Turnin 作業 10

- Turnin Code: cpp.hw10
- Due Date: ~~6/3 Wednesday~~ 6/4 Thursday 23:59:00 (midnight) **Hard Deadline**

繳交方式說明

本次作業繳交將以資料夾的形式繳交，需要為每一題建立一個資料夾（資料夾名稱為該題題目前方之代號，第一題為「p1」，第二題為「p2」，餘以此類推）。

繳交說明可參考「Turnin 作業 4」中 p6 到 p10

任何未依照正確繳交格式的檔案將以 0 分計。



本文使用「 」及「 」代表「空白字元」與「Enter 換行字元」，並且將使用者輸入的部份使用灰階方式顯示。另外，題目的執行結果中，如果出現「(」、「)」、「:」、「;」、「.」與「,」等符號，皆為英文半形！

多型與覆寫允許同一個類別中定義許多名字相同但實作內容不同的函式。舉例來說：貓會「喵喵」叫、狗會「汪汪」叫，我們換成程式碼的寫法分別會是 `cat.meow();`、`dog.bark();`。如果他們現在需要繼承一個動物的類別 Animal，大多數動物會發出聲音，因此該類別中有一個成員函式 `virtual [data type] make_sound();`。我們可以根據此函式實作 `cat.make_sound();` 使其發出「喵喵」聲；實作 `dog.make_sound();` 使其發出「汪汪」聲。

透過這種方式，我們可以減少 `dog.meow();` 或是 `cat.bark();` 此類無法理解與實作明顯不合理的函式名稱，並提升程式碼的可讀性與可維護性。

p1 學生類別 1（靜態多型）

在物件導向程式設計中，為了處理不同類型的物件，我們常會使用繼承與多型。本題已經設計四個身分：一般學生（Student）、外籍生（ForeignStudent）、本地生（LocalStudent，非外籍生）與本地兼職學生（LocalParttimeStudent）。

由於本題限制不使用虛擬函式（virtual 關鍵字），我們必須在基底類別中定義 StudentType 列舉（Enumeration）並在物件建立時正確設定其類型。最後，在函式 show_all_info 中，透過檢查學生的類型並進行正確的類別指標強制轉型，來達到靜態多型（Static Polymorphism）的效果，依序輸出所有學生的詳細資訊。

Student 類別相關定義請參考下列的 student.h 程式（請依據檔名旁的路徑至 ws 取得程式碼）：

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

enum StudentType
{
    Normal,
    Foreign,
    Local,
    LocalParttime
};

class Student
{
public:
    string name;
    StudentType type;
    Student();
    void showInfo();
};
```

ForeignStudent 類別相關定義請參考下列的 ForeignStudent.h 程式（請依據檔名旁的路徑至 ws 取得程式碼）：

```
#include "Student.h"

class ForeignStudent : public Student
{
public:
    string nationality;
    ForeignStudent();
    void showInfo();
};
```

LocalStudent 類別相關定義請參考下列的 LocalStudent.h 程式（請依據檔名旁的路徑至 ws 取得程式碼）：

```
#include "ForeignStudent.h"

class LocalStudent : public Student
{
public:
    LocalStudent();
    void showInfo();
};
```

LocalParttimeStudent 類別相關定義請參考下列的 LocalParttimeStudent.h 程式（請依據檔名旁的路徑至 ws 取得程式碼）：

```
#include "LocalStudent.h"

class LocalParttimeStudent : public LocalStudent
{
public:
    LocalParttimeStudent();
    void showInfo();
};
```

請參考下列的 main.cpp 程式（請依據檔名旁的路徑至 ws 取得程式碼）：

```
#include "ShowAllInfo.h"
using namespace std;
int main()
{
    int amt = 0;
    cout << "Input student quantity: ";
    cin >> amt;
    Student *csie2c[amt];
    int i;
    for (int i = 0; i < amt; i++)
    {
        char student_type;
        cout << "Input student type(S/F/L/P): ";
        cin >> student_type;

        switch (student_type)
        {
            case 'S':
                csie2c[i] = new Student;
                cout << "[Student] created." << endl;
                break;
            case 'F':
                csie2c[i] = new ForeignStudent;
                cout << "[ForeignStudent] created." << endl;
                break;
            case 'L':
                csie2c[i] = new LocalStudent;
                cout << "[LocalStudent] created." << endl;
                break;
            case 'P':
                csie2c[i] = new LocalParttimeStudent;
                cout << "[LocalParttimeStudent] created." << endl;
```

