**Implementation Environment**

**Hardware**

Cpu : i5-8265U

**OS** : Window 10

**C++**

IDE : VS CODE

Compiler version : 6.3.0

Other library : #include <ctime> 추가

**JAVA**

IDE : IntelliJ IDEA

Java version : 1.8.0\_201

Other libaray : import java.util.Calendar 추가.

**Specific explanation about the code**

C++ 기준으로 코드를 구현하고, 자바로 코드 구현을 옮겼기 때문에

C++ 기준으로 설명한다.

(자바로 코드 구현을 옮길 때 추가된 구현은 troubleshooting 항목에 적어놓았다.)

코드의 작동 과정은 크게 세 부분으로 나눌 수 있다.

1. 각각 전화번호부에 객체를 추가하는 입력 부분
2. 전화번호부에서 객체를 삭제하는 삭제 부분
3. 전화번호부에 있는 객체들을 보여주는 출력 부분

처음에 유저에게 입력을 받아서, 프로그램을 종료할지, 혹은 위 세가지 모드 중

어떤 모드로 들어갈지를 정한다.

1. 입력 모드

1을 입력 받았을 경우 입력 모드로 들어간다.

입력 모드에서는 Person, Work, Family, Friend 객체를 입력 받을 수 있다.

Person의 경우 Firstname, Lastname, PhoneNumber 원소가 필요하고,

Work의 경우 Person을 구성하는데 필요한 원소에 Team이 추가로,

Family의 경우 Person을 구성하는데 필요한 원소에 Birthday이 추가로,

Friend의 경우 Person을 구성하는데 필요한 원소에 Age가 추가로 필요하다.

이러한 입력을 받을 때, 제대로 된 입력만 받을 수 있게 regex를 이용하여 입력을 걸러주었다.

기본적으로 빈칸을 포함해서, 첫글자~마지막 글자까지 형식에 맞지 않는 부분이 있으면 입력을 거부한다.

Name의 경우, “<FirstName> <LastName>”의 꼴이 아니면 입력을 거부하게 하였다.

PhoneNumber의 경우, (02 | 010) – xxxx –xxxx 의 꼴이 아니면 입력을 거부하게 하였다.

BirthDay의 경우 월과 일을 확인하여, 유효한 월일이 아니면 입력을 거부하게 하였다.

예를 들어, 960229 (윤년은 포함 안함) , 131301 (13월은 없음), 950431 (4월에 31일은 없음)

과 같은 입력을 거부한다. (check\_Month\_Day() 함수/메소드를 통해 구현됨)

Age의 경우 자연수 입력만 받게 하였다.

Team의 경우 모든 string 입력을 받는다.

이러한 입력을 받을 때, while문을 사용하여 들어온 입력이 조건에 맞는지를 체크하고,

프로그램이 잘못된 입력형태라고 판단하면 다시 입력 받기를 시도한다.

이렇게 입력받는 과정은 C++의 경우 get\_xxx\_from\_User() 함수를 통해,

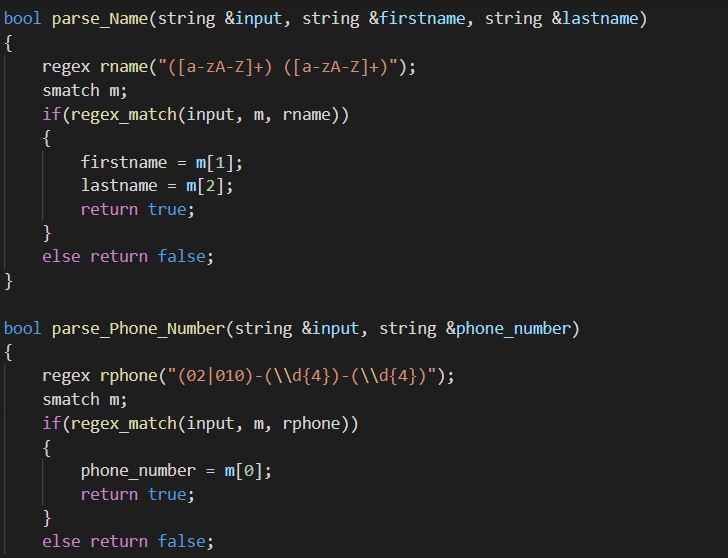
JAVA의 경우 helper class의 get\_xxx\_from\_User() 메소드를 통해 구현하였다.

