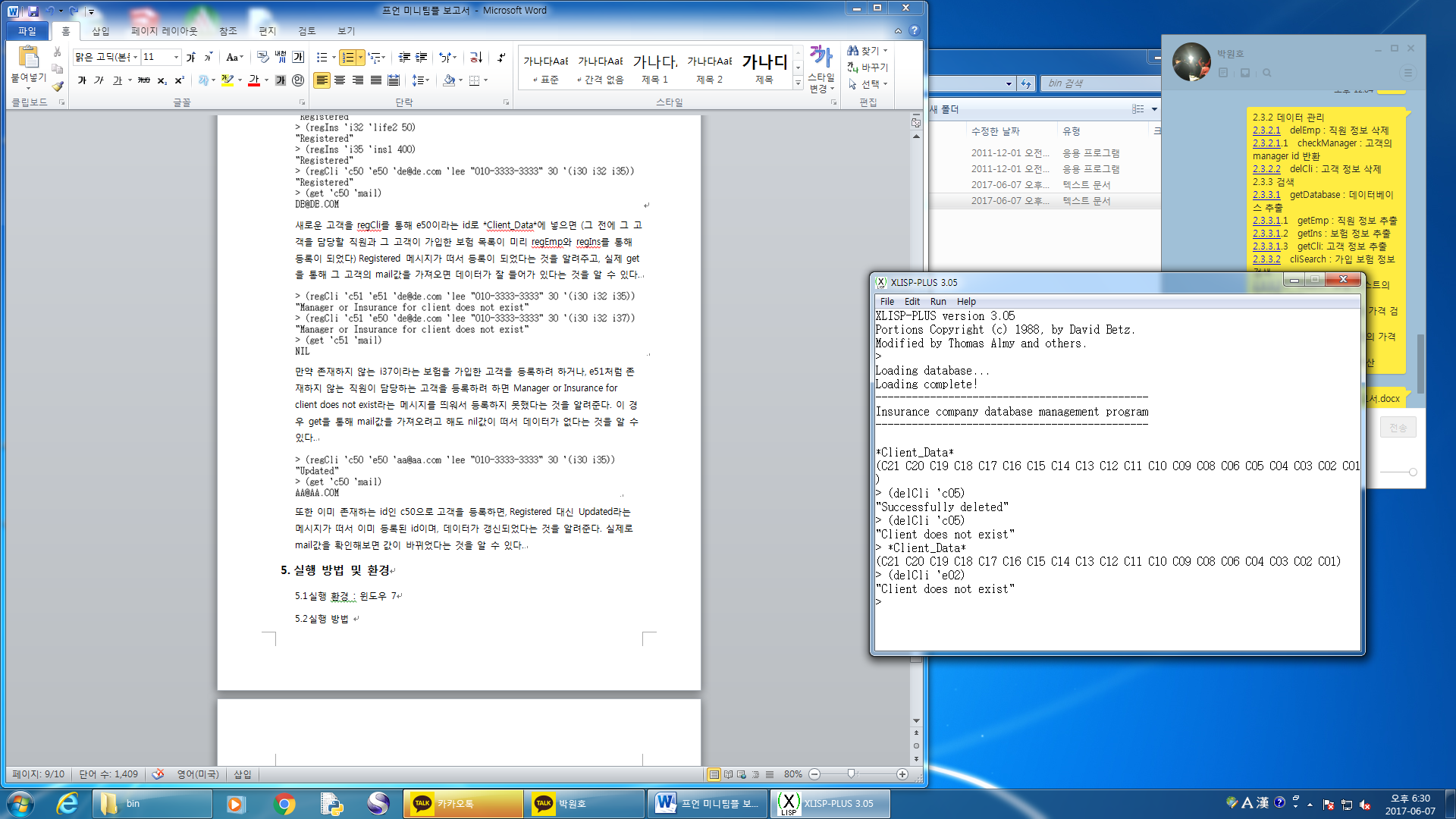


삭제할 직원을 e02 id로 delEmp함수에 인자로 넣으면 “Delete failed, Employee manage any Clients”라는 메시지가 떠서 삭제에 실패했다는 것을 알려주고, 실제 (mapcar ‘checkManager \*Client\_Data\*)를 통해 값을 살펴보면 직원이 관리하는 고객이 존재함을 알 수 있다.

