객체지향프로그래밍과 자료구조 (실습)

Lab 5. (보충설명) Inheritance, Dynamic Array, Sorting

정보통신공학과

교수 김 영 탁

(Tel : +82-53-810-2497; Fax : +82-53-810-4742

http://antl.yu.ac.kr/; E-mail : ytkim@yu.ac.kr)

Outline

- Inheritance
- Dynamic Array of Class
- Algorithm on the dynamic array of class
  - sorting

상속 (inheritance) 개요

◆객체 지향형 프로그래밍 (Object-oriented programming)

- Powerful programming technique
- 추상화 개념과 함께 상속(***inheritance***) 기능을 제공
- 소프트웨어 재사용 (**software-reuse**) 과 다형성 (***polymorphism***) 기능을 제공

◆일반화 (generalized, generic) 클래스와 특화 (specialized)된 클래스

- 먼저 일반화된 클래스를 설계한 후, 이를 기반으로 필요에 따라 특화 (specialized)된 클래스를 추가 생성함으로써 소프트웨어 재사용
- 특화된 파생 클래스는 일반화된 기반 클래스의 속성을 상속 받은 후, 특화된 속성을 추가

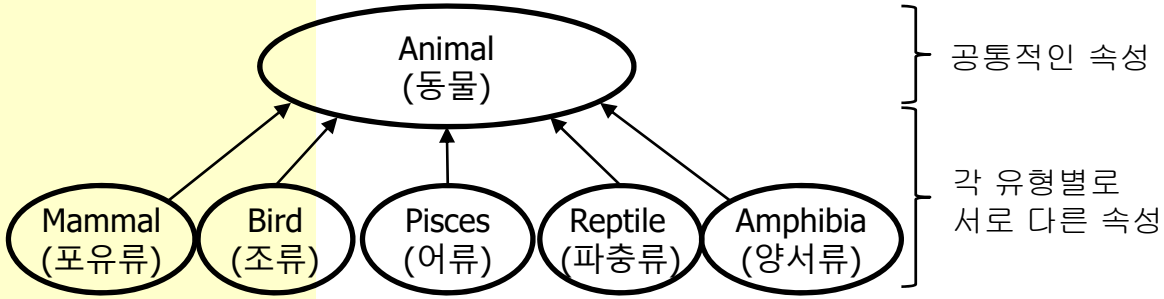
상속이란

◆ "상속(Inheritance)"

- 이미 개발되어 사용되는 클래스에 새로운 멤버 (속성)을 추가하여 새로운 클래스를 생성하게 함
- 새로운 클래스는 기존 클래스의 기능을 다시 구현할 수 있게 하거나, 확장할 수 있게 함
- 상속받은 클래스는 기존 클래스의 특별한 유형으로 볼 수 있으며, 상속받는 자식 클래스는 상속을 해 주는 부모클래스와 "is a"관계가 성립됨

Mammal (e.g., dog, cat) is an animal.
Bird is an animal.
Fish is an animal.

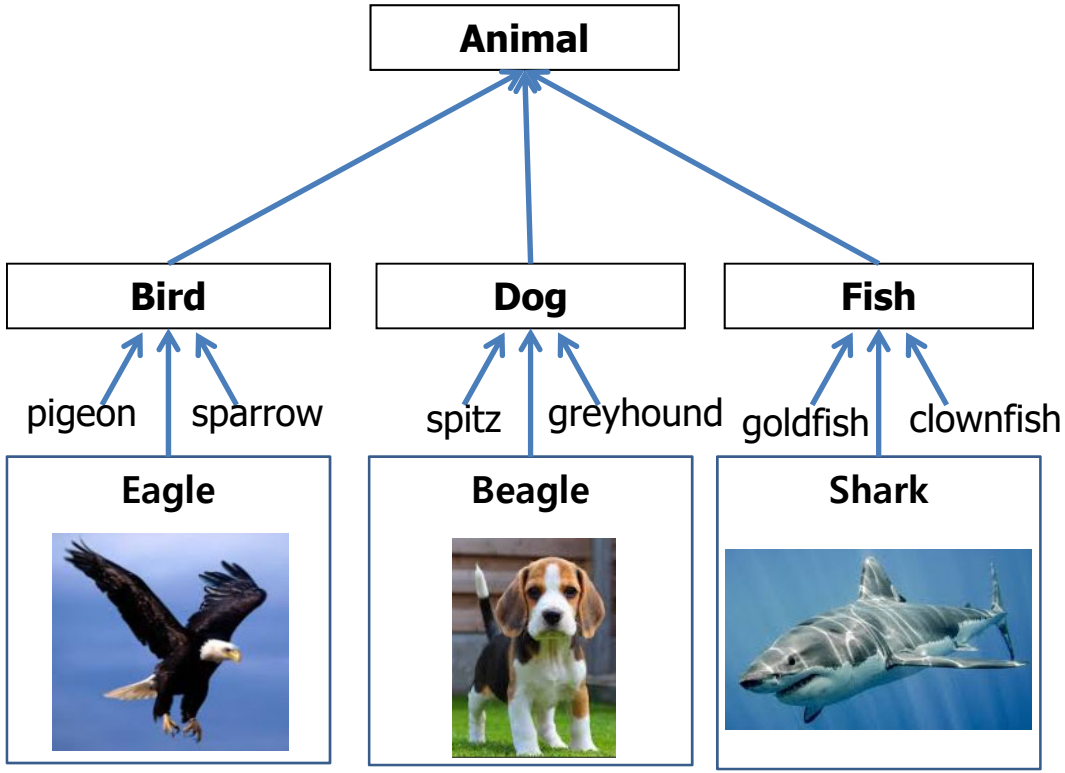
Mammal has 4 **legs**.
Bird has 2 **wings**.
Fish has **fins**.
But, Mammal, Bird and Fish have spine, eye, and blood *in common*.



