352-354p

순수 가상함수와 추상 클래스

```
class Employee
{
private:
    char name[100];
public:
    Employee(char * name) { . . . . }
    void ShowYourName() const { . . . . }
    virtual int GetPay() const
    {
        return 0;
    }
    virtual void ShowSalaryInfo() const
    { }
};
```

순수 가상함수: 몸체가 정의되지 않은 함수
추상 클래스: 하나 이상의 순수 가상함수를 멤버로 가진 클래스. 객체생성이 불가능한 클래스.

오버라이딩을 목적으로 정의된 함수들!
따라서 몸체부분의 정의는 의미가 없다!

```
virtual int GetPay() const = 0;
virtual void ShowSalaryInfo() const = 0;
```

순수 가상함수로 대체 가능!

356-357p

다형성(Polymorphism)

```
class First
{
public:
    virtual void SimpleFunc() { cout<<"First"<<endl; }
};

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

int main(void)
{
    First * ptr=new First();
    ptr->SimpleFunc();    // 아래에 동일한 문장이 존재한다.
    delete ptr;
    ptr=new Second();
    ptr->SimpleFunc();    // 위에 동일한 문장이 존재한다.
    delete ptr;
    return 0;
}
```

Polymorphism :

문장은 같은데 결과는 다르다!

virtual 함수의 영향.

357-360p

Virtual Destructor
(VirtualDestructor.cpp 1/2)

◆ **소멸자를 가상으로 선언**

- **유도 클래스의 생성자에서 할당한 메모리까지 해제 가능.**
- **소멸자가 가상이 아니면 기초클래스 포인터 이용하여 유도클래스 객체를 delete할 때는 메모리 누수 발생.**

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

class First {
private:
        char* strOne;
public:
        First(char* str) {
                strOne = new char[strlen(str) + 1];
        }
        virtual ~First() {
                cout << "~First()" << endl;
                delete [] strOne;
        }
};
```

없애고 수행해 보기.

[결과]
~Second()
~First()

Virtual Destructor
(VirtualDestructor.cpp 1/2)

357-360p

```
class Second : public First {
private:
        char* strTwo;
public:
        Second(char* str1, char* str2) : First(str1) {
                strTwo = new char[strlen(str2) + 1];
        }
        virtual ~Second() {
                cout << "~Second()" << endl;
                delete [] strTwo;
        }
};

int main(void) {
        First* ptr = new Second((char*)"simple", (char*)"complex");
        delete ptr;
        return 0;
}
```

[결과]
~Second()
~First()

First 포인터이지만 Second 객체이므로 destructor는 Second의 것이 호출됨.

360-363p

참조자와 상속
(ReferenceAttribute.cpp 1/3)

- **포인터와 같음**
  - **기초클래스 참조자는 유도클래스 객체를 참조 가능**
  - **virtual 함수의 경우 참조자의 타입이 아닌 객체의 타입에 따라 호출됨.**

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

class First {
public:
        void FirstFunc() {
                cout << "FirstFunc()" << endl;
        }
        virtual void SimpleFunc() {
                cout << "First's SimpleFunc()" << endl;
        }
};
```

[결과]

```
FirstFunc()
SecondFunc()
ThirdFunc()
Third's
SimpleFunc()
FirstFunc()
SecondFunc()
Third's
SimpleFunc()
FirstFunc()
Third's
SimpleFunc()
```

참조자와 상속
(ReferenceAttribute.cpp 2/3)

```
class Second : public First {
public:
        void SecondFunc() {
                cout << "SecondFunc()" << endl;
        }
        virtual void SimpleFunc() {
                cout << "Second's SimpleFunc()" << endl;
        }
};

class Third : public Second {
public:
        void ThirdFunc() {
                cout << "ThirdFunc()" << endl;
        }
        virtual void SimpleFunc() {
                cout << "Third's SimpleFunc()" << endl;
        }
};
```

[결과]

```
FirstFunc()
SecondFunc()
ThirdFunc()
Third's
SimpleFunc()
FirstFunc()
SecondFunc()
Third's
SimpleFunc()
FirstFunc()
Third's
SimpleFunc()
```

360-363p

참조자와 상속
(ReferenceAttribute.cpp 3/3)

```
int main(void) {
        Third obj;
        obj.FirstFunc();
        obj.SecondFunc();
        obj.ThirdFunc();
        obj.SimpleFunc();

        Second& sref = obj;
        sref.FirstFunc();
        sref.SecondFunc();
        sref.SimpleFunc();

        First& fref = obj;
        fref.FirstFunc();
        fref.SimpleFunc();
        return 0;
}
```

virtual 함수이므로 객체 타입에 따른 함수 호출됨.

