Basics: From C to C++

Computer Programming for Engineers (DSAF003-42)
Fall, 2021

Practice 7 : Inheritance

Instructor:

Youngjoong Ko (nlp.skku.edu)

Inheritance example

- **IS-A** relationship (student is a person)

```cpp
#include <iostream>
#include <string>
using namespace std;
class Person{
private:
    string name;
    int age;
public:
    Person(string myname,int myage):
    name(myname), age(myage){}
    void PrintPersonInfo(){
        cout << "Hi, I'm " << name << " and "
        << age << " years old." << endl;
    }
};
class Student{
private:
    string name;
    int age;
    string major;
public:
    Student(string myname,int myage,string mymajor):
    name(myname), age(myage), major(mymajor){}
    void PrintStudentInfo(){
        cout << "Hi, I'm " << name << " and "
        << age << " years old." << endl;
        cout << "And my major is " << major <<endl;
    }
};
int main(){
    Person A("Minsu",22);
    Student B("Taehun",26,"AI");

    A.PrintPersonInfo();
    cout << endl;
    B.PrintStudentInfo();

    return 0;
};
```

```
Hi, I'm Minsu and 22 years old.

Hi, I'm Taehun and 26 years old.
And my major is AI
```

```cpp
#include <iostream>
#include <string>
using namespace std;
class Person{
private:
    string name;
    int age;
public:
    Person(string myname,int myage):
    name(myname), age(myage){}
    void PrintPersonInfo(){
        cout << "Hi, I'm " << name << " and "
        << age << " years old." << endl;
    }
};
class Student : public Person{
private:
    string major;
public:
    Student(string myname,int myage,string mymajor):
    Person(myname,myage), major(mymajor){}
    void PrintStudentInfo(){
        PrintPersonInfo();
        cout << "And my major is " << major <<endl;
    }
};
int main(){
    Person A("Minsu",22);
    Student B("Taehun",26,"AI");

    A.PrintPersonInfo();
    cout << endl;
    B.PrintStudentInfo();

    return 0;
};
```

```
Hi, I'm Minsu and 22 years old.

Hi, I'm Taehun and 26 years old.
And my major is AI
```

Private of base class

- **Member function of derived class can not access private variables of base class directly**

```cpp
#include <iostream>
#include <string>
using namespace std;
class Person{
private:
    string name;
    int age;
public:
    Person(string myname,int myage):
    name(myname), age(myage){}
    void PrintInfo(){
        cout << "Hi, I'm " << name << " and "
        << age << " years old." << endl;
    }
    string getname() const {return name;}
    int getage() const {return age;}
    void changename(string newname){
        name = newname;
    }
    void changeage(int newage){
        age = newage;
    }
};
```

```cpp
class Student : public Person{
private:
    string major;
public:
    Student(string myname,int myage,string mymajor):
    Person(myname,myage), major(mymajor){}
    void PrintInfo(){
        cout << "Hi, I'm " << getname() << "(" << getage()
        << ")" << " and my major is " << major << endl;
    }
    string getmajor() const {return major;}
    void changemajor(string newmajor){
        major = newmajor;
    }
};
int main(){
    Student A("Taehun",26,"AI");

    A.PrintInfo();
    cout << endl;
    A.changename("minsu");
    A.changeage(20);
    A.changemajor("software");
    A.PrintInfo();

    return 0;
};
```

```
Hi, I'm Taehun(26) and my major is AI

Hi, I'm minsu(20) and my major is software
```

Constructor of derived class

```cpp
#include <iostream>
using namespace std;
class Base{
private: int baseNum;
public:
    Base():baseNum(1){
        cout<<"Base()"<<endl;
    }
    Base(int n):baseNum(n){
        cout<<"Base(int n)"<<endl;
    }
    void printNum(){
        cout<<"baseNum: "<<baseNum<<endl;
    }
};
class Derived : public Base{
private: int derivedNum;
public:
    Derived():derivedNum(10){
        cout<<"DerivedBase()"<<endl;
    }
    Derived(int n):derivedNum(n){
        cout<<"DerivedBase(int n)"<<endl;
    }
    Derived(int n1, int n2)
    :Base(n1),derivedNum(n2){
        cout<<"DerivedBase(int n1, int n2)"<<endl;
    }
    void printDerivedNum(){
        printNum();
        cout<<"derivedNum: "<<derivedNum<<endl;
    }
};
```

```cpp
int main(){
    cout<<"---Case 1---"<<endl;
    Derived dr1;
    dr1.printDerivedNum();

    cout<<"---Case 2---"<<endl;
    Derived dr2(20);
    dr2.printDerivedNum();

    cout<<"---Case 3---"<<endl;
    Derived dr3(2,30);
    dr3.printDerivedNum();

    return 0;
}
```

```
---Case 1---
Base()
DerivedBase()
baseNum: 1
derivedNum: 10
---Case 2---
Base()
DerivedBase(int n)
baseNum: 1
derivedNum: 20
---Case 3---
Base(int n)
DerivedBase(int n1, int n2)
baseNum: 2
derivedNum: 30
```