상속 관계의 예

◆ class Animal

- class Animal <- class Dog, class Bird, class Fish
- class Bird <- class Eagle, class Pigeon, class Sparrow
- class Fish <- class Shark, class Goldfish, class Clownfish



상속 관련 용어

- **부모 클래스 (parent class), 기반클래스, 수퍼클래스**
  - 상속을 해 주는 기존 기반 클래스 (base class)
  - Super class라고도 함
- **자식 클래스 (child class), 파생클래스, 서브클래스**
  - 부모 클래스의 속성을 상속받고, 추가 속성을 정의하는 새로운 파생 클래스 (derived class)
  - Sub-class라고도 함
- **조상 클래스들 (ancestor classes)**
  - 부모 클래스, 부모의 부모인 조부모 및 상속 관계에 있는 직계 선대 클래스들
- **후손 클래스들 (descendant classes)**
  - 자식 클래스, 손주 클래스 및 상속 관계에 있는 직계 후대(후손) 클래스들

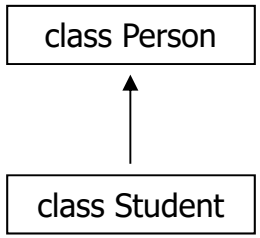
상속 구조의 클래스 예

- **기반 클래스 (base class)**
  - 공통적으로 사용될 수 있는 속성을 가지는 일반화된 클래스 (parent class, superclass)
  - (예) 학생과 직장인 등 모든 사람에게 공통적인 속성을 가지는 class Person
- **파생 클래스 (derived class)**
  - 기반 클래스의 속성을 상속받고, 특화된 속성을 추가한 클래스 (child class, subclass)
  - (예) class Person으로 부터 상속받는 class Student, class Staff

```
class Person // Existing base class
{
public:
   string getName() {return name;}
private:
   string name;
};

class Student : Person  // Derived class
{
public:
    int getSt_ID() {return student_id;}
private:
    int student_id;
};
```

클래스간의 상속을 나타내는 표시



"*Is a*" vs. "*Has a*" 관계

◆상속관계: "is a" 관계

- 자식 클래스는 부모 클래스와 "is a" 관계를 가짐
- 예)
  고양이는 동물이다.
  독수리는 동물이다.

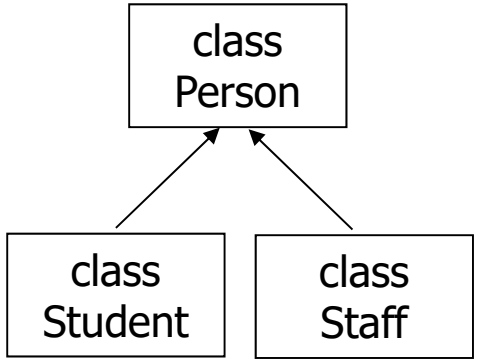
◆포함관계: "has a" 관계

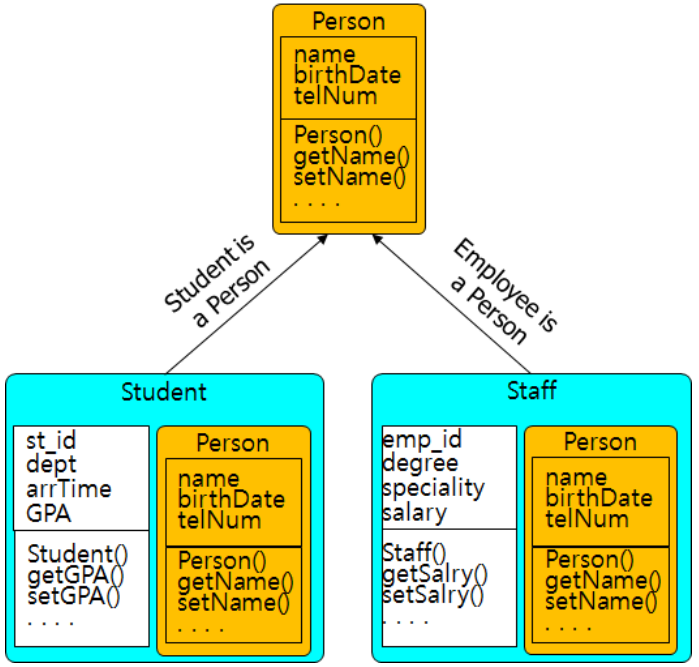
- 하나의 클래스가 다른 클래스를 속성 (데이터 멤버)으로 가지는 경우
- 예)
  동물은 눈(eye)과 척추(spine)을 가진다.
  class Person은 데이터 멤버로 string name을 가진다.