[결과]

```
FirstFunc()
SecondFunc()
ThirdFunc()
Third's SimpleFunc()
FirstFunc()
SecondFunc()
Third's SimpleFunc()
FirstFunc()
Third's SimpleFunc()
```

363-364p

OOP 프로젝트 06단계
(BankingSystemVer06.cpp 1/11)

- **Account 클래스를 상속하는 클래스 2개 추가**
  NormalAccount
  - **객체 생성 시 이율 등록**

  HighCreditAccount
  - **객체 생성 시 기본이율 등록 가능**
  - **계좌 개설 시 신용등급 A,B,C 중 하나 등록**
  - **A,B,C 등급에 각각 7%, 4%, 2%의 추가 이율 제공**
- **개설 시 입금하는 금액에 대해서는 이자계산 안함**
- **개설 후 입금할 때는 입금 시 즉시 이자 적용**
- **이자 계산 시 소수점 아래 버림**

oop_05.cpp 파일 다운로드 받아 수정하기.

```
#define _CRT_SECURE_NO_WARNINGS
#include <iostream>
#include <cstring>
using namespace std;
const int NAME_LEN = 20;
enum { MAKE = 1, DEPOSIT, WITHDRAW, INQUIRE, EXIT }; // 프로그램  메뉴
enum { LEVEL_A = 7, LEVEL_B = 4, LEVEL_C = 2 }; // 신용등급
enum { NORMAL = 1, CREDIT = 2 }; // 계좌의 종류
```

신용 등급에 따른 추가 이자율.

cin 오류 처리 함수 추가 1
(BankingSystemVer06.cpp 2/11)

- cin 수행 후, 입력이 정상적으로 처리되었는지 판단 : cin.fail()
- 상태 초기화 : cin.clear()
- 입력버퍼 초기화 : cin.ignore

```
int getNumber(const char* title) {
	int retNum;
	int bFail;

	cout << title;
	while (1) {
		cin >> retNum;

		if (cin.fail()) {
			cin.clear(); // 상태 초기화
			cin.ignore(256, '₩n'); // 입력버퍼 초기화
	}
		else
			break;
	}
	return retNum;
}
```

cin 오류 처리 함수 추가 2
(BankingSystemVer06.cpp 3/11)

- cin 수행 후, 입력이 정상적으로 처리되었는지 판단 : cin.fail()
- 상태 초기화 : cin.clear()
- 입력버퍼 초기화 : cin.ignore

```
void getString(const char* title, char* str) {
	int bFail;

	cout << title;
	while (1) {
		cin >> str;

		if (cin.fail()) {
			cin.clear(); // 상태 초기화
			cin.ignore(256, '₩n'); // 입력버퍼 초기화
		}
		else
			break;
	}
}
```

364-372p

OOP 프로젝트 06단계
(BankingSystemVer06.cpp 4/11)

```
class Account
{
private:
	int accID;
	int balance;
	char* cusName;

public:
	Account(int ID, int money, char* name);
	Account(const Account& ref);

	int GetAccID() const;
	virtual void Deposit(int money);
	int Withdraw(int money);
	void ShowAccInfo() const;
	~Account();
};
```

OOP 프로젝트 06단계
(BankingSystemVer06.cpp 5/11)

364-372p

```
class NormalAccount : public Account
{
private:
        int interRate;    // 이자율 %단위
public:
        NormalAccount(int ID, int money, char* name, int rate)
                : Account(ID, money, name), interRate(rate)
        {  }

        virtual void Deposit(int money)
        {
                Account::Deposit(money);          // 원금추가
                Account::Deposit(money * (interRate / 100.0));    // 이자추가
        }
};
```

이자율은 0~100 으로 입력.

364-372p

OOP 프로젝트 06단계
(BankingSystemVer06.cpp 6/11)

```
class HighCreditAccount : public NormalAccount
{
private:
    int specialRate;
public:
    HighCreditAccount(int ID, int money, char* name, int rate, int special)
        : NormalAccount(ID, money, name, rate), specialRate(special)
    {  }

    virtual void Deposit(int money)
    {
        NormalAccount::Deposit(money);          // 원금과 이자추가
        Account::Deposit(money * (specialRate / 100.0));    // 특별이자추가
    }
};
```

364-372p

OOP 프로젝트 06단계
(BankingSystemVer06.cpp 7/11)

```
class AccountHandler
{
private:
        Account* accArr[100];
        int accNum;

public:
        AccountHandler();
        void ShowMenu(void) const;
        void MakeAccount(void);
        void DepositMoney(void);
        void WithdrawMoney(void);
        void ShowAllAccInfo(void) const;
        ~AccountHandler();

protected:
        void MakeNormalAccount(void);
        void MakeCreditAccount(void);
};
```

364-372p

OOP 프로젝트 06단계
(BankingSystemVer06.cpp 8/11)

```
void AccountHandler::ShowMenu(void) const {
        cout << "-----Menu------" << endl;
        cout << "1. 계좌개설" << endl;
        cout << "2. 입     금" << endl;
        cout << "3. 출     금" << endl;
        cout << "4. 계좌정보 전체 출력" << endl;
        cout << "5. 프로그램 종료" << endl;
}
void AccountHandler::MakeAccount(void) {
        int sel;
        cout << "[계좌종류선택]" << endl;
        cout << "1.보통예금계좌 ";
        cout << "2.신용신뢰계좌 " << endl;
        sel = getNumber( "선택: ");

        if (sel == NORMAL)
                MakeNormalAccount();
        else
                MakeCreditAccount();
}
```

364-372p

OOP 프로젝트 06단계
(BankingSystemVer06.cpp 9/11)

```
void AccountHandler::MakeNormalAccount(void)
{
	int id;
	char name[NAME_LEN];
	int balance;
	int interRate;

	cout << "[보통예금계좌 개설]" << endl;
	id = getNumber("계좌ID: ");
	getString("이  름: ", name);
	balance = getNumber("입금액: ");
	interRate = getNumber("이자율: ");
	cout << endl;

	accArr[accNum++] =
		new NormalAccount(id, balance, name, interRate);
}
```