각 get\_xxx\_from\_User() 함수/메소드는 내부에 while문에서 parse\_xxx 함수/메소드를 호출하여

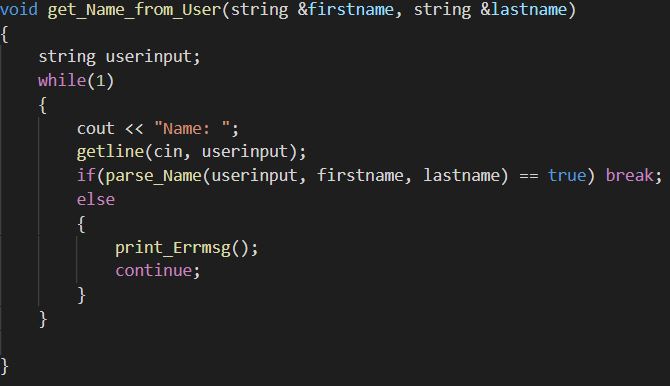
사용자 입력이 제대로 들어왔는지를 판단한다.

만약 입력이 제대로 들어왔다면 parse\_xxx는 사용자 입력을 클래스에 들어갈 수 있는 형태로 바꿔주고 true를 리턴하고, 입력이 제대로 들어오지 않았다면 데이터에 특별한 조작을 하지 않고

False를 리턴하게 하였다.



<Regex와 parse\_name 예시>



<Get\_name\_from\_user 예시>

모든 입력을 마치면 입력에 맞는 새로운 객체를 만들어서 phonebook에 추가하고,

사용자에게 객체가 추가되었다는 메시지를 출력한다.

2. 삭제 모드

2를 입력 받았을 경우 삭제 모드로 들어간다.

사용자로부터 인덱스를 받아, 그 인덱스에 해당하는 객체를 phonebook에서 제거하고

객체가 성공적으로 삭제되었음을 알려준다.

인덱스의 형태는 자연수만 정상적으로 입력 받으며,

그렇지 않은 경우 다시 입력을 요청 받는다.

이 과정은 입력모드 에서와 같이 get\_index\_from\_user 와 parse\_index 함수를 이용해 구현하였다.

만약 현재 있는 객체의 인덱스가 아닌 인덱스를 입력 받았을 경우

오류메시지를 출력하고 initial state로 돌아간다.

객체를 삭제할 때는, C++의 경우 new를 이용하여 동적으로 객체를 할당 받았기 때문에

Delete를 이용하여 객체에 할당된 힙 메모리를 해제해주었고,

JAVA의 경우 Garbage Collector가 알아서 자원을 회수해 주기 때문에 객체를 제거한 이후

딱히 다른 과정을 거치지는 않았다.

3. 출력 모드

3을 입력 받았을 경우 출력 모드로 들어간다.

출력모드에서는 phonebook에 있는 모든 객체의 정보를 출력해서 유저에게 보여준다.

HW1에서 모든 객체는 Person 혹은 Person의 자식 클래스의 인스턴스 이므로,

polymorphism을 이용해 각 객체의 print함수를 이용해 전체 Phonebook list를 순회하면서

각 객체의 정보를 출력하게 하였다.

**Troubleshooting points while implementing your code**

처음 구현할 때는 C++로 구현을 하였고,

C++의 로직을 그대로 이용하여 자바에서 다시 코드를 짰다.

**C++ :**

제공받은 HW 스펙에서 <Person>형의 벡터에 Instance들을 담으라는 부분이 있었는데,

Person형이 아닌, Person형의 파생클래스인 Friend, Family, Work등의 클래스는

Person형의 벡터에 저장하게 되면 파생클래스가 추가로 가지고있는 birthday와 같은 정보들이 모두 손실되는 문제가 있어서,

Person이 아닌 Person \*형의 벡터를 사용하고 동적 메모리 할당을 이용하여 구현하였다.

그리고 사용자의 입력을 걸러내는 것 때문에 regex를 사용하였다.

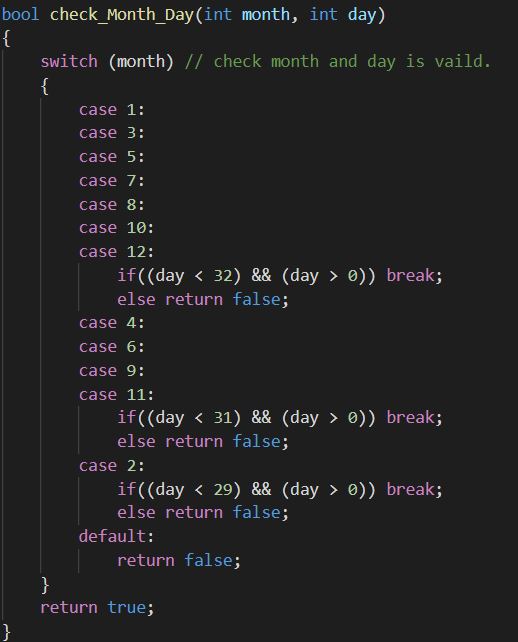
가장 신경을 많이 쓴 부분은 날짜를 처리하는 부분이었는데,

사용자가 입력한 연, 월, 일중에 말이 안되는 월과 일을 골라내야 하는 부분이 있었다.