상속관계에서의 속성들의 포함관계

◆ 상속관계에서 자식클래스와 부모클래스의 속성 포함 관계

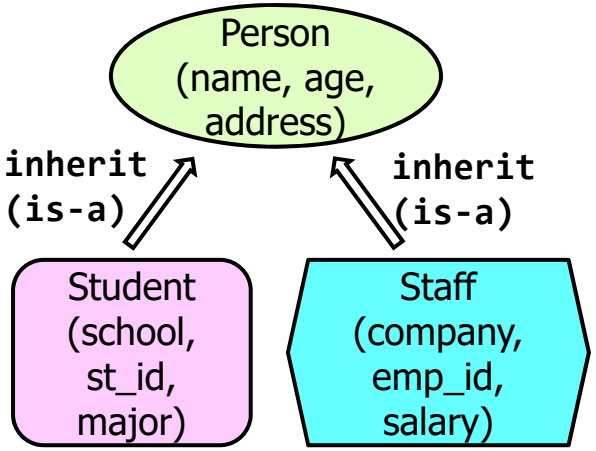
- 자식클래스는 부모클래스의 모든 속성을 모두 포함하게 됨
- 부모클래스는 보다 일반적인 (공통적으로 사용할 수 있는) 속성을 가짐
- 자식 클래스는 보다 일반적인 속성과 함께 특화된 속성을 추가로 가짐



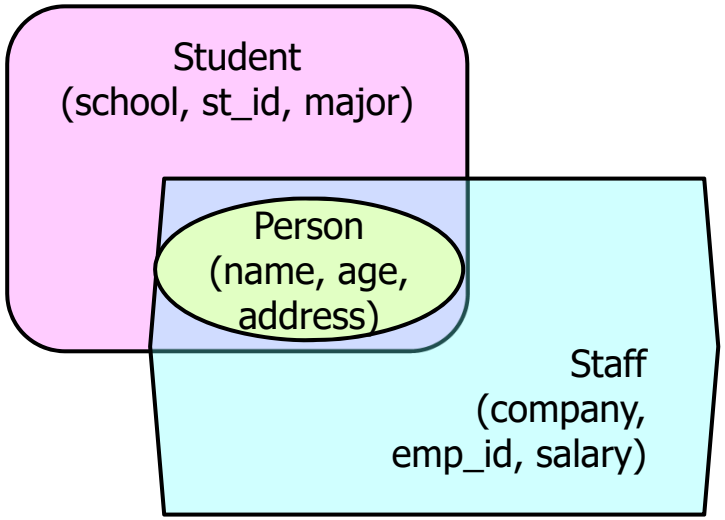


상속관계에서의 속성들의 포함관계

◆ 상속관계 및 부모 클래스의 속성 포함 관계



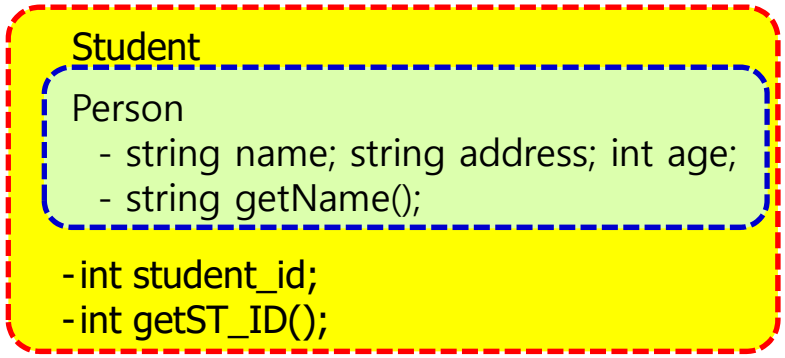
(Inheritance of class)



(Attributes in each class)

상속관계에서의 멤버들의 포함관계 (2)

```
class Person
{
public:
    string getName() {return name;}
private:
    string name;
    string address;
    int age;
};
class Student : Person
{
public:
    int getST_ID(){return student_id;}
private:
    int student_id;
};
```



- **Objects of Person (Parent) have members**

```
string name;
string address;
int age;
string getName()
  {return name;}
```