364-372p

OOP 프로젝트 06단계
(BankingSystemVer06.cpp 10/11)

```
void AccountHandler::MakeCreditAccount(void) {
    int id;
    char name[NAME_LEN];
    int balance, interRate, creditLevel;
    cout << "[신용신뢰계좌 개설]" << endl;
    id = getNumber("계좌ID: ");
    getString("이  름: ", name);
    balance = getNumber("입금액: ");
    interRate = getNumber("이자율: ");
    creditLevel = getNumber("신용등급(1toA, 2toB, 3toC): ");
    cout << endl;
    switch (creditLevel) {
    case 1: accArr[accNum++] =
            new HighCreditAccount(id, balance, name, interRate, LEVEL_A);
        break;
    case 2: accArr[accNum++] =
            new HighCreditAccount(id, balance, name, interRate, LEVEL_B);
        break;
    case 3: accArr[accNum++] =
            new HighCreditAccount(id, balance, name, interRate, LEVEL_C);
        break;
    }
}
```

364-372p

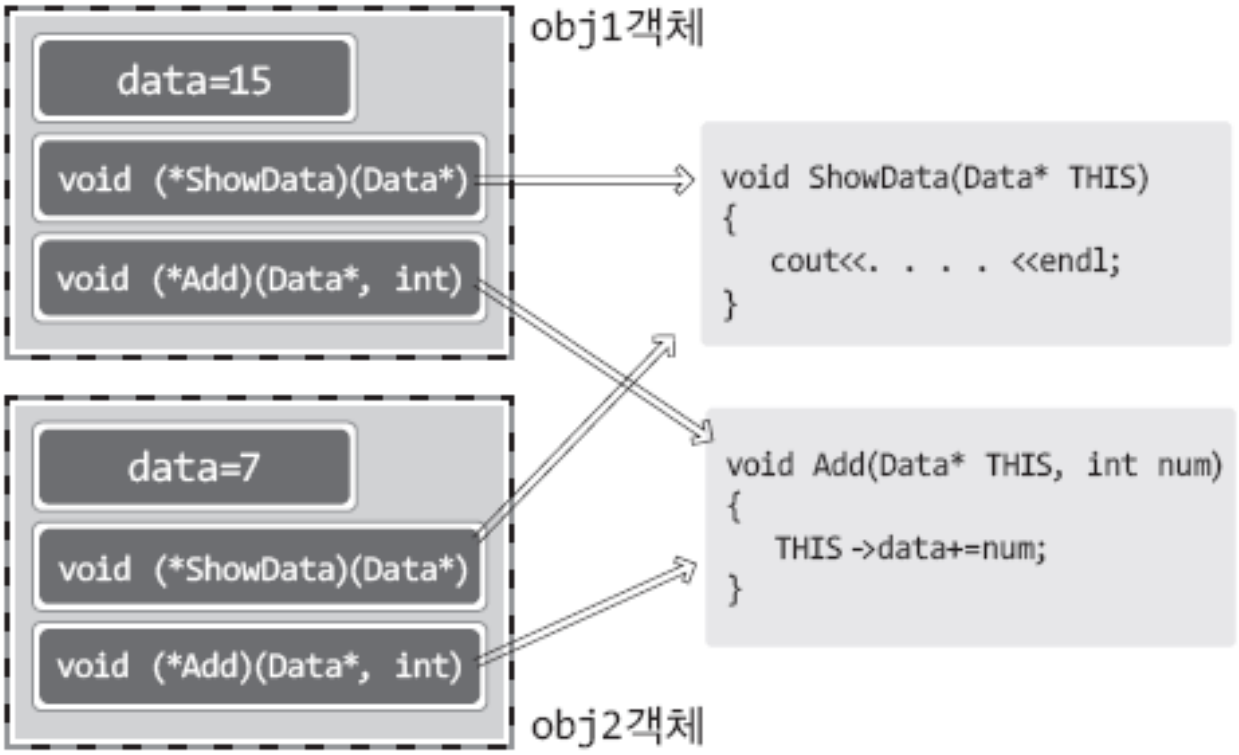
OOP 프로젝트 06단계
(BankingSystemVer06.cpp 11/11)

```
int main(void) {
	AccountHandler manager;
	int choice = 0;
	while (1) {
		manager.ShowMenu();
		choice = getNumber("선택: ");
		cout << endl;
		switch (choice) {
		case MAKE:	manager.MakeAccount(); break;
		case DEPOSIT:	manager.DepositMoney(); break;
		case WITHDRAW:manager.WithdrawMoney(); break;
		case INQUIRE:	manager.ShowAllAccInfo(); break;
		case EXIT:	return 0;
		default:	cout << "Illegal selection.." << endl;
		}
	}
	return 0;
}
```

9. 가상(Virtual)의 원리와 다중상속

클래스 내 함수의 동작원리

- 물리적 : 같은 클래스의 모든 객체가 멤버함수를 공유
- 논리적 : 함수호출 시 객체의 정보가 전달되어 객체 안에 멤버함수가 존재하는 형태



▶ [그림 09-2: obj1과 obj2의 구성]

C++ 스타일의 클래스 함수 표현
(RealObjUnder1.cpp)

```
#include <iostream>
using namespace std;
class Data {
private:
	int data;
public:
	Data(int num) : data(num) {  }
	void ShowData() {
		cout << "Data: " << data << endl;
	}
	void Add(int num) {
		data += num;
	}
};
int main(void) {
	Data obj(15);
	obj.Add(17);
	obj.ShowData();
	return 0;
}
```