Exercise 1

- **Define the class BestStudent**
  - BestStudent is a derived class of Student

```cpp
#include <iostream>
#include <string>
using namespace std;
class Person{
private:
    string name;
    int age;
public:
    Person(string myname,int myage):
    name(myname), age(myage){}
    void PrintInfo(){
        cout << "Hi, I'm " << name << " and "
        << age << " years old." << endl;
    }
    string getname() const {return name;}
    int getage() const {return age;}
    void changename(string newname){
        name = newname;
    }
    void changeage(int newage){
        age = newage;
    }
};
```

```cpp
class Student : public Person{
private:
    string major;
public:
    Student(string myname,int myage,string mymajor):
    Person(myname,myage), major(mymajor){}
    void PrintInfo(){
        cout << "Hi, I'm " << getname() << "(" << getage()
        << ")" << " and my major is " << major << endl;
    }
    string getmajor() const {return major;}
    void changemajor(string newmajor){
        major = newmajor;
    }
};
int main(){
    BestStudent A("Taehun",26,"AI",95);

    A.PrintInfo();
    return 0;
};
```

```
Hi, I'm Taehun(26, AI) and my score is 95
```

Private, Protected, Public

- Private < protected < public (access level)
- Most use public inheritance

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

class Base{
private:
    int num1=1;
protected:
    int num2=2;
public:
    int num3=3;
    void ShowData(){
        cout << num1<< endl;
        cout << num2<< endl;
        cout << num3<< endl;
    }
};
```

```cpp
class public_Base : public Base{
public:
    void ShowBaseMember(){
        // cout << "num1: " <<num1<< endl;
        cout << "num2: " <<num2<< endl;
        cout << "num3: " <<num3<< endl;
    }
};
class protected_Base : protected Base{
public:
    void ShowBaseMember(){
        // cout << "num1: " <<num1<< endl;
        cout << "num2: " <<num2<< endl;
        cout << "num3: " <<num3<< endl;
    }
};
class private_Base : private Base{
public:
    void ShowBaseMember(){
        // cout << "num1: " <<num1<< endl;
        cout << "num2: " <<num2<< endl;
        cout << "num3: " <<num3<< endl;
    }
};
```

```cpp
int main(){
    public_Base A;
    A.ShowBaseMember();
    cout << "A's num3: " << A.num3 <<endl;

    protected_Base B;
    B.ShowBaseMember();
    // cout << "B's num3: " << B.num3 <<endl;

    private_Base C;
    C.ShowBaseMember();
    // cout << "C's num3: " << C.num3 <<endl;

    return 0;
}
```

```
num2: 2
num3: 3
A's num3: 3
num2: 2
num3: 3
num2: 2
num3: 3
```

Exercise 2

- **Define Player, MainPlayer, SubPlayer class**
  - MainPlayer and SubPlayer is a derived class of Player class and have showinfo() function
  - MainPlayer's salary is inputted value.
  - SubPlayer's salary is num of matches*pay per match (both are input)
  - SubPlayer class has addmatch() function that increase num of matches

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

class Player{
private:
    string name;
};
class MainPlayer : public Player{
private:
    int salary;
};
class SubPlayer : public Player{
private:
    int matches;
    int pay_per_match;
};
```

```cpp
int main(){
    MainPlayer a("태훈", 200);
    a.showinfo();

    SubPlayer b("충원",15,5);
    b.showinfo();
    b.addmatch(5);
    b.showinfo();

    return 0;
}
```

```
태훈's salary: 200
충원's salary: 75
충원's salary: 100
```

Assignment

- **Using Exercise2 define StarPlayer, PlayerList class**
  - StarPlayer is a derived class of MainPlayer
  - StarPlayer have showinfo(), addmatch() functions
  - StarPlayer's salary is inputted salary + num of matches*pay per match
- **PlayerList have addplayer(), showinfo(), showtotal() functions**
  - addplayer function add their player
  - showinfo function call each player's showinfo()
  - showtotal function print total salary of all players

```
int main(){
    PlayerList players;
    players.addplayer(new MainPlayer("태훈",200));
    players.addplayer(new MainPlayer("준희",300));

    SubPlayer* b = new SubPlayer("충원",15,5);
    b->addmatch(5);
    players.addplayer(b);

    StarPlayer* c = new StarPlayer("두영",500,30,10);
    c->addmatch(10);
    players.addplayer(c);

    players.showinfo();
    players.showtotal();

    return 0;
}
```

```
태훈's salary: 200
준희's salary: 300
충원's salary: 100
두영's salary: 900
Total Salary: 1500
```