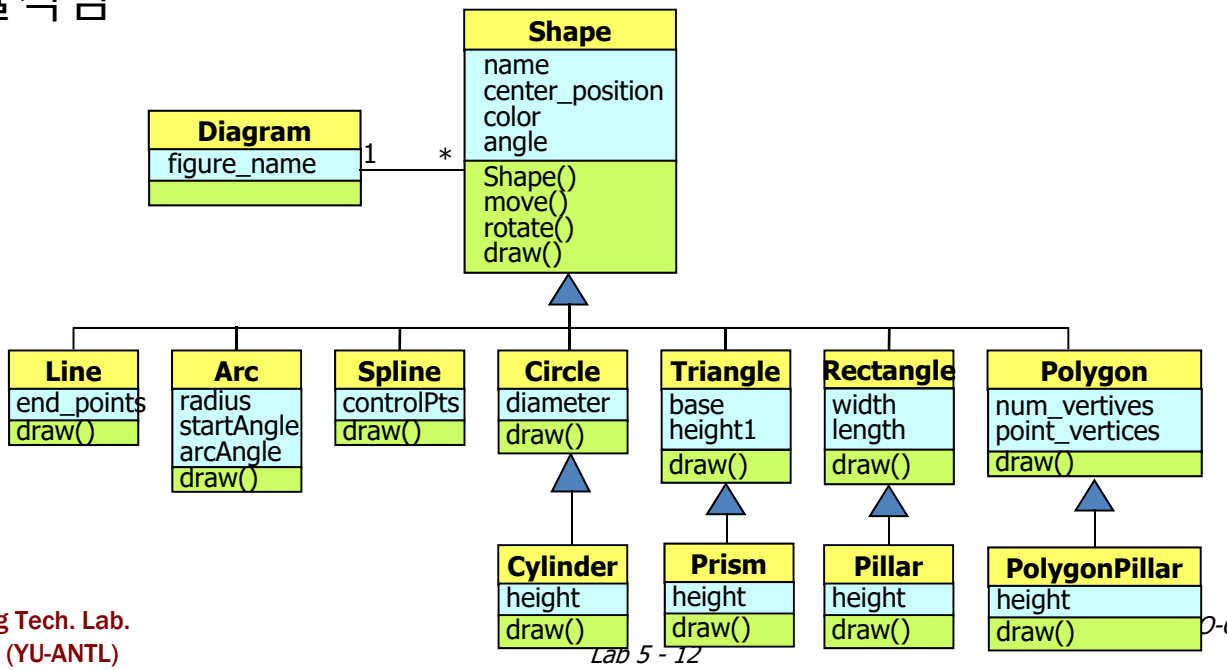
- **Objects of Student (Child) have members**

```
string name;
string address;
int age;
string getName()
  {return name;}
int student_id;
int getST_ID()
  {return student_id;}
```

상속과 소프트웨어 재사용

◆ 상속은 소프트웨어 재사용 기능 제공

- 상속 기반의 소프트웨어 개발에서는 자식 (파생) 클래스는 이미 기능과 성능이 잘 검증되고 신뢰성이 높은 기존의 부모 클래스를 기반으로 새로운 클래스를 생성하게 하며, 기존 부모 클래스의 소프트웨어를 재사용할 수 있게 함
- 다양하게 파생된 클래스 사용 중에 공통적인 속성의 수정 또는 보완이 필요한 경우, 파생클래스가 상속받은 부모 클래스를 수정 및 보완함으로써 쉽게 정비 (maintenance)할 수 있으며, 각 자식 클래스에서 공통적인 속성을 개별적으로 변경하는 것 보다 효율적임