C 스타일의 클래스 함수 표현
(RealObjUnder2.cpp)

377-378p

```
#include <iostream>
using namespace std;
typedef struct Data
{
        int data;
        void (*ShowData)(Data*);
        void (*Add)(Data*, int);
} Data;
void ShowData(Data* THIS) { cout << "Data: " << THIS->data << endl; }
void Add(Data* THIS, int num) { THIS->data += num; }

int main(void) {
        Data obj1 = { 15, ShowData, Add };
        Data obj2 = { 7, ShowData, Add };
        obj1.Add(&obj1, 17);
        obj2.Add(&obj2, 9);
        obj1.ShowData(&obj1);
        obj2.ShowData(&obj2);
        return 0;
};
```

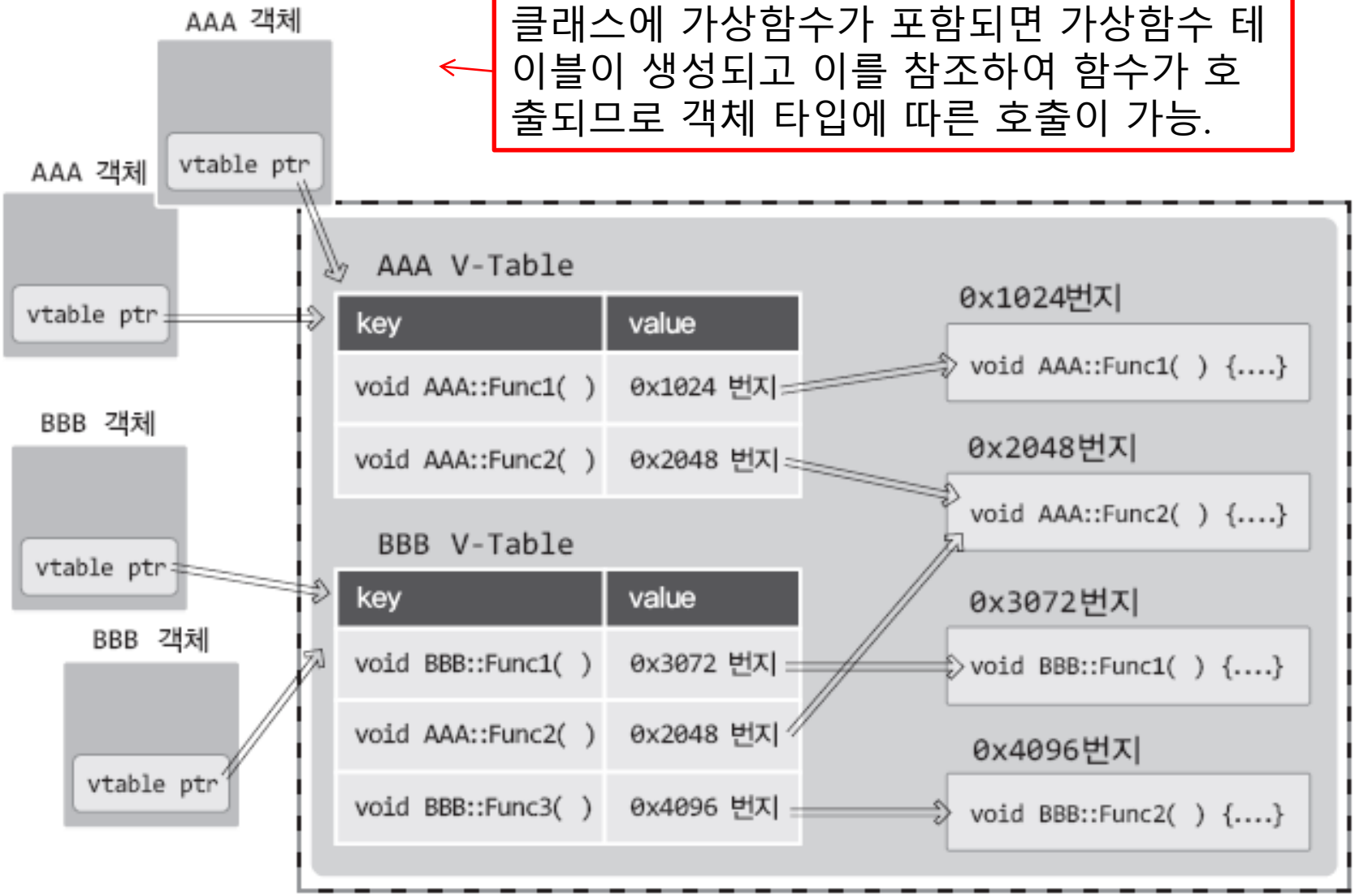
클래스를 흉내 냄.

객체 정보를 달리 주면서 같은 함수를 호출.