```
                break;
            }
        }
        cout << "----------------------" << endl;

        show_all_info(csie2c, amt);
}
```

除了完成四個類別 Student、ForeignStudent、LocalStudent 與 LocalParttimeStudent 的實作檔，還需要實作 main.cpp 中第 38 行的 show_all_info 函式。在此函式中，你需要透過檢查每個物件所儲存的 type 變數，將指標強制轉型為正確的子類別，進而呼叫對應版本的 *[object]->showInfo();，以達到靜態多型的效果。

檔案名稱	**檔案用途與撰寫要求**	**-**
main.cpp	主程式	題目提供原始碼
Student.h	Student 類別的定義	題目提供原始碼
Student.cpp	Student 類別的實作	需繳交
ForeignStudent.h	ForeignStudent 類別的定義	題目提供原始碼
ForeignStudent.cpp	ForeignStudent 類別的實作	需繳交
LocalStudent.h	LocalStudent 類別的定義	題目提供原始碼
LocalStudent.cpp	LocalStudent 類別的實作	需繳交
LocalParttimeStudent.h	LocalParttimeStudent 類別的定義	題目提供原始碼
LocalParttimeStudent.cpp	LocalParttimeStudent 類別的實作	需繳交
ShowAllInfo.h	show_all_info 函式的定義	需繳交
ShowAllInfo.cpp	show_all_info 函式的實作	需繳交

請完成名為 **Student.cpp**、**ForeignStudent.cpp**、**LocalStudent.cpp**、**LocalParttimeStudent.cpp**、**ShowAllInfo.h** 與 **ShowAllInfo.cpp** 的 C++ 程式。

本題的相關程式將使用以下的 Makefile 進行編譯：

```
all: main.cpp ShowAllInfo.o LocalParttimeStudent.o LocalStudent.o
ForeignStudent.o Student.o
    c++ main.cpp ShowAllInfo.o LocalParttimeStudent.o LocalStudent.o
ForeignStudent.o Student.o
ShowAllInfo.o: ShowAllInfo.cpp ShowAllInfo.h LocalParttimeStudent.h
LocalStudent.h ForeignStudent.h Student.h
    c++ -c ShowAllInfo.cpp
LocalParttimeStudent.o: LocalParttimeStudent.cpp LocalParttimeStudent.h
LocalStudent.h ForeignStudent.h Student.h
    c++ -c LocalParttimeStudent.cpp
LocalStudent.o: LocalStudent.cpp LocalStudent.h ForeignStudent.h Student.h
    c++ -c LocalStudent.cpp
ForeignStudent.o: ForeignStudent.cpp ForeignStudent.h Student.h
    c++ -c ForeignStudent.cpp
Student.o: Student.cpp Student.h
    c++ -c Student.cpp
clean:
```

```
    rm -f *.o *~ *.*~ a.out*
```

此題的執行結果可參考如下：

```
[3:23▲user@ws▲p1]./a.out↵
Input▲student▲quantity:▲5↵
Input▲student▲type(S/F/L/P):▲S↵
[Student]▲created.↵
Input▲student▲type(S/F/L/P):▲F↵
[ForeignStudent]▲created.↵
Input▲student▲type(S/F/L/P):▲L↵
[LocalStudent]▲created.↵
Input▲student▲type(S/F/L/P):▲P↵
[LocalParttimeStudent]▲created.↵
Input▲student▲type(S/F/L/P):▲F↵
[ForeignStudent]▲created.↵
----------------------↵
I'm▲a▲student.↵
I'm▲a▲foreign▲student.↵
I'm▲a▲local▲student.↵
I'm▲a▲local▲part-time▲student.↵
I'm▲a▲foreign▲student.↵
[3:23▲user@ws▲p1]
```



- 本題相關的程式碼路徑已註明於檔名右側，同學們可以透過路徑複製到自己的家目錄。
- 類別撰寫要求請參考上方表格。
- 本題若有使用浮點數的需求，請使用 double 型態。
- 本題應繳交檔案如下：
  - Student.cpp
  - ForeignStudent.cpp
  - LocalStudent.cpp
  - LocalParttimeStudent.cpp
  - ShowAllInfo.h
  - ShowAllInfo.cpp

p2 學生類別 2（動態多型與覆寫）

本題併入下次作業 cpp.hw11。本次作業此題無需繳交。

承上題，本題將要求同學們用動態多型的方式輸出與上題相同內容。

由於本題限制使用必須虛擬函式（virtual 關鍵字），讓在不同物件執行相同名稱相同引數的函式時，不再需要透過強制轉型的方式就可直接呼叫自己的版本，這就是動態多型（Dynamic Polymorphism）的效果。

Student 類別相關定義請參考下列的 student.h 程式（請依據檔名旁的路徑至 ws 取得程式碼，請修改後繳交）

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

enum StudentType
{
    Normal,
    Foreign,
    Local,
    LocalParttime
};

class Student
{
public:
    string name;
    StudentType type;
    Student();
    void showInfo();
};
```

ForeignStudent 類別相關定義請參考下列的 ForeignStudent.h 程式（請依據檔名旁的路徑至 ws 取得程式碼，請依照需求修改（不修改亦可）但一定要繳交）

```
#include "Student.h"

class ForeignStudent : public Student
{
public:
    string nationality;
    ForeignStudent();
    void showInfo();
};
```