예를 들어, 950229 (윤년이 없으면 2월 29일은 없음), 091301 (13월은 없음) 등의 입력을 걸러내야했다.

이를 해결하기 위해 사용할 수 있는 유용한 라이브러리가 있나 찾아봤지만, 딱히 쓸만한 라이브러리가 보이지 않아 그냥 각 월마다 해당하는 날짜를 고려하여 직접 구현하였다.

(1월 : 31일, 2월 : 28일, 3월 : 31일 …..)



<날짜 입력 거르기>

또한 Family 클래스의 dDay() 메소드를 구현할 때도 날짜를 세야 하는 번거로운 과정이 들어가 있었으므로 마찬가지로 직접 월마다 날짜수를 구해서 구현하였다.

**JAVA :**

C++의 로직을 그대로 사용하려고 하니, 내가 쓴 로직은 reference type을 이용하여

메소드 내부에서 argument로 전달받은 인자의 값을 바꿔주는 함수들이 있었는데,

JAVA의 경우 메소드 내부에서 인자의 값을 바꿀 수 없는 경우가 있었다.

예를 들어, String 클래스의 경우에는 immutable object라서 값을 바꿀 수가 없었고,

Integer의 경우 call-by-reference가 아니라 call-by-value형태로 값이 전달되다 보니

메소드 내부에서 integer의 값을 바꿔주어도 메소드 외부에서는 값이 바뀌지 않았다.

따라서 C++의 로직을 그대로 사용하기 위해,

String 클래스의 경우, mutable한 StringBuffer 클래스를 인자로 넘겨주어 메소드 내부에서

객체 값을 바꿀 수 있게 한 다음,

Person등의 객체를 만들 때 StringBuffer를 다시 String으로 형 변환 하였고,

Integer의 경우 대신 내가 직접 만든 mutable한 IntHolder라는 클래스를 이용해

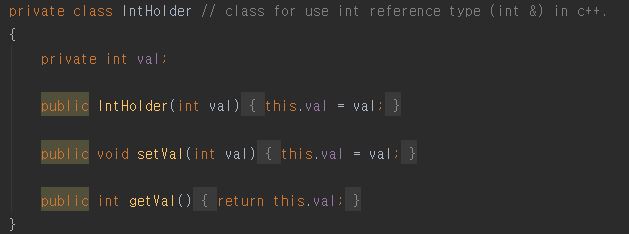
메소드 내부에서 이 객체의 값을 바꾸게 하여 기능을 구현하고,

외부로는 IntHolder에 저장된 int값을 정상적으로 리턴 하게 하여 사용자에게 구현을 숨겼다.

IntHolder같은 경우는 Public하게 쓰이면 문제가 있을 수 있기 때문에

자잘한 일을 도와주는 Helper클래스를 만들고, 그 클래스 내부에서 private하게 선언하여

그 클래스 내부에서만 사용하는 형태로 Encapsulation 하여 다른 부분에서는 사용하지 못하게 하였다.