379-382p

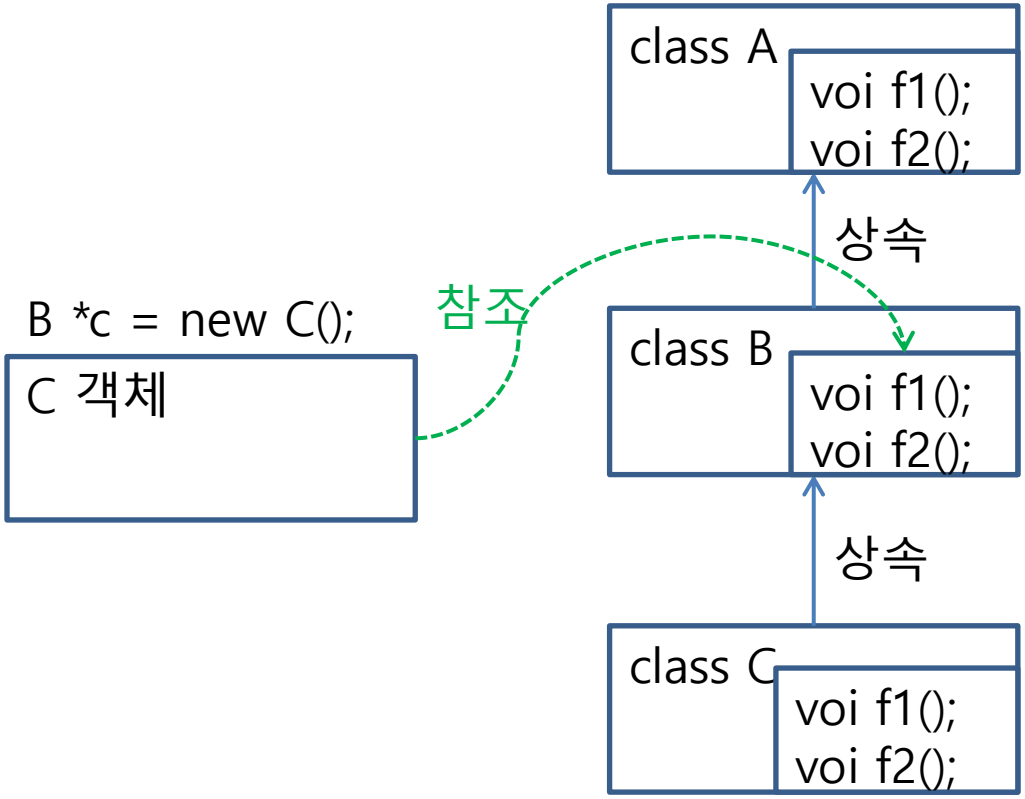
가상함수의 동작원리와 가상함수 테이블

클래스에 가상함수가 포함되면 가상함수 테이블이 생성되고 이를 참조하여 함수가 호출되므로 객체 타입에 따른 호출이 가능.