LocalStudent 類別相關定義請參考下列的 LocalStudent.h 程式（請依據檔名旁的路徑至 ws 取得程式碼，請依照需求修改（不修改亦可）但一定要繳交）

```
#include "ForeignStudent.h"
```

```
class LocalStudent : public Student
{
public:
    LocalStudent();
    void showInfo();
};
```

LocalParttimeStudent 類別相關定義請參考下列的 LocalParttimeStudent.h 程式（請依據檔名旁的路徑至 ws 取得程式碼，請依照需求修改（不修改亦可）但一定要繳交□□

```
#include "LocalStudent.h"

class LocalParttimeStudent : public LocalStudent
{
public:
    LocalParttimeStudent();
    void showInfo();
};
```

請參考下列的 main.cpp 程式（請依據檔名旁的路徑至 ws 取得程式碼）：

```
#include "ShowAllInfo.h"
using namespace std;
int main()
{
    int amt = 0;
    cout << "Input student quantity: ";
    cin >> amt;
    Student *csie2c[amt];
    int i;
    for (int i = 0; i < amt; i++)
    {
        char student_type;
        cout << "Input student type(F/L/P): ";
        cin >> student_type;

        switch (student_type)
        {
            case 'F':
                csie2c[i] = new ForeignStudent;
                cout << "[ForeignStudent] created." << endl;
                break;
            case 'L':
                csie2c[i] = new LocalStudent;
                cout << "[LocalStudent] created." << endl;
```

```
                break;
            case 'P':
                csie2c[i] = new LocalParttimeStudent;
                cout << "[LocalParttimeStudent] created." << endl;
                break;
        }
    }
    cout << "----------------------" << endl;

    show_all_info(csie2c, amt);

}
```

除了修改四個類別 Student、ForeignStudent、LocalStudent 與 LocalParttimeStudent 的定義檔，還需要實作 main.cpp 中第 38 行的 show_all_info 函式。在此函式中，你需要直接呼叫物件的 showInfo() 函式，不需特地對物件強制轉型，這就是動態多型。

檔案名稱	**檔案用途與撰寫要求**	**-**
main.cpp	主程式	題目提供原始碼
Student.h	Student 類別的定義，未正確修改此類別者，本題不予計分	題目提供原始碼，修改後繳交
Student.cpp	Student 類別的實作	需繳交，可繳交 p1 的同名檔案
ForeignStudent.h	ForeignStudent 類別的定義	題目提供原始碼，修改後繳交
ForeignStudent.cpp	ForeignStudent 類別的實作	需繳交，可繳交 p1 的同名檔案
LocalStudent.h	LocalStudent 類別的定義	題目提供原始碼，修改後繳交
LocalStudent.cpp	LocalStudent 類別的實作	需繳交，可繳交 p1 的同名檔案
LocalParttimeStudent.h	LocalParttimeStudent 類別的定義	題目提供原始碼，修改後繳交
LocalParttimeStudent.cpp	LocalParttimeStudent 類別的實作	需繳交，可繳交 p1 的同名檔案
ShowAllInfo.h	show_all_info 函式的定義	需繳交
ShowAllInfo.cpp	show_all_info 函式的實作，函式使用強制轉型呼叫正確版本函式者，不予計分	需繳交

請完成名為 **Student.h、Student.cpp、ForeignStudent.h、ForeignStudent.cpp、LocalStudent.h、LocalStudent.cpp、LocalParttimeStudent.h、LocalParttimeStudent.cpp、ShowAllInfo.h** 與 **ShowAllInfo.cpp** 的 C++ 程式。其他在 p1 已經繳交的類別實作檔亦可於此題重用，但你仍然必須為此題再一次繳交，否則 0 分計。

本題的相關程式將使用以下的 Makefile 進行編譯：

```
all: main.cpp ShowAllInfo.o LocalParttimeStudent.o LocalStudent.o
ForeignStudent.o Student.o
```

```
    c++ main.cpp ShowAllInfo.o LocalParttimeStudent.o LocalStudent.o
ForeignStudent.o Student.o
ShowAllInfo.o: ShowAllInfo.cpp ShowAllInfo.h LocalParttimeStudent.h
LocalStudent.h ForeignStudent.h Student.h
    c++ -c ShowAllInfo.cpp
LocalParttimeStudent.o: LocalParttimeStudent.cpp LocalParttimeStudent.h
LocalStudent.h ForeignStudent.h Student.h
    c++ -c LocalParttimeStudent.cpp
LocalStudent.o: LocalStudent.cpp LocalStudent.h ForeignStudent.h Student.h
    c++ -c LocalStudent.cpp
ForeignStudent.o: ForeignStudent.cpp ForeignStudent.h Student.h
    c++ -c ForeignStudent.cpp
Student.o: Student.cpp Student.h
    c++ -c Student.cpp
clean:
    rm -f *.o *~ *.*~ a.out*
```

