Object Oriented Programming (IGS2130)

Lab 10

Instructor:
Choonwoo Ryu, Ph.D.

INHA UNIVERSITY

Exercise #1: OOP Project-Step 03

- Upgrade our bank application version 0.2 (lab 7) to 0.3
  - Improve Account class
    - Create a copy constructor to support deep copy

      🡐 `char* m_cusName`
    - Convert all possible member functions into const member functions

  - Use the strcpy_s() function instead of strcpy(), to get rid of the error message in Visual Studio

Account class

```cpp
class Account {
private:
    int m_accID;
    int m_balance;
    char* m_cusName;

public:
    Account(int ID, int balance, char* cname)
        : m_accID{ ID }, m_balance{ balance }
    {
        int len = strlen(cname) + 1;
        m_cusName = new char[len];
        strcpy_s(m_cusName, len, cname);
    }
    ~Account() {
        delete[]m_cusName;
    }
    int GetAccID(void) {
        return m_accID;
    }
    void Deposit(int money) {
        if (money > 0)
            m_balance += money;
    }
```

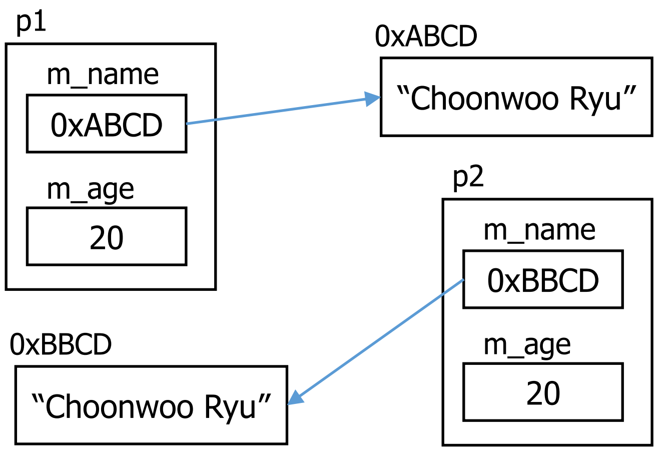
```cpp
    int Withdraw(int money) {
        if ((money < 0) || (money > m_balance))
            return -1;
        m_balance -= money;
        return money;
    }
    void ShowAccInfo(void) {
        cout << "Account ID: " << m_accID
<< endl;
        cout << "Name: " << m_cusName << endl;
        cout << "Balance: " << m_balance
<< endl << endl;
    }
};
```

Example of copy constructor

- Deep copy

```
int main(){
    Person p1("Choonwoo Ryu", 20);
    Person p2{ p1 };
    cout << "p1: "; p1.showInfo();
    cout << "p2: "; p2.showInfo();
    return 0;
}
```

```
p1: Name: Choonwoo Ryu, Age: 20
p2: Name: Choonwoo Ryu, Age: 20
before delete[]m_name;
after delete[]m_name;
before delete[]m_name;
after delete[]m_name;
```



```
#include <iostream>
using namespace std;
class Person {
private:
    char* m_name;
    int m_age;
public:
    Person(const char *name, int age):
        m_age{age}
    {
        m_name = new char[strlen(name) + 1];
        strcpy(m_name, name);
    }
    Person(const Person& cp) :
        m_age{ cp.m_age }
    {
        m_name = new char[strlen(cp.m_name) + 1];
        strcpy(m_name, cp.m_name);
    }
    ~Person() {
        cout << "before delete[]m_name;\n";
        delete[]m_name;
        cout << "after delete[]m_name;\n";
    }
    void showInfo(void) {
        cout << "Name: " << m_name;
        cout << ", Age: " << m_age << endl;
    }
};
```

Example of const member function

- Const member functions
  - Will not modify the object or call any non-const member functions
  - const class objects can only explicitly call const member functions

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

class Something {
public:
    int m_value;
    Something() : m_value{ 0 } { }
    void setValue(int value) { m_value = value; }
    // const member function
    int getValue() const { return m_value; }
};

int main() {
    // calls default constructor
    const Something something{};
    cout << something.getValue() << endl;

    return 0;
}
```

add const keyword after parameter list, but before function body