379-382p

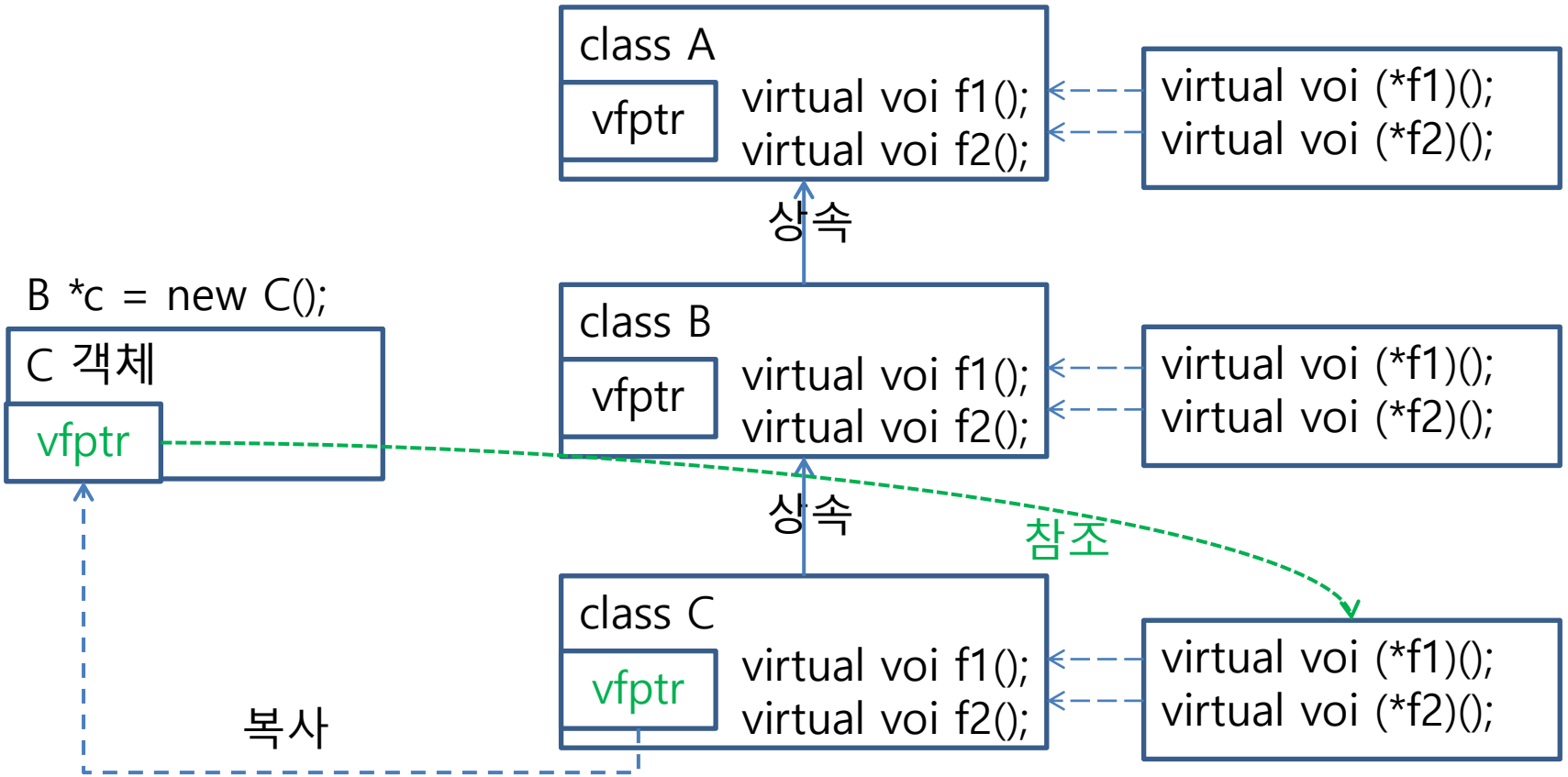
일반함수의 동작원리



virtual 함수가 없는 클래스의 객체인 경우,
포인터 타입에 따른 클래스의 함수를 보게 됨.

379-382p

가상함수의 동작원리와 가상함수 테이블



virtual 함수를 포함한 클래스의 객체인 경우,
각 객체는 객체 클래스의 vfptr을 가짐

383-386p

다중상속

```
#include <iostream>
using namespace std;
class BaseOne {
public:
        void SimpleFunc() { cout << "BaseOneSimple" << endl; }
        void SimpleFuncOne() { cout << "BaseOne" << endl; }
};
class BaseTwo {
public:
        void SimpleFunc() { cout << "BaseTwoSimple" << endl; }
        void SimpleFuncTwo() { cout << "BaseTwo" << endl; }
};
class MultiDrived : public BaseOne, protected BaseTwo {
public:
        void ComplexFunc()      {
                BaseOne::SimpleFunc();
                BaseTwo::SimpleFunc();
                SimpleFuncOne();
                SimpleFuncTwo();
        }
};
int main(void) {
        MultiDrived mdr;
        mdr.ComplexFunc();
        return 0;
}
```

Comma(,)로 연결. 기초 클래스의 상속 형태는 각각 지정 가능.

두 클래스에 같은 이름의 멤버가 있을 때는 기초클래스 이름을 명시하여 호출.

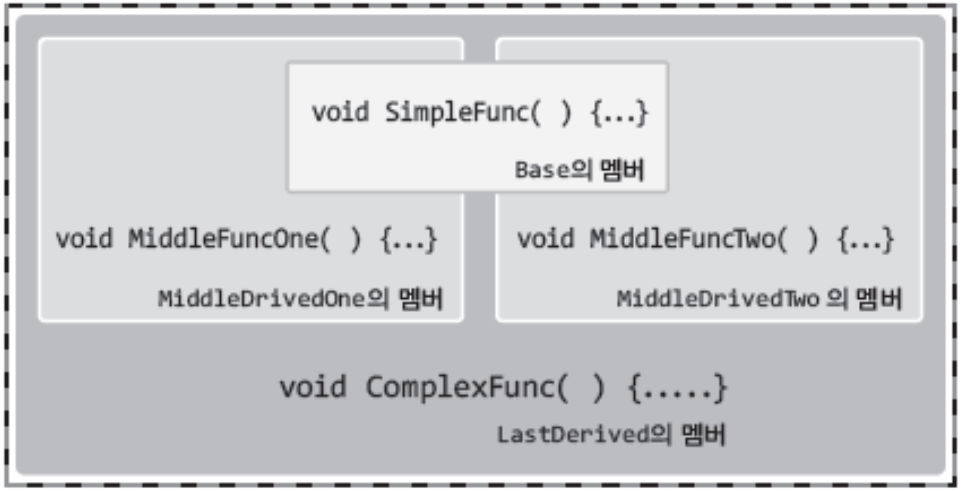
- **둘 이상의 클래스를 동시에 상속하는 것**
- **득보다 실이 많은 문법이므로 될 수 있으면 사용하지 말자.**