此題的執行結果可參考如下：

```
[3:23▲user@ws▲p2]./a.out↵
Input▲student▲quantity:▲5↵
Input▲student▲type(F/L/P):▲F↵
[ForeignStudent]▲created.↵
Input▲student▲type(F/L/P):▲L↵
[LocalStudent]▲created.↵
Input▲student▲type(F/L/P):▲P↵
[LocalParttimeStudent]▲created.↵
Input▲student▲type(F/L/P):▲F↵
[ForeignStudent]▲created.↵
Input▲student▲type(F/L/P):▲P↵
[LocalParttimeStudent]▲created.↵
----------------------↵
I'm▲a▲foreign▲student.↵
I'm▲a▲local▲student.↵
I'm▲a▲local▲part-time▲student.↵
I'm▲a▲foreign▲student.↵
I'm▲a▲local▲part-time▲student.↵
[3:23▲user@ws▲p2]
```



- 本題相關的程式碼路徑已註明於檔名右側，同學們可以透過路徑複製到自己的家目錄。
- 類別撰寫要求請參考上方表格
- 本題若有使用浮點數的需求，請使用 double 型態。
- 本題應繳交檔案如下：
  - Student.h



- Student.cpp
- ForeignStudent.h
- ForeignStudent.cpp
- LocalStudent.h
- LocalStudent.cpp
- LocalParttimeStudent.h
- LocalParttimeStudent.cpp
- ShowAllInfo.h
- ShowAllInfo.cpp

p3 銀行帳戶類別實作 1（靜態多型）

接續 cpp.hw9 的 p3□鳳梨銀行對儲蓄帳戶新增了一系列的規則。

BankAccount 類別相關定義請參考下列的 bank.h 程式（請依據檔名旁的路徑至 ws 取得程式碼）：

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

class BankAccount
{
protected:
    string name;
    int balance;
    struct
    {
        string record;
        int count = 1;
    } transaction;

public:
    BankAccount(string name_str);
    int get_balance();
    void set_balance(int new_balance);
    string get_name();
    void write_record(string record);
    void show_record();
    void save(int amount);
    void show_balance();
    void withdraw(int amount);
    void transfer(int amount, BankAccount &recipient);
};
```

SavingAccount 類別相關定義請參考下列的 saving_account.h 程式（請依據檔名旁的路徑至 ws 取得程式碼）：

```
#include "bank.h"

class SavingAccount : public BankAccount
{
private:
    double deposit_interest_rate = 0.016;
    int handling_fee = 10;

public:
    SavingAccount(string name_str);
    void calculate_interest(int year);
    void show_balance();
    void withdraw(int amount);
    void transfer(int amount, BankAccount &recipient);
};
```

請參考下列的 main.cpp 程式（請依據檔名旁的路徑至 ws 取得程式碼）：

```
#include "saving_account.h"
using namespace std;

int main()
{
    BankAccount * customer[3];
    BankAccount Justin("Justin");
    BankAccount Amy("Amy");
    SavingAccount Phineas("Phineas");
    customer[0] = &Justin;
    customer[1] = &Amy;
    customer[2] = &Phineas;

    cout << "[1] ------------------------" << endl;
    customer[0]->show_balance();
    customer[1]->show_balance();

    cout << "[2] ------------------------" << endl;
    customer[1]->save(4090);
    customer[1]->save(0);
    customer[1]->save(-216);

    cout << "[3] ------------------------" << endl;
    customer[1]->transfer(1080, Justin);
    customer[1]->transfer(4050, Justin);
    customer[1]->transfer(0, Justin);
    customer[1]->transfer(-512, Justin);

    cout << "[4] ------------------------" << endl;
```

```
    customer[1]->withdraw(12000);
    customer[1]->withdraw(3000);
    customer[1]->withdraw(0);

    cout << "[5] ------------------------" << endl;
    customer[0]->show_balance();
    customer[1]->show_balance();

    cout << "[6] ------------------------" << endl;
    customer[1]->show_record();

    cout << "[7] ------------------------" << endl;
    ???     show_balance();
    ???     save(10000);
    ???     show_balance();
    ???     calculate_interest(10); // for 10 years
    ???     show_balance();

    cout << "[8] ------------------------" << endl;
    ???     withdraw(450);
    ???     show_balance();

    cout << "[9] ------------------------" << endl;
    ???     show_balance();
    ???     transfer(10, Amy);
    ???     show_balance();
    customer[1]->show_balance();

    cout << "[10] ------------------------" << endl;
    ???     withdraw(11231);
    ???     show_balance();
    ???     transfer(11231, Justin);
    ???     show_balance();
    ???     withdraw(11230);
    ???     show_balance();

    cout << "[11] ------------------------" << endl;
    ???     show_record();