상속을 기반으로 한 객체 지향형 프로그래밍의 예

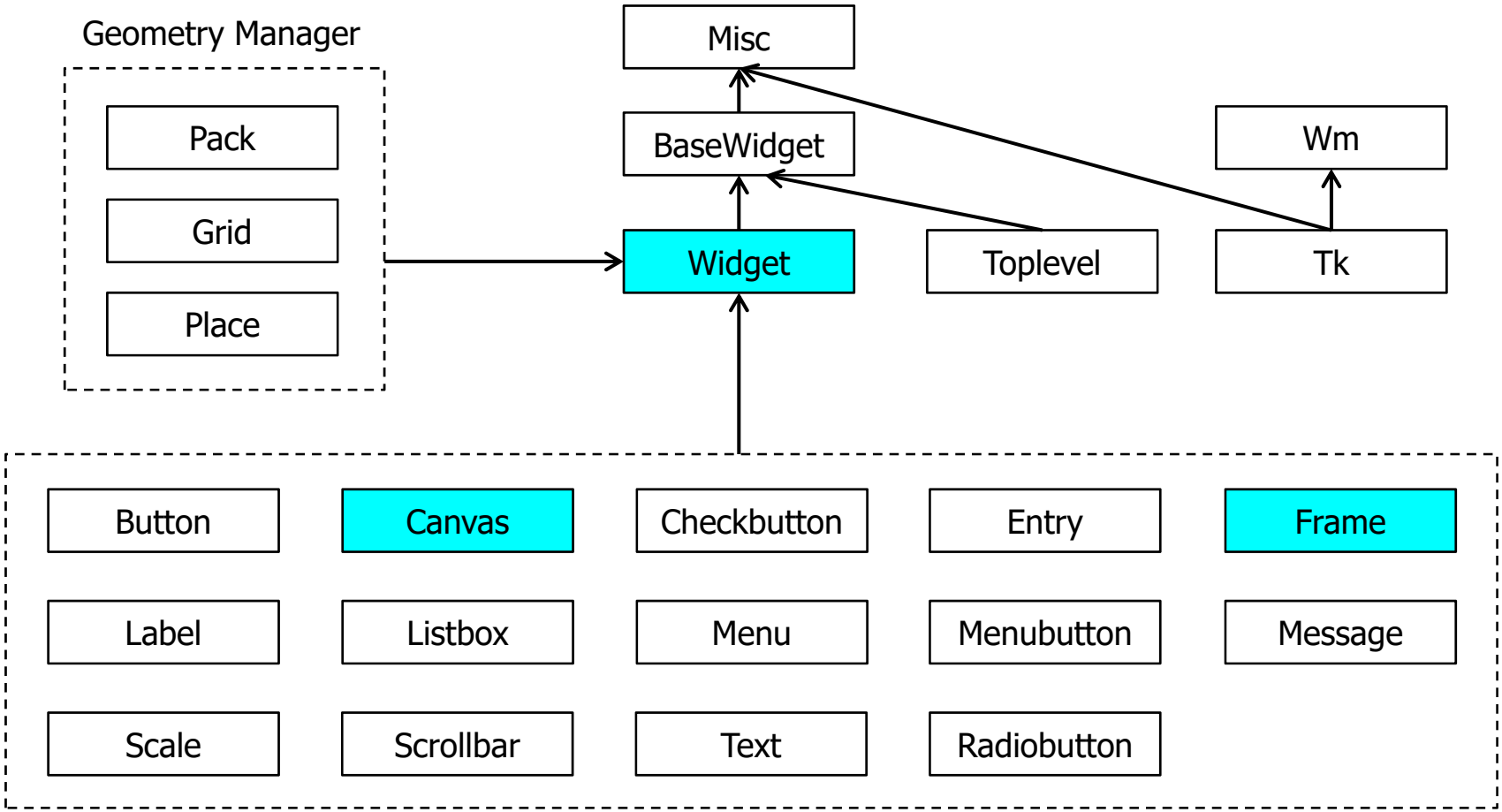
◆ 상속을 고려한 객체 지향형 클래스 설계

- 하나의 부모/조상 클래스는 다양한 자식/자손 클래스에서 재활용
- 부모/조상 클래스를 일반화 (generalized) 소프트웨어로 설계하고, 자식/후손 클래스를 특성화 (specialized) 소프트웨어로 설계

◆ 상속을 기반으로 한 객체 지향형 클래스의 사용 예

- Windows, Java, Python에서의 GUI (graphic user interface)의 클래스들은 모두 상속 개념의 객체 지향형 클래스
- Widget 클래스를 상속받아 Frame, Button, Label, Menu, Text 등의 파생 클래스 생성

class Widget



Base Class의 Private Data 접근

- **파생(자식)클래스는 기반(부모) 클래스의 속성을 모두 상속받음**
  - 기반(부모) 클래스의 private member도 모두 상속받으나, 부모클래스의 private member를 이름으로 직접 접근할 수는 없음
  - 부모클래스의 private member는 부모클래스의 public interface 멤버 함수를 통해서만 접근이 가능함
- **상속관계에 있는 파생(자식) 클래스 구현에서 기본(부모) 클래스의 속성을 간편하게 접근하는 방법?**
  - "protected" 접근 지정자를 사용하여, 상속관계에 있는 파생(자식) 클래스 구현에서 기본(부모) 클래스의 속성을 간편하게 접근할 수 있도록 제한적으로 허용

protected 접근 지정자

◆protected 접근 지정자

- 상속 관계에 있는 파생(자식) 클래스에서 기반(부모) 클래스의 속성을 이름으로 직접 접근할 수 있도록 제한적으로 허용
- 상속관계가 아닌 다른 클래스에서는 private 처럼 관리되며, 이름으로 직접 접근할 수 없도록 보호됨

◆protected 접근 지정자가 객체지향형 프로그래밍의 정보 보호 원칙을 위배?

