Week 7-2 : Virtual Function & Polymorphism

Part 2. Polymorphism
Same sentence but different result.
Polymorphism : the method of implementing all of the super-class' member. Sub-class has its own member and super class' member.
Is-a relation.

- Polymorphism

```cpp
class Person
{
private:
    int age;
    char name[50];
public:
    Person(int myage, char * myname) : age(myage)
    {
        strcpy(name, myname);
    }
    void ShowName() const
    {
        cout << "My name is" << name << endl;
    }
    void ShowAge() const
    {
        cout << "My age is" << age << endl;
    }
};

class Student : public Person
```

Student is a person. (is-a relation), student class inherits person class. Student is implemented by 'public Person'. And Student is inherited from Person's member.

- Virtual Function

```cpp
#include <iostream>
using namespace std;

class First
{
public:
    virtual void MyFunc() { cout<<"FirstFunc"<<endl; }
};

class Second: public First
{
public:
    virtual void MyFunc() { cout<<"SecondFunc"<<endl; }
};

class Third: public Second
{
```
Result is different. because ptr's referencing object is different.

- Virtual Destructor

```cpp
class First
{
    ....
public:
    virtual void ~First() { .... }
};

class Second: public First
{
    ....
public:
    virtual void ~Second() { .... }
};

class Third: public Second
{
    ....
public:
    virtual void ~Third() { .... }
};
```
```cpp
#include <iostream>
using namespace std;

class Account {
    protected:
        char acc_num[50];  // Account number
        int balance;       // balance

    public:
```

~Third() is called, because it is virtual destructor. After this
~Second(), ~First() are called sequently.

**[Exercise]**
The following class is Account class that is defined for the bank
account information.
MyAccount class that inherits Account class publicly should have a
member variable
that contains customer’s transit limitation information(trans_limit)
besides other
Account class member variables.
  - Write the blank in MyAccount.cpp
  - Define constructor function and ShowData function in
MyAccount class
  - Call the constructor of Base class through member
initializer
  - Define class by referencing main function and execution
results
Account(char* num, int bal) {
    strcpy(acc_num, num);
    balance = bal;
}

class MyAccount : public Account {

    /* Blank */

};

int main(void) {
    MyAccount acc("302-1234-5678-07", 5000, 10000);
    acc.ShowData();

    //getchar();
    return 0;
};

Result:

Account Number: 302-1234-5678-07
Account Balance: 5000 Won
Account Limit: 10000 Won