Relationships among classes

Relationships among classes: *composition* and *inheritance*

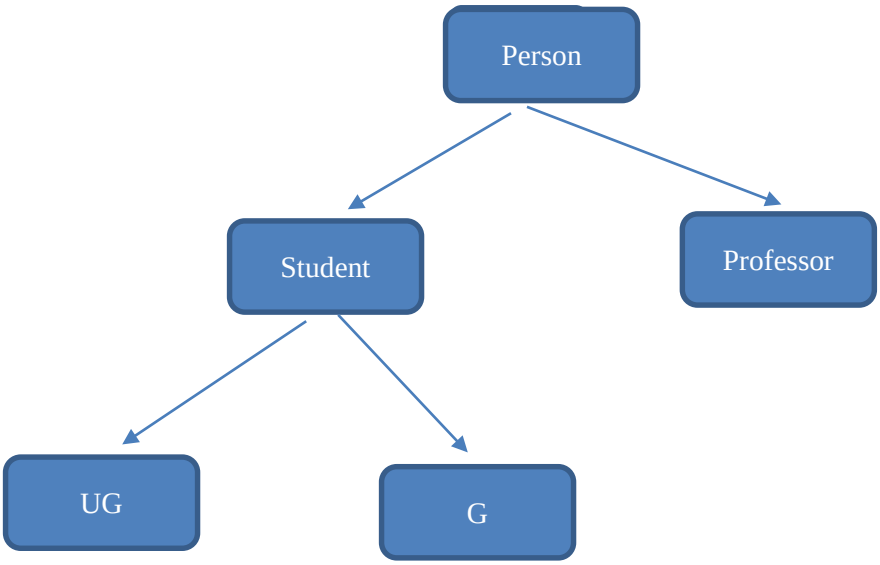
Composition follows a has-a relationship; inheritance an is-a relationship.

Inheritance allows us to define a class in terms of another class. [Software reuse]

For example, what do I know about a Student? A student is a type of Person. What about a Professor? Also a type of Student. The *inheritance hierarchy* or *class hierarchy* for this is below.

Superclass ≡ Base class

Subclass ≡ Derived class



Example1 - Composition (has-a)

Date class:

```
//date.h
//date class

//prevent multiple inclusions of header file
#ifndef DATE_H
#define DATE_H

#include <iostream>
#include <string>

class Date {
   friend istream& operator>> (istream&, Date&);
   friend ostream& operator<< (ostream&, const Date&);
public:
   Date (int=0, int=0, int=0);
   void setDate (int, int, int);
private:
   int month, day, year;
};

#endif


//date.cpp
//implementation of Date class
#include "Date.h"
#include <string>

Date::Date (int m, int d, int y) : month(m), day(d), year(y){
}

void Date::setDate (int m, int d, int y){
   month = m;
   day = d;
   year = y;
}

istream& operator>> (istream& in, Date& x){
   in >> x.month >> x.day >> x.year;
   return in;
}

ostream& operator<< (ostream& out, const Date& x){
   static string monthName[] = {"", "January", "February", "March",
      "April", "May", "June", "July", "August", "September", "October",
      "November", "December"};
   out << monthName[x.month] << ' ' << x.day << ", " << x.year;
   return out;
}
```

Recall that header files are included in each file in which the class is used. There is at lease one more source code (.cpp) file in which the program's "main" function is defined. So,

```
#ifndef TIME_H
#define TIME_H
…
#endif
```

prevents multiple declarations in the program. In a large file several source files may require the same header file declarations. Sometimes one header file must "include" other header files. If the header file was already previously included, the rest of the file (until #endif) is ignored.

Person class:

```
//person.h
//public interface for person class

#ifndef PERSON_H
#define PERSON_H

#include <string>
#include "Date.h"
using namespace std;

class Person {
public:
   Person (string, int);
   void setDOB (int m, int d, int y);
   void setDOD (int m, int d, int y);
   string getName();
   void getDOB();
   void getDOD();
   void printDOB();
   void printDOD();
private:
   string name;
   Date dob, dod;
   int sex;
};

#endif
```

```
//person.cpp
//implementation of person class

#include "person.h"

Person::Person (string n, int s): name(n), sex(s){
}

string Person::getName(){
   return name;
}

void Person::setDOB (int m, int d, int y){
   dob.setDate(m, d, y);
}

void Person::setDOD (int m, int d, int y){
   dod.setDate(m, d, y);
}

void Person::getDOB(){
// return dob;
}

void Person::getDOD(){
```

```
// return dod;
}

void Person::printDOB(){
   cout << dob;
}

void Person::printDOD(){
   cout << dod;
}
```