- 파생(자식) 클래스에서 기반(부모) 클래스의 속성을 이름으로 직접 접근할 수 있도록 제한적으로 허용하는 것이 객체지향형 프로그래밍의 정보 보호 원칙을 위배한다는 의견도 있음
- 구현에서의 간편성/편리성을 제공하기 위하여 파생(자식) 클래스에게만 제한적으로 허용

파생클래스에서 기반 클래스 멤버함수의 재 지정 (redefinition)

◆ 파생클래스의 멤버함수

- 기반클래스의 멤버함수는 자동적으로 포함됨
- 특화된 멤버함수 추가
- 기반클래스의 멤버함수를 재지정할 수 있음

재지정된 멤버함수에서 기본 클래스의 멤버함수 호출

◆재지정된 멤버함수

- 기반클래스의 멤버함수를 파생클래스에서 재지정한 경우에도 기반클래스의 멤버함수 자체가 변경되지는 않음

◆상속관계에서 멤버함수가 재지정된 경우

```
Shape shp;  // base class
Triangle tri;  // class Triangle : class Shape
Prism prsm; // class Prism : class Triangle

shp.draw()  → calls Shape's draw()
tri.draw()  → calls Triangle's draw()
prsm.draw() → calls Prism's draw()
```

상속되지 않는 함수들

◆기반클래스의 멤버함수 중 아래의 특별한 경우를 제외한 멤버함수들을 상속됨

◆상속되지 않는 멤버함수들:

- 생성자 (constructors)
- 소멸자 (destructors)
- 복제 생성자 (copy constructor)
  - 복제 생성자가 구현되지 않으면 default 복제함수가 사용됨
  - 클래스에서 포인터와 동적메모리 생성 기능이 사용되면 복제 생성자를 구현하여야 함
- 대입 연산자 (assignment operator, '=')
  - 대입연산자가 구현되지 않으면 default 대입 연산 기능이 사용됨

상속된 클래스의 대입연산자 ("=") 와 복제 생성자

◆ 연산자 오버로딩으로 구현된 대입연산자와 복제 생성자 (copy constructor)

- 연산자 오버로딩으로 구현된 대입연산자와 복제 생성자는 상속되지 않음
- 기반 클래스의 대입연산자와 복제 생성자를 파생 클래스에서 사용할 수 있음
  - 파생클래스의 대입연산자 구현에서 기반클래스의 대입연산자를 사용
  - 파생클래스의 복제생성자 구현에서 기반클래스의 복제생성자를 사용

상속된 클래스의 대입연산자 ("=") 구현 예

◆기반 클래스의 대입연산자를 파생 클래스 대입 연산자 오버로딩 구현에서 사용하는 예

```
Derived& Derived::operator=(const Derived & rightSide)
{
    Base::operator=(rightSide);
    data = rightSide.data;
    ...
}
```

```
Shape& Shape::operator=(const Shape&  right)
{
    pos = right.pos; // position (x, y)
    angle = right.angle;
    name = right.name;
    color = right.color;

    return *this;
}
```

```
Triangle& Triangle::operator=(const Triangle& right)
{
    Shape::operator=(right);
    base = right.base;
    tri_height = right.tri_height;

    return *this;
}
```

상속된 클래스의 복제 생성자 (Copy Constructor)

◆복제 생성자 (Copy Constructor)

- 생성자 중에서 전달되는 인수가 그 클래스의 객체인 경우
- 전달된 객체을 복사하여 새로운 객체를 생성

◆상속된 파생 클래스의 복제 생성자에서 부모 클래스의 생성자를 호출

- 파생클래스 객체에는 기반 클래스의 데이터 멤버를 함께 포함하고 있음
- 파생클래스의 초기화 색션에서 복제 대상 객체를 기반 클래스의 생성자에 전달하여 복제를 수행

◆예)

```
Triangle::Triangle(const Triangle &tr)   // copy constructor
:Shape(tr), base(tr.base), tri_height(tr.tri_height)
{
    cout << "  Copy constructor (" << name << ").\n";
}
```

C++ class에서의 동적 배열의 생성 및 소멸

◆ ***new*** **classA [] 연산자를 사용하여 생성, delete [] 연산자를 사용하여 소멸**

- (예)

```
#define ArraySize 10
double * pDA;
pDA = new double[ArraySize];
. . .
delete [] pDA;
pDA = NULL;
```

- new 연산자를 사용하여 동적 배열을 생성할 때 대괄호(bracket)내에 동적 배열의 크기를 지정
- 동적 배열 생성 후, 그 주소를 반환하며, 동적 생성 배열 주소를 포인터에 대입하여 배열처럼 사용할 수 있음
- 동적 배열 사용이 종료된 후, delete 연산자 다음에 []를 표시하여 동적 배열이 소멸되도록 함
- 동적 배열 소멸뒤에도 포인터 pDA는 존재하며, 값을 NULL로 설정하여 소멸된 동적배열 사용이 실행되지 않도록 함