삭제할 직원을 e01 id로 delEmp함수에 인자로 넣으면 “Successfully deleted”라는 메시지가 떠서 삭제가 되었다는 것을 알려주고, 실제 \*Employee\_Data\*를 통해 값을 살펴보면 데이터가 삭제되었다는 것을 알 수 있다.

또한 다시 같은 id인 e01를 id로 delEmp함수에 인자로 넣으면 “Employee does not exist”라는 메시지가 떠서 이미 삭제되었거나 없는 id라는 것을 알려준다.

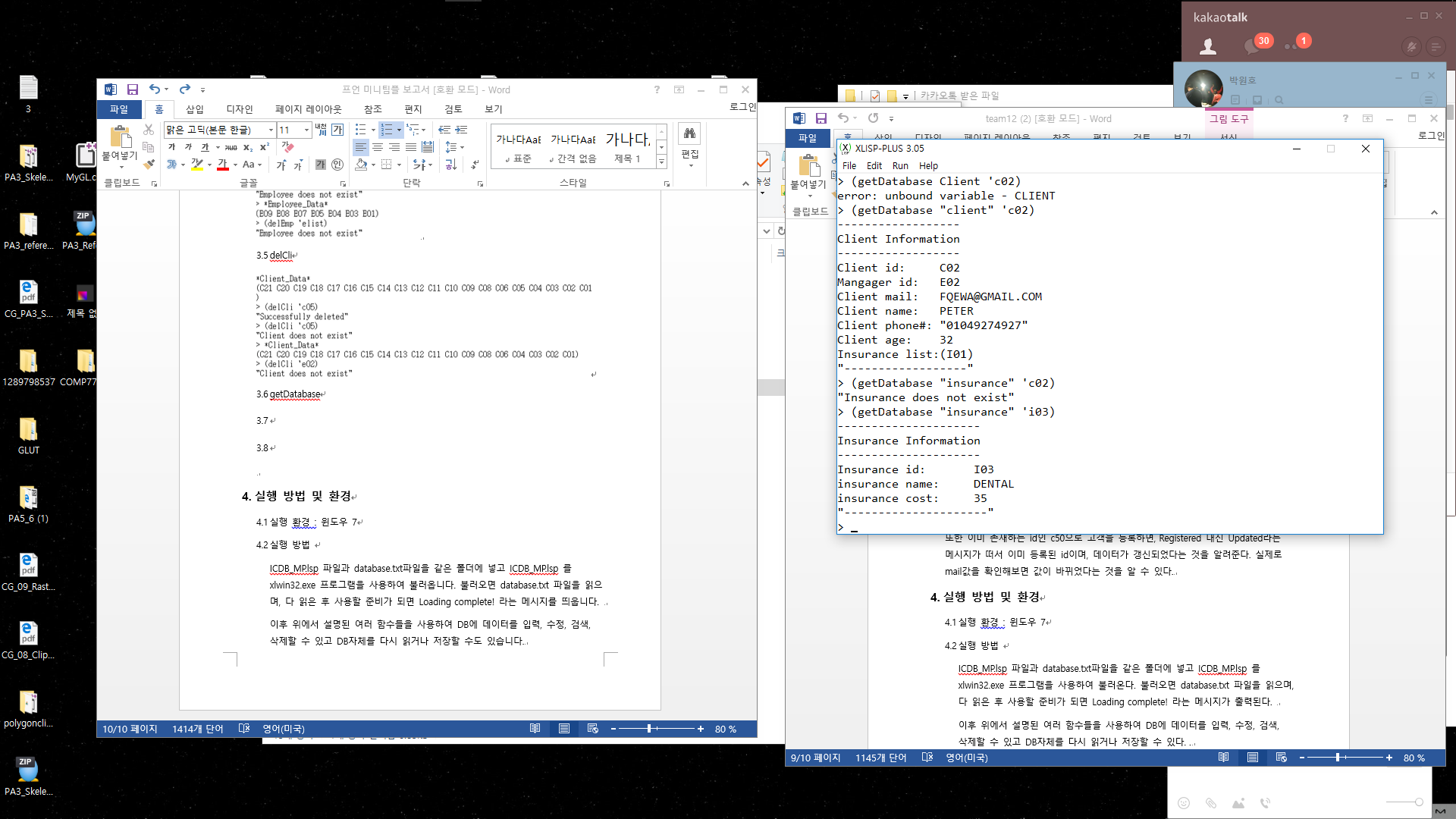
* 1. delCli

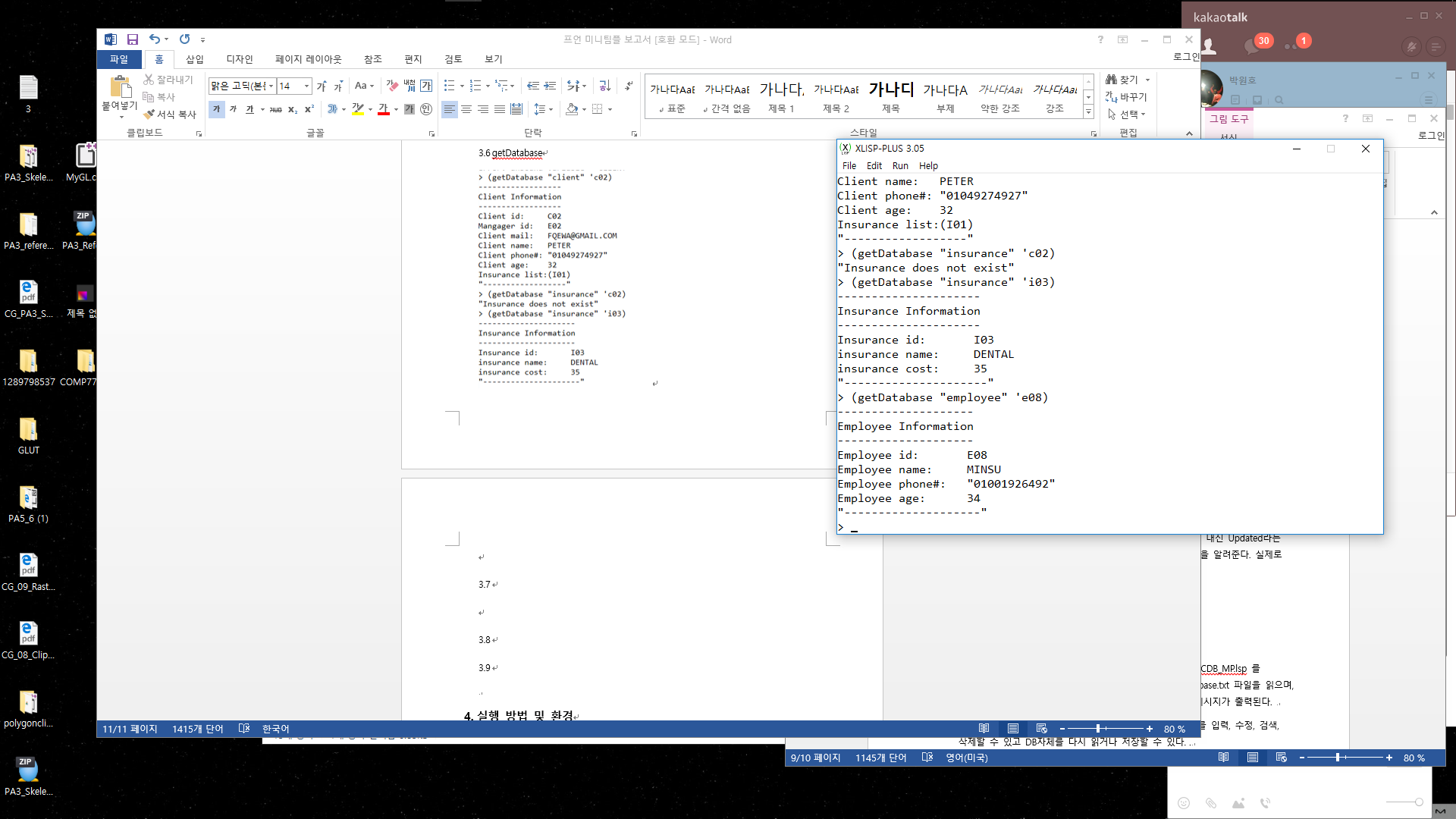


삭제할 고객을 c05라는 id로 delCli함수에 인자로 넣으면 “Successfully deleted”라는 메시지가 떠서 삭제가 되었다는 것을 알려주고, 실제 \*Client\_Data\*를 통해 값을 살펴보면 데이터가 삭제되었다는 것을 알 수 있다.