    return 0;
}
```

鳳梨銀行 SavingAccount 固有規則：

1. 本利和的計算以**年**為單位且年利率為 1.6%：$期末本利和 = 期初本金金額 \times (1 + 每期利率)^{期數}$，計算結果請無條件捨棄小數。

邪惡的鳳梨銀行希望 SavingAccount 使用者將錢持續存在銀行裡面而使用了複利計算本利合，但是銀行發現複利的計算方式沒辦法吸引對於那些存款較少的使用者持續讓錢儲存在銀行，因此頒訂了新規定：

1. 不論提款多少錢，必須支付 10 元的手續費費用。提款時，$(欲提領金額 + 手續費的金額) \lt 帳戶內

的餘額$ 時，相當於沒有足夠的錢進行交易導致交易失敗。
2. 轉帳同提款的新規則。

上述規定僅限制於 SavingAccount 使用者。

在 main.cpp 的第七階段至第十一階段中，存在大量的「？？？」 留白。該段程式碼旨在操作儲蓄帳戶指標 customer[2]指向 Phineas然而，由於 BankAccount 類別中的相關功能並未宣告為虛擬函式（virtual）若直接使用指標呼叫 customer[2]→calculate_interest() 將會導致編譯失敗；且呼叫 withdraw 或 transfer 也只會觸發父類別版本，鳳梨銀行將無法收取任何的手續費。

與 p1 相同，請用靜態多型的方式將 main.cpp 中的「？？？」進行正確取代。請注意，該檔案越過指標 customer[2] 逕行呼叫 Phineas 物件下任一成員函式以及修改指定區域外程式碼者，該題 0 分計。

檔案名稱	**檔案用途與撰寫要求**	**-**
main.cpp	主程式，修改的部分未使用指標呼叫者，不予計分	題目提供原始碼，修改後繳交
bank.h	BankAccount 類別的定義	題目提供原始碼
bank.cpp	BankAccount 類別的實作	需繳交
saving_account.h	SavingAccount 類別的定義	題目提供原始碼
saving_account.cpp	SavingAccount 類別的實作	需繳交

請完成名為 **main.cpp**、**bank.cpp** 與 **saving_account.cpp** 的 C++ 語言程式，相關檔案撰寫要求已詳述於上方表格。

註：本題輸出請勿沿用上次作業的 ANSI 色碼（用於控制終端機文字樣式），否則視為錯誤。

本題將使用以下的 Makefile 進行編譯：

```
all: main.cpp bank.o saving_account.o bank.h saving_account.h
    c++ main.cpp bank.o saving_account.o
saving_account.o: bank.h saving_account.cpp saving_account.h
    c++ saving_account.cpp -c
bank.o: bank.cpp bank.h
    c++ bank.cpp -c
clean:
    rm -f *.o *~ *.*~ a.out*
```

此題的執行結果可參考如下：