Class Date

```
class Date
{
        friend istream& operator>>(istream&, Date&);
        friend ostream& operator<<(ostream&, const Date&);
public:
        Date();  // default constructor
        Date(int y, int m, int d); // constructor
        void setDate(int newYear, int newMonth, int newDay);
        int getYear() { return year; }
        int getYearDay();
        int getWeekDay();
        int getElapsedDays(); // get elapsed days from AD 1. 1. 1.
        const Date operator=(const Date rightSide);
        bool operator>(Date rightSide);
        bool operator<(Date rightSide);
        bool operator==(Date rightSide);
        bool operator!=(Date rightSide);
        bool isLeapYear(int y); // check whether the given year y is leap year
private:
        bool isLeapYear(); // check whether the year is leap year
        bool isValidDate(int y, int m, int d);
        int year;
        int month;
        int day;
};

bool isLeapYear(int y); // used in genRandDate()
Date genRandDate();
```

MyString

◆ MyString

- used in the generations of random name and dept_name
- genRandName() 함수는 4 ~ 7 문자로 구성되는 학생 이름을 생성하며, 첫 번째 문자는 대문자로 설정
- genRandDeptName() 함수는 3 ~ 4 개의 대문자로 구성되는 학과 코드를 생성

◆ MyString.h

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

string genRandName();
string genRandDeptString();
```

Class Person

```
#include <iostream>
#include <string>
#include "Date.h"
class Person
{
        friend ostream& operator<< (ostream&, const Person&);
public:
        Person() { birthDate = Date(0, 0, 0); name = ""; };
        Person(string nm, Date bd) { birthDate = bd; name = nm; };
        void setName(string n) { name = n; }
        void setBirthDate(Date bd) { birthDate = bd; }
        string getName() const { return name; }
        Date getBirthDate() const { return birthDate; }
protected:
        Date birthDate;
        string name;
};
```

Class Student

```
class StudentArray;
class Student : public Person
{
        friend class StudentArray;
        friend ostream& operator<< (ostream&, const Student&);
public:
        Student();  // default constructor
        Student(int id);
        Student(int id, string n, Date dob, string dept_n, double gpa);
        int getST_id() const { return st_id; }
        string getDept_name() const { return dept_name; };
        double getGPA() const { return gpa; }
        Date getBirthDate() const { return birthDate; }
        void setST_id(int id) { st_id = id; }
        void setDept_name(string dp_n) { dept_name = dp_n; };
        void setGPA(double g) { gpa = g; }
        const Student& operator=(const Student& right);
        bool operator>(const Student& right);
        bool operator==(const Student& right);

private:
        int st_id;
        string dept_name;
        double gpa;
};

Student genRandStudent(int id);
void genST_ids(int num_students, int* st_ids);
```

Class StudentArray

```
#include <iostream>
#include "Student.h"

using namespace std;

class StudentArray
{
        friend ostream& operator<< (ostream&, const StudentArray&);
public:
        StudentArray(int size); // constructor
        StudentArray(const StudentArray& obj);
        ~StudentArray();
        int size() const { return num_students; }
        Student& operator[] (int index) const;
        void sortByBirthDate();
        void sortByName();
        void sortByST_ID();
        void sortByGPA();
private:
        Student* students;
        int num_students;
        bool isValidIndex(int index) const;
};
```

main()

```
/* main.cpp (Date, Person, Student, StudentArray) (1) */
#include <iostream>
#include <fstream>
#include "StudentArray.h"
#include <string>
#define NUM_STUDENTS 10

void main()
{
        StudentArray studentArray(NUM_STUDENTS);
        Student st;
        ofstream fout;
        int st_ids[NUM_STUDENTS];

        fout.open("output.txt");
        if (fout.fail())
        {
                cout << "Fail to open an output file (output.txt)\n";
                exit(1);
        }

        genST_ids(NUM_STUDENTS, st_ids);
        fout << "Initializing student array (num_students: " << NUM_STUDENTS << ")" << endl;
        for (int i = 0; i < NUM_STUDENTS; i++)
        {
                st = genRandStudent(st_ids[i]);
                studentArray[i] = st;
        }
        fout << studentArray;
```

```
/* main.cpp (Date, Person, Student, StudentArray) (2) */

        fout << "\nSorting studentArray by student id : " << endl;
        studentArray.sortByST_ID();
        fout << studentArray;

        fout << "\nSorting studentArray by student name : " << endl;
        studentArray.sortByName();
        fout << studentArray;

        fout << "\nSorting studentArray by GPA : " << endl;
        studentArray.sortByGPA();
        fout << studentArray;

        fout << "\nSorting studentArray by BirthDate : " << endl;
        studentArray.sortByBirthDate();
        fout << studentArray;