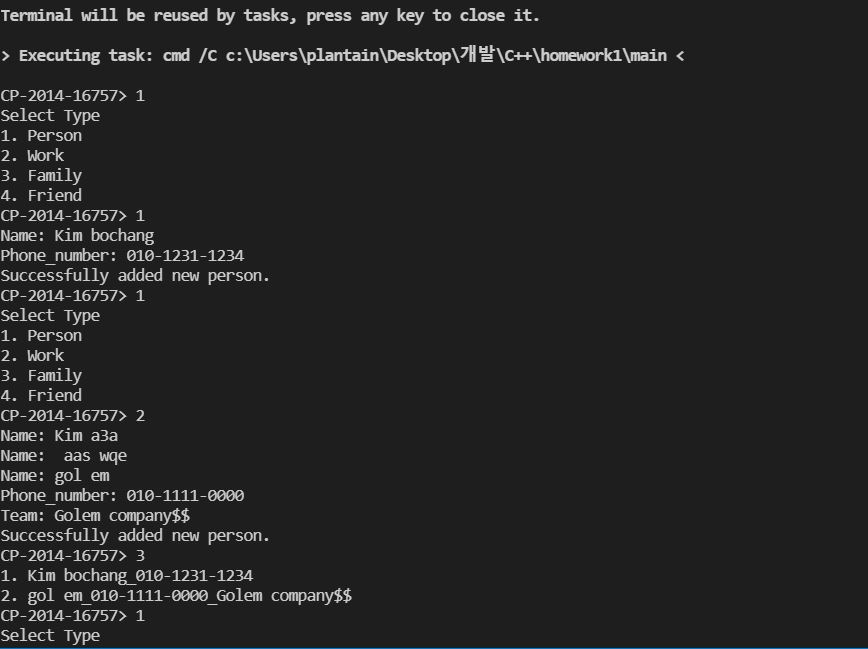
<IntHolder 클래스>

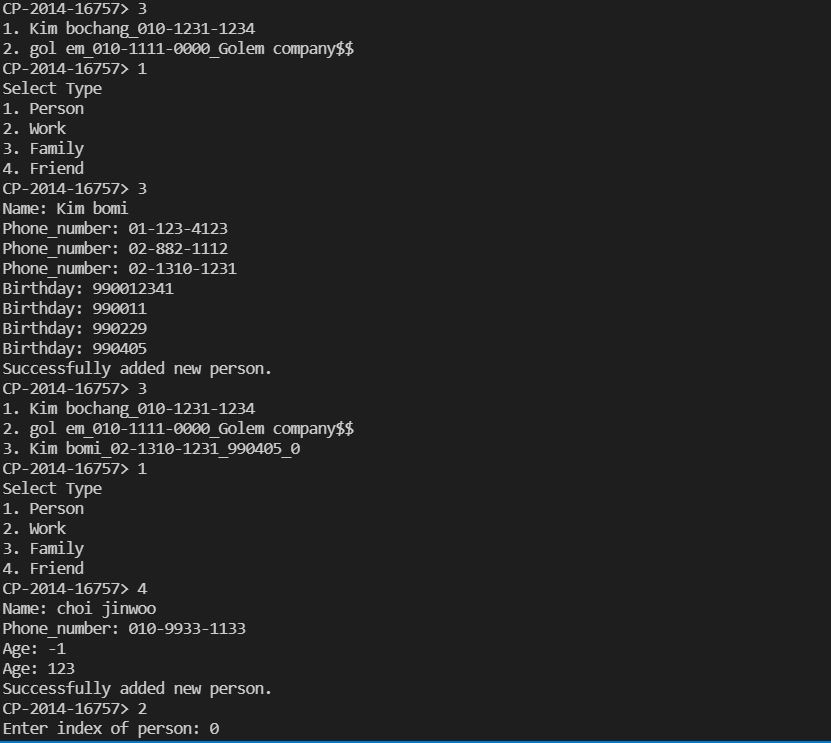
또한 자바의 경우는 날짜를 담기 위해 Calendar라는 클래스를 제공하므로,

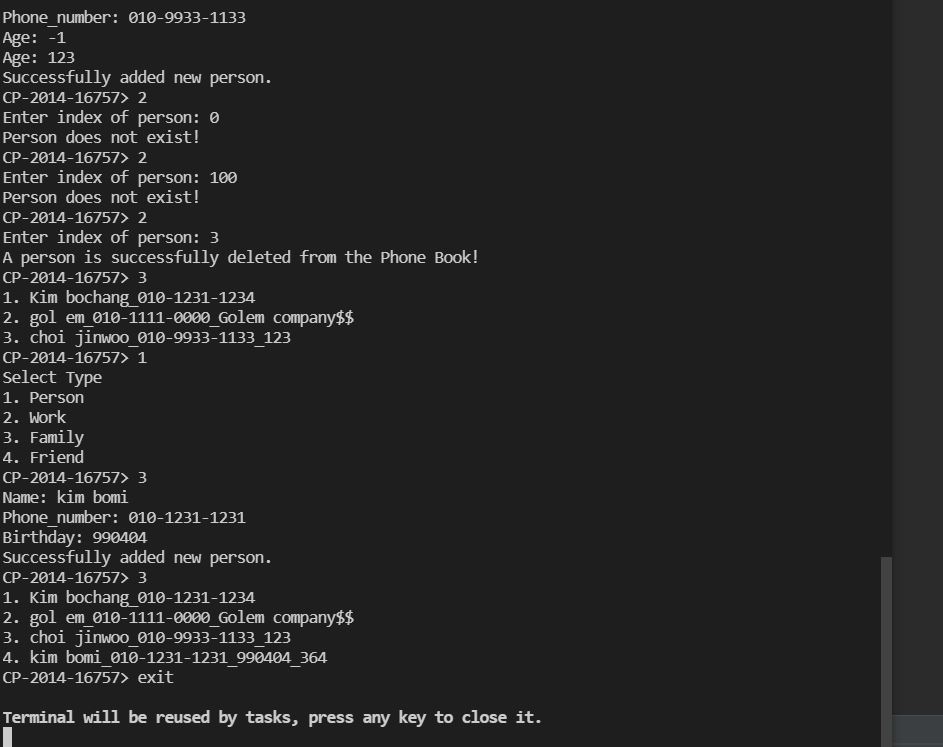
dDay 메소드 구현을 위해, 오늘 날짜를 Calendar 클래스를 통해 가져오게 하였다.

**Screenshot of the program working**

C++







JAVA