```
0
```

strcpy_s()

```
strcpy_s( char *dest, rsize_t dest_size, const char *src );

dest: the destination string buffer
dest_size: Size of the destination string buffer in char units
src: Null-terminated source string buffer
```

Exercise #2: OOP Project-Step 04

- Upgrade our bank application version 0.3 to 0.4
  - Create Account handler class named AccountHandler
    - Manage the Account class objects
    - Include Account pointer array as a member variable

```
Account* accArr[MAX_ACC_NUM]; // Account array
int accNum = 0;         // # of accounts
```

    - Include all non-member function as member functions

```
void ShowMenu(void);
void MakeAccount(void);
void DepositMoney(void);
void WithdrawMoney(void);
void ShowAllAccInfo(void);
int GetAccIdx(int);
```

  - Modify main() function so it can run with the AccountHandler object

Definition of AccountHandler class

- AccountHandler class

```
class AccountHandler {
private:
    Account* m_accArr[MAX_ACC_NUM];
    int m_accNum;

    int GetAccIdx(int id) const;
public:
    AccountHandler();
    ~AccountHandler();
    void ShowMenu(void) const;
    void MakeAccount(void);
    void DepositMoney(void);
    void WithdrawMoney(void);
    void ShowAllAccInfo(void) const;
};
```

```
Account* accArr[MAX_ACC_NUM];
int accNum = 0;
```

```
void ShowMenu(void);
void MakeAccount(void);
void DepositMoney(void);
void WithdrawMoney(void);
void ShowAllAccInfo(void);
int GetAccIdx(int);
```

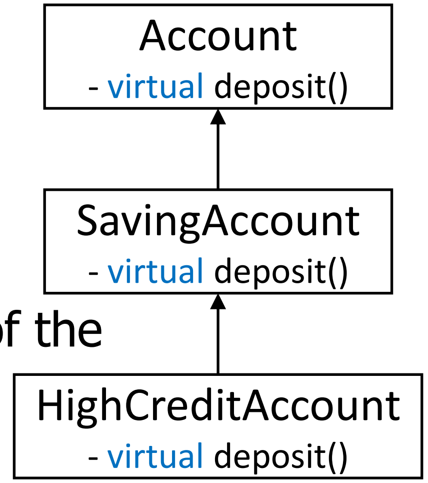
main() function

```cpp
int main(void) {
    int choice, i;

    while (1) {
        ShowMenu();
        cout << "Select menu: ";
        cin >> choice;
        cout << endl;

        switch (bank(choice)) {
        case bank::MAKE:
            MakeAccount();
            break;
        case bank::DEPOSIT:
            DepositMoney();
            break;
        case bank::WITHDRAW:
            WithdrawMoney();
            break;
        case bank::INQUIRE:
            ShowAllAccInfo();
            break;
        case bank::EXIT:
            for (i = 0; i < accNum; i++)
                delete accArr[i];
            return 0;
        default:
            cout << "Illegal selection.." << endl;
        }
    }
    return 0;
}
```

```cpp
int main(void) {
    AccountHandler manager;
    int choice;
    bool run = true;

    while (run) {
        manager.ShowMenu();
        cout << "Select menu: ";
        cin >> choice;
        cout << endl;

        switch (bank(choice)) {
        case bank::MAKE:
            manager.MakeAccount();
            break;
         ………