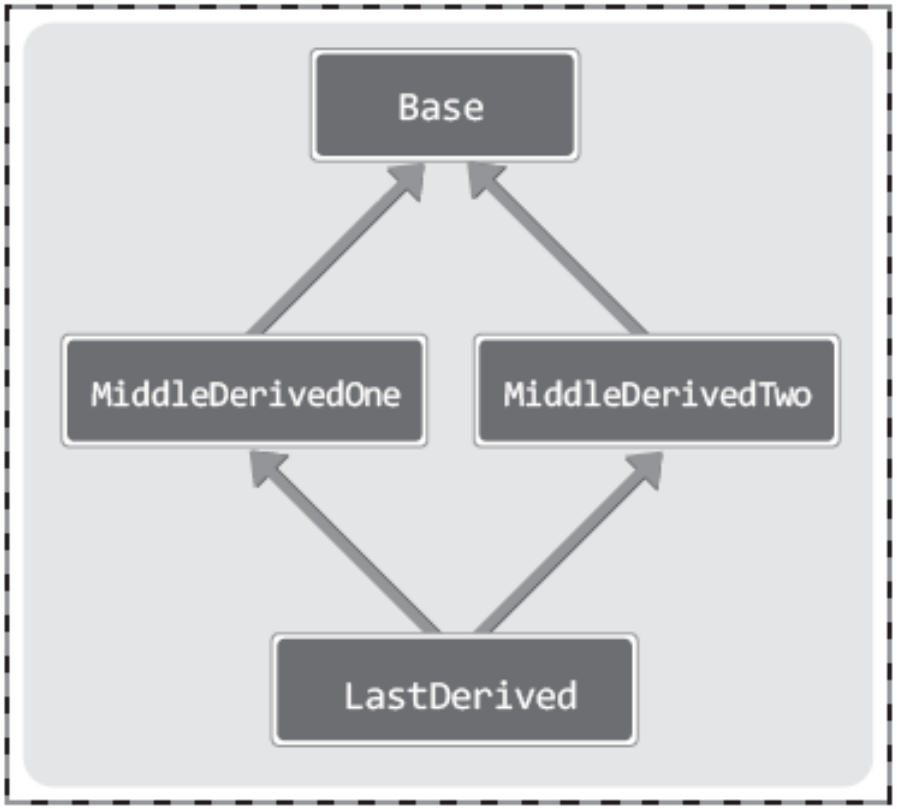
386-390p

가상상속

```
class MiddleDerivedOne : virtual public Base { . . . . };
class MiddleDerivedTwo : virtual public Base { . . . . };
```

간접적으로 2번 상속받는 클래스는 상위에서 상속받을 때 virtual로 선언해 주어 Base 클래스가 1번만 포함되도록 한다.





390-398p

OOP 프로젝트 07단계

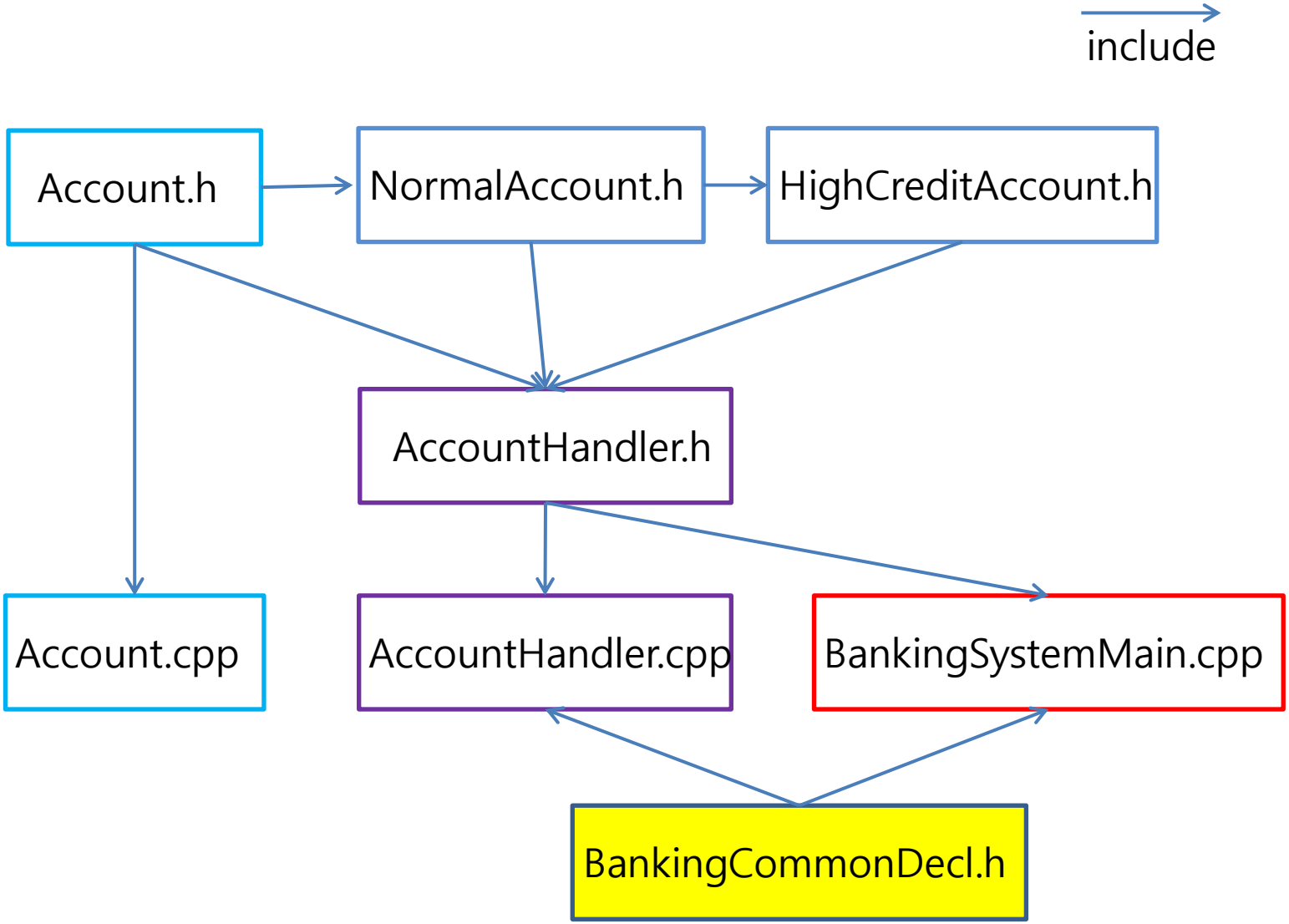
◆ **파일분할**

클래스 별로 .h 와 .cpp 파일 생성

- Account.h, Account.cpp : Account 클래스
- NormalAccount.h : NormalAccount 클래스
- HighCreditAccount.h : HighCreditAccount 클래스
- AccountHandler.h, AccountHandler.cpp : Accounthandler 클래스
- BankingCommonDecl.h : 공통 헤더와 함수 선언
- BankingSystemMain.cpp : main 함수