```
[3:23▲user@ws▲p3]./a.out↵
[1]▲----------------------↵
Justin's▲current▲balance:▲0↵
Amy's▲current▲balance:▲0↵
[2]▲----------------------↵
[▲SUCCESS▲]▲Amy▲saved▲\$4090↵
```

[▲▲▲FAIL▲▲]▲Amy▲failed▲to▲save▲\$0↵
[▲▲▲FAIL▲▲]▲Amy▲failed▲to▲save▲\$-216↵
[3]▲----------↵
[▲SUCCESS▲]▲Amy▲transfered▲\$1080▲to▲Justin↵
[▲▲▲FAIL▲▲]▲Amy▲failed▲to▲transfer▲\$4050▲to▲Justin↵
[▲▲▲FAIL▲▲]▲Amy▲failed▲to▲transfer▲\$0▲to▲Justin↵
[▲▲▲FAIL▲▲]▲Amy▲failed▲to▲transfer▲\$-512▲to▲Justin↵
[4]▲----------↵
[▲▲▲FAIL▲▲]▲Amy▲failed▲to▲withdraw▲\$12000↵
[▲SUCCESS▲]▲Amy▲withdrew▲\$3000↵
[▲▲▲FAIL▲▲]▲Amy▲failed▲to▲withdraw▲\$0↵
[5]▲----------↵
Justin's▲current▲balance:▲1080↵
Amy's▲current▲balance:▲10↵
[6]▲----------↵
Amy's▲transaction▲record:▲↵
1▲[▲SUCCESS▲]▲Amy▲saved▲\$4090↵
2▲[▲▲▲FAIL▲▲]▲Amy▲failed▲to▲save▲\$0↵
3▲[▲▲▲FAIL▲▲]▲Amy▲failed▲to▲save▲\$-216↵
4▲[▲SUCCESS▲]▲Amy▲transfered▲\$1080▲to▲Justin↵
5▲[▲▲▲FAIL▲▲]▲Amy▲failed▲to▲transfer▲\$4050▲to▲Justin↵
6▲[▲▲▲FAIL▲▲]▲Amy▲failed▲to▲transfer▲\$0▲to▲Justin↵
7▲[▲▲▲FAIL▲▲]▲Amy▲failed▲to▲transfer▲\$-512▲to▲Justin↵
8▲[▲▲▲FAIL▲▲]▲Amy▲failed▲to▲withdraw▲\$12000↵
9▲[▲SUCCESS▲]▲Amy▲withdrew▲\$3000↵
10▲[▲▲▲FAIL▲▲]▲Amy▲failed▲to▲withdraw▲\$0↵
[7]▲----------↵
Phineas's▲current▲balance:▲0▲(Savings▲Account)↵
[▲SUCCESS▲]▲Phineas▲saved▲\$10000↵
Phineas's▲current▲balance:▲10000▲(Savings▲Account)↵
Phineas▲accrued▲interest▲for▲10▲yr(s).↵
Phineas's▲current▲balance:▲11720▲(Savings▲Account)↵
[8]▲----------↵
[▲SUCCESS▲]▲Phineas▲withdrew▲\$450▲with▲\$10▲handling▲fee↵
Phineas's▲current▲balance:▲11260▲(Savings▲Account)↵
[9]▲----------↵
Phineas's▲current▲balance:▲11260▲(Savings▲Account)↵
[▲SUCCESS▲]▲Phineas▲transfered▲\$10▲to▲Amy▲with▲\$10▲handling▲fee↵
Phineas's▲current▲balance:▲11240▲(Savings▲Account)↵
Amy's▲current▲balance:▲20↵
[10]▲----------↵
[▲▲▲FAIL▲▲]▲Phineas▲failed▲to▲withdraw▲\$11231↵
Phineas's▲current▲balance:▲11240▲(Savings▲Account)↵
[▲▲▲FAIL▲▲]▲Phineas▲failed▲to▲transfer▲\$11231▲to▲Justin↵
Phineas's▲current▲balance:▲11240▲(Savings▲Account)↵
[▲SUCCESS▲]▲Phineas▲withdrew▲\$11230▲with▲\$10▲handling▲fee↵
Phineas's▲current▲balance:▲0▲(Savings▲Account)↵
[11]▲----------↵
Phineas's▲transaction▲record:▲↵
1▲[▲SUCCESS▲]▲Phineas▲saved▲\$10000↵
2▲Phineas▲accrued▲interest▲for▲10▲yr(s).↵

```
3▲[▲SUCCESS▲]▲Phineas▲withdrew▲\$450▲with▲\$10▲handling▲fee↵
4▲[▲SUCCESS▲]▲Phineas▲transfered▲\$10▲to▲Amy▲with▲\$10▲handling▲fee↵
5▲[▲▲▲FAIL▲▲]▲Phineas▲failed▲to▲withdraw▲\$11231↵
6▲[▲▲▲FAIL▲▲]▲Phineas▲failed▲to▲transfer▲\$11231▲to▲Justin↵
7▲[▲SUCCESS▲]▲Phineas▲withdrew▲\$11230▲with▲\$10▲handling▲fee↵
[3:23▲user@ws▲p3]
```



```
[2]▲------------------------↵
[▲SUCCESS▲]▲Amy▲saved▲\$4090↵
[▲▲▲FAIL▲▲]▲Amy▲failed▲to▲saved▲\$0↵
[▲▲▲FAIL▲▲]▲Amy▲failed▲to▲saved▲\$-216↵
```

上述文字紅色字體部分應為failed to save...上述執行結果中已更正相關內容。批改時，接受此錯誤輸出。



- 本題相關的程式碼路徑已註明於檔名右側，同學們可以透過路徑複製到自己的家目錄。
- 本題輸出請勿沿用上次作業的 ANSI 色碼（用於控制終端機文字樣式），否則視為錯誤
- 類別撰寫要求請參考上方表格
- 本題應繳交檔案如下：
  - main.cpp
  - bank.cpp
  - saving_account.cpp

p4 銀行帳戶類別實作 2（動態多型與覆寫）

本題併入下次作業 cpp.hw11本次作業此題無需繳交。

接續上題，現在要寫成動態多型的方式。

BankAccount 類別相關定義請參考下列的 bank.h 程式（請依據檔名旁的路徑至 ws 取得程式碼，請修改後繳交

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

class BankAccount
{
```

```
protected:
    string name;
    int balance;
    struct
    {
        string record;
        int count = 1;
    } transaction;

public:
    BankAccount(string name_str);
    int get_balance();
    void set_balance(int new_balance);
    string get_name();
    void write_record(string record);
    void show_record();
    void save(int amount);
    void show_balance();
    void withdraw(int amount);
    void transfer(int amount, BankAccount &recipient);
};
```