또한 직원 id인 e02로 delCli함수에 인자로 넣으면 “Client does not exist”라는 메시지가 떠서 이미 삭제되었거나 없는 id라는 것을 알려준다.

* 1. getDatabase





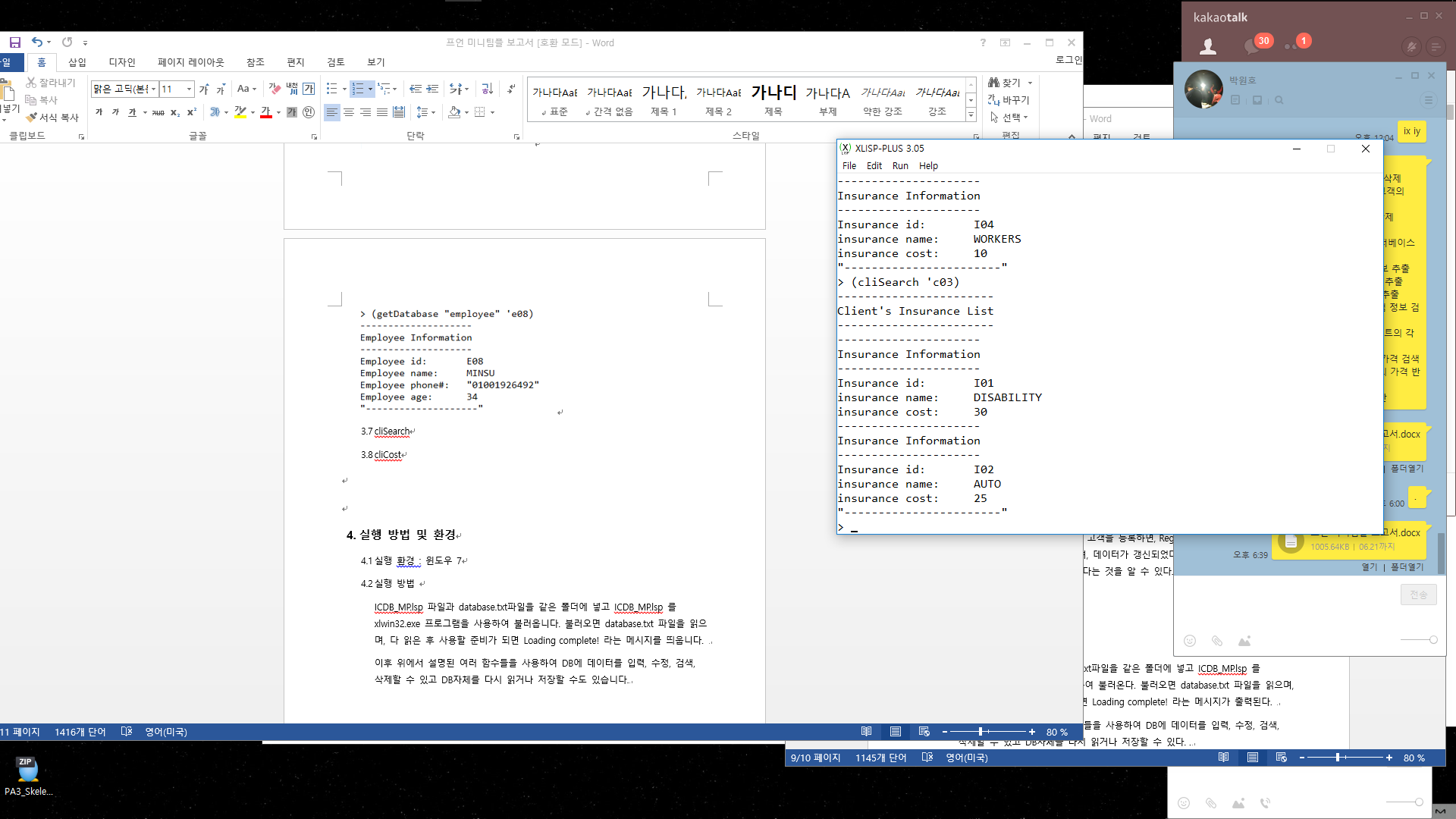
검색할 정보를 “client”라는 type과 c02라는 id로 getDatabase함수에 인자로 넣으면 고객의 모든 정보가 각 줄마다 출력된다.