390-398p

OOP 프로젝트 07단계



390-398p

OOP 프로젝트 07단계
(Account.h)

```
#ifndef __ACCOUNT_H__
#define __ACCOUNT_H__
class Account
{
private:
        int accID;
        int balance;
        char * cusName;
public:
        Account(int ID, int money, char * name);
        Account(const Account & ref);

        int GetAccID() const;
        virtual void Deposit(int money);
        int Withdraw(int money) ;
        void ShowAccInfo() const ;
        ~Account();
};
#endif
```

390-398p

OOP 프로젝트 07단계
(Account.cpp)

```
#define _CRT_SECURE_NO_WARNINGS
#include <cstring>
#include <iostream>
#include "Account.h"
Account::Account(int ID, int money, char * name)
		: accID(ID), balance(money) {
		cusName=new char[strlen(name)+1];
		strcpy(cusName, name);
}
Account::Account(const Account & ref)
		: accID(ref.accID), balance(ref.balance) {
		cusName=new char[strlen(ref.cusName)+1];
		strcpy(cusName, ref.cusName);
}
int Account::GetAccID() const { return accID; }
void Account::Deposit(int money) { balance+=money; }
int Account::Withdraw(int money) {
		if(balance<money)
			return 0;

		balance-=money;
		return money;
}
void Account::ShowAccInfo() const {
		cout<<"계좌ID: "<<accID<<endl;
		cout<<"이  름: "<<cusName<<endl;
		cout<<"잔  액: "<<balance<<endl;
}
Account::~Account() { delete []cusName; }
```

390-398p

OOP 프로젝트 07단계
(NormalAccount.h)

```
#ifndef __NORMAL_ACCOUNT_H__
#define __NORMAL_ACCOUNT_H__

#include "Account.h"
class NormalAccount : public Account
{
private:
	int interRate;	// 이자율 %단위
public:
	NormalAccount(int ID, int money, char * name, int rate)
		: Account(ID, money, name), interRate(rate)
	{ }
	virtual void Deposit(int money)
	{
		Account::Deposit(money);		// 원금추가
		Account::Deposit(money*(interRate/100.0));	// 이자추가
	}
};
#endif
```

390-398p

OOP 프로젝트 07단계
(HighCreditAccount.h)

```
#ifndef __HIGHCREDIT_ACCOUNT_H__
#define __HIGHCREDIT_ACCOUNT_H__

#include "NormalAccount.h"

class HighCreditAccount : public NormalAccount {
private:
    int specialRate;
public:
    HighCreditAccount(int ID, int money, char * name, int rate, int special)
            : NormalAccount(ID, money, name, rate), specialRate(special)
    {  }
    virtual void Deposit(int money) {
        NormalAccount::Deposit(money);          // 원금과 이자추가
        Account::Deposit(money*(specialRate/100.0));    // 특별이자추가
    }
};
#endif
```

390-398p

OOP 프로젝트 07단계
(AccountHandler.h)

```
#ifndef __ACCOUN_HANDLER_H__
#define __ACCOUN_HANDLER_H__
#include "Account.h"
class AccountHandler {
private:
        Account * accArr[100];
        int accNum;
public:
        AccountHandler();
        void ShowMenu(void) const;
        void MakeAccount(void);
        void DepositMoney(void);
        void WithdrawMoney(void);
        void ShowAllAccInfo(void) const;
        ~AccountHandler();
protected:
        void MakeNormalAccount(void);
        void MakeCreditAccount(void);
};
#endif
```

390-398p

OOP 프로젝트 07단계
(AccountHandler.cpp)

```
#include "BankingCommonDecl.h"
#include "AccountHandler.h"
#include "NormalAccount.h"
#include "HighCreditAccount.h"

void AccountHandler::ShowMenu(void) const
{
	cout<<"-----Menu------"<<endl;
	cout<<"1. 계좌개설"<<endl;
	cout<<"2. 입     금"<<endl;
	cout<<"3. 출     금"<<endl;
	cout<<"4. 계좌정보 전체 출력"<<endl;
	cout<<"5. 프로그램 종료"<<endl;
}
.....
.....
.....
```

390-398p

OOP 프로젝트 07단계
(BankingCommonDecl.h)

```
#ifndef __BANKING_COMMON_H__
#define __BANKING_COMMON_H__

#define _CRT_SECURE_NO_WARNINGS
#include <iostream>
#include <cstring>

using namespace std;
const int NAME_LEN=20;

// 프로그램 사용자의 선택 메뉴
enum {MAKE=1, DEPOSIT, WITHDRAW, INQUIRE, EXIT};
enum {LEVEL_A=7, LEVEL_B=4, LEVEL_C=2}; // 신용등급
enum {NORMAL=1, CREDIT=2}; // 계좌의 종류

int getNumber(const char* title);
void getString(const char* title, char* str);

#endif;
```

390-398p

OOP 프로젝트 07단계
(BankingSystemMain.cpp)

```
#include "BankingCommonDecl.h"
#include "AccountHandler.h"

int getNumber(const char* title) {
        ......
}
void getString(const char* title, char* str) {
        ......
}
int main(void) {
        ......
}
```