SavingAccount 類別相關定義請參考下列的 saving_account.h 程式（請依據檔名旁的路徑至 ws 取得程式碼，請依照需求修改（不修改亦可）但一定要繳交□□

```

#include "bank.h"

class SavingAccount : public BankAccount
{
private:
    double deposit_interest_rate = 0.016;
    int handling_fee = 10;

public:
    SavingAccount(string name_str);
    void calculate_interest(int year);
    void show_balance();
    void withdraw(int amount);
    void transfer(int amount, BankAccount &recipient);
};
```

請參考下列的 main.cpp 程式（請依據檔名旁的路徑至 ws 取得程式碼）：

```

#include "saving_account.h"
using namespace std;

int main()
```

```
{
    BankAccount * customer[3];
    BankAccount Justin("Justin");
    BankAccount Amy("Amy");
    SavingAccount Phineas("Phineas");
    customer[0] = &Justin;
    customer[1] = &Amy;
    customer[2] = &Phineas;

    cout << "[1] ------------------------" << endl;
    customer[0]->show_balance();
    customer[1]->show_balance();

    cout << "[2] ------------------------" << endl;
    customer[1]->save(4090);
    customer[1]->save(0);
    customer[1]->save(-216);

    cout << "[3] ------------------------" << endl;
    customer[1]->transfer(1080, Justin);
    customer[1]->transfer(4050, Justin);
    customer[1]->transfer(0, Justin);
    customer[1]->transfer(-512, Justin);

    cout << "[4] ------------------------" << endl;
    customer[1]->withdraw(12000);
    customer[1]->withdraw(3000);
    customer[1]->withdraw(0);

    cout << "[5] ------------------------" << endl;
    customer[0]->show_balance();
    customer[1]->show_balance();

    cout << "[6] ------------------------" << endl;
    customer[1]->show_record();

    cout << "[7] ------------------------" << endl;
    customer[2]->show_balance();
    customer[2]->save(10000);
    customer[2]->show_balance();
    Phineas.calculate_interest(10); // for 10 years
    customer[2]->show_balance();

    cout << "[8] ------------------------" << endl;
    customer[2]->withdraw(450);
    customer[2]->show_balance();

    cout << "[9] ------------------------" << endl;
    customer[2]->show_balance();
```

```
    customer[2]->transfer(10, Amy);
    customer[2]->show_balance();
    customer[1]->show_balance();

    cout << "[10] ------------------------" << endl;
    customer[2]->withdraw(11231);
    customer[2]->show_balance();
    customer[2]->transfer(11231, Justin);
    customer[2]->show_balance();
    customer[2]->withdraw(11230);
    customer[2]->show_balance();

    cout << "[11] ------------------------" << endl;
    customer[2]->show_record();

    return 0;
}
```

檔案名稱	**檔案用途與撰寫要求**	**-**
main.cpp	主程式	題目提供原始碼
bank.h	BankAccount 類別的定義	題目提供原始碼，修改後繳交
bank.cpp	BankAccount 類別的實作	需繳交，可繳交 p3 的同名檔案
saving_account.h	SavingAccount 類別的定義	題目提供原始碼，修改後繳交
saving_account.cpp	SavingAccount 類別的實作	需繳交，可繳交 p3 的同名檔案

請完成名為 **bank.cpp**、**bank.h**、**saving_account.h** 與 **saving_account.cpp** 的 C++ 語言程式，相關檔案撰寫要求已詳述於上方表格。其他在 p3 已經繳交的類別實作檔亦可於此題重用，但你仍然必須為此題再一次繳交，否則 0 分計。

註：本題輸出請勿沿用上次作業的 ANSI 色碼（用於控制終端機文字樣式），否則視為錯誤。

本題將使用以下的 Makefile 進行編譯：

```
all: main.cpp bank.o saving_account.o bank.h saving_account.h
    c++ main.cpp bank.o saving_account.o
saving_account.o: bank.h saving_account.cpp saving_account.h
    c++ saving_account.cpp -c
bank.o: bank.cpp bank.h
    c++ bank.cpp -c
clean:
    rm -f *.o *~ *.*~ a.out*
```

此題的執行結果可參考如下：

```
[3:23 user@ws p4]./a.out
[1] ------------------------
Justin's current balance: 0
```