Main:

```
//composition.cpp
//This file contains main

#include <string>
#include "Date.h"
#include "Person.h"

int main(){
   Person author("Thomas Jefferson", 0);
   author.setDOB(4, 13, 1743);
   author.setDOD(7, 4, 1826);
   cout << "The author of the Declaration of Independence,  "
      << author.getName();
   cout << ",\n was born on ";
   author.printDOB();
   cout << " and died on ";
   author.printDOD();
   cout << ".\n";

   cout << "\n\n\nPress any key to close console window:  ";
   char c; cin >> c;
   return 0;
}
```

Output:

```
The author of the Declaration of Independence,  Thomas Jefferson,
 was born on April 13, 1743 and died on July 4, 1826.
```

Inheritance

To declare a derived class, e.g.:

```
class Box: public Rectangle
```

Why do we need the 'public' protection modifier? This allows instances of descendent classes to access public members of the parent and other ancestor classes. If we leave it out we deprive descendent objects from accessing members of the ancestor classes. (This would be the same, then, as using the 'private' modifier.)

Example - Inheritance (is-a)

The Student class:

```
//student.h
//student class

#include "Person.h"

class Student : public Person {
public:
   Student (string, int, string);
   void setDOM (int, int, int);
   void printDOM ();
private:
   string ID;
   Date dom;
   int credits;
   float gpa;
};
```

```
//student.cpp
//student class implementation

#include "Student.h"

Student::Student (string n, int s, string i) : Person(n, s), ID(i),
credits(0){
}

void Student::setDOM (int m, int d, int y) {
   dom.setDate(m, d, y);
}

void Student::printDOM (){
   cout << dom;
}
```

Main:

```
//inheritance.cpp
//this file contains main

#include <string>
#include "Date.h"
#include "Student.h"
using namespace std;

int main(){
   Student s("Bernard M. Baruch, IV", 0, "123456789");
   s.setDOB(5, 13, 1977);
   s.setDOM(8, 29, 1997);
   s.printName();
   cout << "\n Born:  "; s.printDOB();
   cout << "\n Matriculated:  "; s.printDOM();
```

```
   cout << endl;

   cout << "\n\n\nPress any key to close console window:  ";
   char c; cin >> c;
   return 0;
}
```

Output:

```
Bernard M. Baruch, IV
 Born:  May 13, 1977
 Matriculated:  August 29, 1997
```

Example 3: using protected access specifier

Date class - no change

Person class:

```
//person.h
//public interface for person class

#ifndef PERSON_H
#define PERSON_H

#include <string>
#include "Date.h"
using namespace std;

class Person {
public:
   Person (string, int);
   void setDOB (int m, int d, int y);
   void setDOD (int m, int d, int y);
   string getName();
   void getDOB();
   void getDOD();
   void printDOB();
   void printDOD();
   void printName();
protected:
   string name;
   Date dob, dod;
   int sex;
};

#endif
```

```
//person.cpp
//implementation of person class

#include "person.h"

Person::Person (string n, int s): name(n), sex(s){
}
```

```
string Person::getName(){
   return name;
}

void Person::printName(){
   cout << name;
}

void Person::setDOB (int m, int d, int y){
   dob.setDate(m, d, y);
}

void Person::setDOD (int m, int d, int y){
   dod.setDate(m, d, y);
}

void Person::getDOB(){
// return dob;
}

void Person::getDOD(){
// return dod;
}

void Person::printDOB(){
   cout << dob;
}

void Person::printDOD(){
   cout << dod;
}
```

Student class:

```
//student.h
//student class

#include "Person.h"

class Student : public Person {
public:
   Student (string, int, string);
   void setDOM (int, int, int);
   void printDOM ();
   void printSex();
protected:
   string ID;
   Date dom;
   int credits;
   float gpa;
};
```

```
//student.cpp
//student class implementation
```

```
#include "Student.h"

Student::Student (string n, int s, string i) : Person(n, s), ID(i),
credits(0){

}

void Student::setDOM (int m, int d, int y) {
   dom.setDate(m, d, y);
}

void Student::printDOM (){
   cout << dom;
}

void Student::printSex(){
   cout << (sex ? "female" : "male");
}
```

Main driver:

```
//inheritance2.cpp
//this file contains main

#include <string>
#include "Date.h"
#include "Student.h"
using namespace std;

int main(){
   Student s("Bernard M. Baruch, IV", 0, "123456789");
   s.setDOB(5, 13, 1977);
   s.setDOM(8, 29, 1997);
   s.printName();
   cout << "\n         Born:  "; s.printDOB();
   cout << "\n          Sex:  "; s.printSex();
   cout << "\n Matriculated:  "; s.printDOM();
   cout << endl;

   cout << "\n\n\nPress any key to close console window:  ";
   char c; cin >> c;
   return 0;
}
```

Output:

```
Bernard M. Baruch, IV
         Born:  May 13, 1977
          Sex:  male
 Matriculated:  August 29, 1997
```