        default:
            cout << "Illegal selection.." << endl;
        }
    }
    return 0;
}
```

Exercise #3: OOP Project-Step 05

- Create two classes to our bank application
  - Class name: SavingAccount
    - Inherit from Account class
    - Contains *interest rate* member
      - The *interest rate*(%) is determined by the argument of the class constructor
  - Class name: HighCreditAccount
    - Inherit from SavingAccount class
    - Has two interest rate: *interest rate* (defined in SavingAccount class) and *special interest rate*
      - *special interest rate*(%): depending on the given *credit rating* (A, B, and C), additional interest will be set at 7%, 4% and 2%, respectively.



- To make the program simple, the interest will be paid when you make a deposit.
- Make the deposit() member function virtual, so AccountHandler class can call the most derived deposit() member function

Exercise #3: OOP Project-Step 05

- Modify AccountHandler class
  - Change the program menu to reflect the account type
    - Modify MakeAccount() function to select account type. According to the selected account type, this function will call MakeSavingAccount() or MakeHighCreditAccount().
    - Create a MakeSavingAccount() function that creates a saving account
    - Create a MakeHighCreditAccount() function that creates a high credit account

Exercise #3: OOP Project-Step 05

- The program should work like the example below:

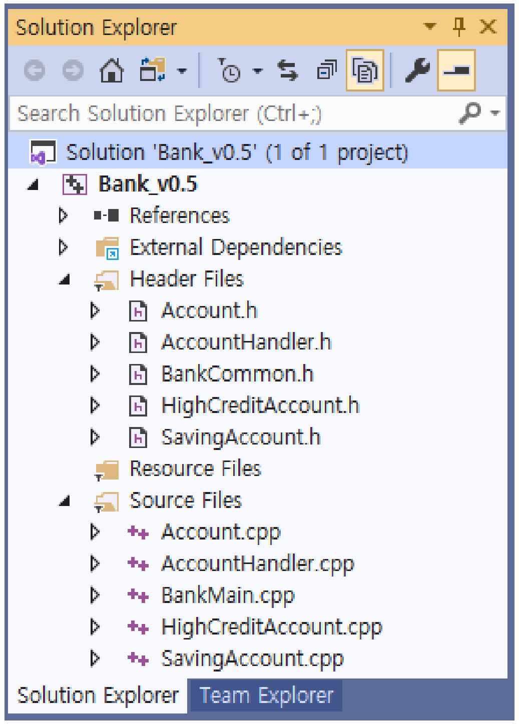
```
-----Menu------
1. Make Accout
2. Deposit
3. Withdrawal
4. Display all
5. Exit program
Select menu: 1

[Select Account Type]
 1. Saving Account
 2. High Credit Account
 Select: 1
[Make Saving Account]
Account ID: 1
Customer Name: Kim
Deposit amount: 10000
Interest Rate: 4
```

```
-----Menu------
1. Make Accout
2. Deposit
3. Withdrawal
4. Display all
5. Exit program
Select menu: 1

[Select Account Type]
 1. Saving Account
 2. High Credit Account
 Select: 2
[Make High Credit Account]
Account ID: 2
Customer Name: Lee
Deposit amount: 10000
Interest Rate: 4
Credit Rating(A:1, B:2, C:3): 1
```

```
-----Menu------
1. Make Accout
2. Deposit
3. Withdrawal
4. Display all
5. Exit program
Select menu: 4

Account ID: 1
Name: Kim
Balance: 10400

Account ID: 2
Name: Lee
Balance: 11100
```

Exercise #3: OOP Project-Step 05

- Dividing the source code into several files
  - Usually, for each class, the class definition is saved in a .h file and the implementation is saved in another .cpp file.
  - Organize the source code by dividing it into a total of 10 files.
    - BankCommon.h
    - Account.h
    - Account.cpp
    - SavingAccount.h
    - SavingAccount.cpp
    - HighCreditAccount.h
    - HighCreditAccount.cpp
    - AccountHandler.h
    - AccountHandler.cpp
    - BankMain.cpp



```
Solution 'Bank_v0.5' (1 of 1 project)
  Bank_v0.5
    References
    External Dependencies
    Header Files
      Account.h
      AccountHandler.h
      BankCommon.h
      HighCreditAccount.h
      SavingAccount.h
    Resource Files
    Source Files
      Account.cpp
      AccountHandler.cpp
      BankMain.cpp
      HighCreditAccount.cpp
      SavingAccount.cpp
```

  - Include all the files in your project and make the project compliable.

Definition of SavingAccount class and HighCreditAccount class

```
class SavingAccount : public Account {
private:
    int m_InterestRate; // %
public:
    SavingAccount(int ID, int balance, char* cname, int rate);
    virtual void Deposit(int money);
};
```

```
class HighCreditAccount : public SavingAccount {
private:
    int m_SpecialRate;
public:
    HighCreditAccount(int ID, int balance, char* cname, int rate, int special);
    virtual void Deposit(int money);
};
```