Amy's▲current▲balance:▲0↵
[2]▲----------↵
[▲SUCCESS▲]▲Amy▲saved▲\$4090↵
[▲▲▲FAIL▲▲]▲Amy▲failed▲to▲save▲\$0↵
[▲▲▲FAIL▲▲]▲Amy▲failed▲to▲save▲\$-216↵
[3]▲----------↵
[▲SUCCESS▲]▲Amy▲transfered▲\$1080▲to▲Justin↵
[▲▲▲FAIL▲▲]▲Amy▲failed▲to▲transfer▲\$4050▲to▲Justin↵
[▲▲▲FAIL▲▲]▲Amy▲failed▲to▲transfer▲\$0▲to▲Justin↵
[▲▲▲FAIL▲▲]▲Amy▲failed▲to▲transfer▲\$-512▲to▲Justin↵
[4]▲----------↵
[▲▲▲FAIL▲▲]▲Amy▲failed▲to▲withdraw▲\$12000↵
[▲SUCCESS▲]▲Amy▲withdrew▲\$3000↵
[▲▲▲FAIL▲▲]▲Amy▲failed▲to▲withdraw▲\$0↵
[5]▲----------↵
Justin's▲current▲balance:▲1080↵
Amy's▲current▲balance:▲10↵
[6]▲----------↵
Amy's▲transaction▲record:▲↵
1▲[▲SUCCESS▲]▲Amy▲saved▲\$4090↵
2▲[▲▲▲FAIL▲▲]▲Amy▲failed▲to▲save▲\$0↵
3▲[▲▲▲FAIL▲▲]▲Amy▲failed▲to▲save▲\$-216↵
4▲[▲SUCCESS▲]▲Amy▲transfered▲\$1080▲to▲Justin↵
5▲[▲▲▲FAIL▲▲]▲Amy▲failed▲to▲transfer▲\$4050▲to▲Justin↵
6▲[▲▲▲FAIL▲▲]▲Amy▲failed▲to▲transfer▲\$0▲to▲Justin↵
7▲[▲▲▲FAIL▲▲]▲Amy▲failed▲to▲transfer▲\$-512▲to▲Justin↵
8▲[▲▲▲FAIL▲▲]▲Amy▲failed▲to▲withdraw▲\$12000↵
9▲[▲SUCCESS▲]▲Amy▲withdrew▲\$3000↵
10▲[▲▲▲FAIL▲▲]▲Amy▲failed▲to▲withdraw▲\$0↵
[7]▲----------↵
Phineas's▲current▲balance:▲0▲(Savings▲Account)↵
[▲SUCCESS▲]▲Phineas▲saved▲\$10000↵
Phineas's▲current▲balance:▲10000▲(Savings▲Account)↵
Phineas▲accrued▲interest▲for▲10▲yr(s).↵
Phineas's▲current▲balance:▲11720▲(Savings▲Account)↵
[8]▲----------↵
[▲SUCCESS▲]▲Phineas▲withdrew▲\$450▲with▲\$10▲handling▲fee↵
Phineas's▲current▲balance:▲11260▲(Savings▲Account)↵
[9]▲----------↵
Phineas's▲current▲balance:▲11260▲(Savings▲Account)↵
[▲SUCCESS▲]▲Phineas▲transfered▲\$10▲to▲Amy▲with▲\$10▲handling▲fee↵
Phineas's▲current▲balance:▲11240▲(Savings▲Account)↵
Amy's▲current▲balance:▲20↵
[10]▲----------↵
[▲▲▲FAIL▲▲]▲Phineas▲failed▲to▲withdraw▲\$11231↵
Phineas's▲current▲balance:▲11240▲(Savings▲Account)↵
[▲▲▲FAIL▲▲]▲Phineas▲failed▲to▲transfer▲\$11231▲to▲Justin↵
Phineas's▲current▲balance:▲11240▲(Savings▲Account)↵
[▲SUCCESS▲]▲Phineas▲withdrew▲\$11230▲with▲\$10▲handling▲fee↵

```
Phineas's▲current▲balance:▲0▲(Savings▲Account)↵
[11]▲----------↵
Phineas's▲transaction▲record:▲↵
1▲[▲SUCCESS▲]▲Phineas▲saved▲\$10000↵
2▲Phineas▲accrued▲interest▲for▲10▲yr(s).↵
3▲[▲SUCCESS▲]▲Phineas▲withdrew▲\$450▲with▲\$10▲handling▲fee↵
4▲[▲SUCCESS▲]▲Phineas▲transfered▲\$10▲to▲Amy▲with▲\$10▲handling▲fee↵
5▲[▲▲▲FAIL▲▲]▲Phineas▲failed▲to▲withdraw▲\$11231↵
6▲[▲▲▲FAIL▲▲]▲Phineas▲failed▲to▲transfer▲\$11231▲to▲Justin↵
7▲[▲SUCCESS▲]▲Phineas▲withdrew▲\$11230▲with▲\$10▲handling▲fee↵
[3:23▲user@ws▲p4]
```



- 本題相關的程式碼路徑已註明於檔名右側，同學們可以透過路徑複製到自己的家目錄。
- 本題輸出請勿沿用上次作業的 ANSI 色碼（用於控制終端機文字樣式），否則視為錯誤
- 類別撰寫要求請參考上方表格
- 本題應繳交檔案如下：
  - bank.cpp
  - bank.h
  - saving_account.h
  - saving_account.cpp

From:
https://junwu.nptu.edu.tw/dokuwiki/ - **Jun Wu的教學網頁**
國立屏東大學資訊工程學系
CSIE, NPTU
Total: 343751

Permanent link:
https://junwu.nptu.edu.tw/dokuwiki/doku.php?id=cpp:2026spring:hw10

Last update: **2026/06/04 01:54**