또한 검색할 정보를 “insurance”라는 type과 i03이라는 id로 getDatabase함수에 인자로 넣으면 보험의 모든 정보가 각 줄마다 출력된다.

마찬가지로 검색할 정보를 “employee”라는 type과 e08이라는 id로 getDatabase함수에 인자로 넣으면 직원의 모든 정보가 각 줄마다 출력된다.

getDatabase함수의 인자에 다른 값을 넣거나 만족하는 정보가 없을 경우, “(type) does not exist”라는 오류 메시지를 출력한다.

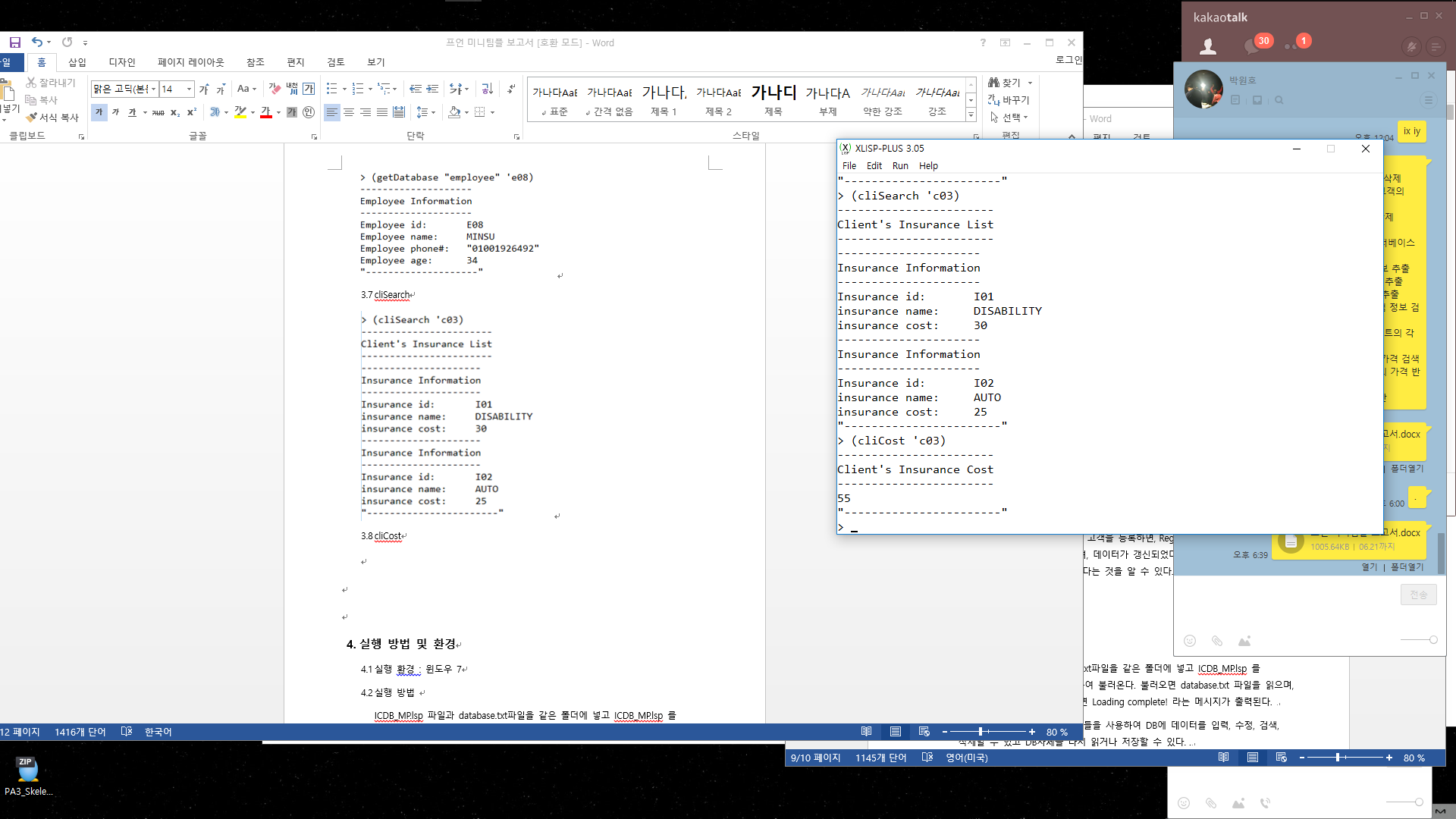
* 1. cliSearch



검색할 고객을 c03이라는 id로 cliSearch함수에 인자로 넣으면 고객이 가입한 보험 리스트를 각 보험마다 getDatabase의 형식대로 출력된다.

cliSearch 함수의 인자에 다른 값을 넣거나 만족하는 정보가 없을 경우, “Client does not exist”라는 오류 메시지를 출력한다

* 1. cliCost



검색할 고객을 c03이라는 id로 cliCost함수에 인자로 넣으면 고객이 가입한 보험 리스트의 각 보험 가격의 총합을 출력한다.

cliCost 함수의 인자에 다른 값을 넣거나 만족하는 정보가 없을 경우, “Client does not exist”라는 오류 메시지를 출력한다.

1. **실행 방법 및 환경**
   1. 실행 환경 : 윈도우 7
   2. 실행 방법

ICDB\_MP.lsp 파일과 database.txt파일을 같은 폴더에 넣고 ICDB\_MP.lsp 를 xlwin32.exe 프로그램을 사용하여 불러온다. 불러오면 database.txt 파일을 읽으며, 다 읽은 후 사용할 준비가 되면 Loading complete! 라는 메시지가 출력된다.

이후 위에서 설명된 여러 함수들을 사용하여 DB에 데이터를 입력, 수정, 검색, 삭제할 수 있고 DB자체를 다시 읽거나 저장할 수 있다. .