        fout << endl;
        fout.close();
}
```

◆ 실행결과

```
Initializing student array (num_students: 10)
StudentArray (size: 10)
Student[ st_id : 24604, name : Ozvsrtk, dept :  UXWF, birth date : (2141- 8-13) , GPA : 62.99]
Student[ st_id : 13902, name : Rvystmw, dept :  GGXR, birth date : (1288-11-11) , GPA : 28.59]
Student[ st_id : 10153, name :   Ikeff, dept :   CQP, birth date : (2548-10- 7) , GPA : 58.90]
Student[ st_id : 10292, name : Wsrenzk, dept :   KKA, birth date : (1118- 7-21) , GPA : 73.76]
Student[ st_id : 22382, name :   Sfadp, dept :  TLSG, birth date : (1924- 5- 9) , GPA : 99.30]
Student[ st_id : 27421, name :   Uvpva, dept :  ZBCO, birth date : (1031- 5- 9) , GPA : 32.90]
Student[ st_id : 28716, name : Bnpljvr, dept :  OEYL, birth date : (1209- 3-11) , GPA :  1.91]
Student[ st_id : 29718, name :    Qnqr, dept :   MYE, birth date : (2410-12- 4) , GPA : 64.13]
Student[ st_id : 29895, name :  Vaowux, dept :   JJL, birth date : (2348- 4-19) , GPA : 36.02]
Student[ st_id : 15447, name :    Sfzk, dept :  CBXC, birth date : (1893- 1- 9) , GPA : 79.38]

Sorting studentArray by student id :
StudentArray (size: 10)
Student[ st_id : 10153, name :   Ikeff, dept :   CQP, birth date : (2548-10- 7) , GPA : 58.90]
Student[ st_id : 10292, name : Wsrenzk, dept :   KKA, birth date : (1118- 7-21) , GPA : 73.76]
Student[ st_id : 13902, name : Rvystmw, dept :  GGXR, birth date : (1288-11-11) , GPA : 28.59]
Student[ st_id : 15447, name :    Sfzk, dept :  CBXC, birth date : (1893- 1- 9) , GPA : 79.38]
Student[ st_id : 22382, name :   Sfadp, dept :  TLSG, birth date : (1924- 5- 9) , GPA : 99.30]
Student[ st_id : 24604, name : Ozvsrtk, dept :  UXWF, birth date : (2141- 8-13) , GPA : 62.99]
Student[ st_id : 27421, name :   Uvpva, dept :  ZBCO, birth date : (1031- 5- 9) , GPA : 32.90]
Student[ st_id : 28716, name : Bnpljvr, dept :  OEYL, birth date : (1209- 3-11) , GPA :  1.91]
Student[ st_id : 29718, name :    Qnqr, dept :   MYE, birth date : (2410-12- 4) , GPA : 64.13]
Student[ st_id : 29895, name :  Vaowux, dept :   JJL, birth date : (2348- 4-19) , GPA : 36.02]

Sorting studentArray by student name :
StudentArray (size: 10)
Student[ st_id : 28716, name : Bnpljvr, dept :  OEYL, birth date : (1209- 3-11) , GPA :  1.91]
Student[ st_id : 10153, name :   Ikeff, dept :   CQP, birth date : (2548-10- 7) , GPA : 58.90]
Student[ st_id : 24604, name : Ozvsrtk, dept :  UXWF, birth date : (2141- 8-13) , GPA : 62.99]
Student[ st_id : 29718, name :    Qnqr, dept :   MYE, birth date : (2410-12- 4) , GPA : 64.13]
Student[ st_id : 13902, name : Rvystmw, dept :  GGXR, birth date : (1288-11-11) , GPA : 28.59]
Student[ st_id : 22382, name :   Sfadp, dept :  TLSG, birth date : (1924- 5- 9) , GPA : 99.30]
Student[ st_id : 15447, name :    Sfzk, dept :  CBXC, birth date : (1893- 1- 9) , GPA : 79.38]
Student[ st_id : 27421, name :   Uvpva, dept :  ZBCO, birth date : (1031- 5- 9) , GPA : 32.90]
Student[ st_id : 29895, name :  Vaowux, dept :   JJL, birth date : (2348- 4-19) , GPA : 36.02]
Student[ st_id : 10292, name : Wsrenzk, dept :   KKA, birth date : (1118- 7-21) , GPA : 73.76]
Sorting studentArray by GPA :
StudentArray (size: 10)
Student[ st_id : 22382, name :   Sfadp, dept :  TLSG, birth date : (1924- 5- 9) , GPA : 99.30]
Student[ st_id : 15447, name :    Sfzk, dept :  CBXC, birth date : (1893- 1- 9) , GPA : 79.38]
Student[ st_id : 10292, name : Wsrenzk, dept :   KKA, birth date : (1118- 7-21) , GPA : 73.76]
Student[ st_id : 29718, name :    Qnqr, dept :   MYE, birth date : (2410-12- 4) , GPA : 64.13]
Student[ st_id : 24604, name : Ozvsrtk, dept :  UXWF, birth date : (2141- 8-13) , GPA : 62.99]
Student[ st_id : 10153, name :   Ikeff, dept :   CQP, birth date : (2548-10- 7) , GPA : 58.90]
Student[ st_id : 29895, name :  Vaowux, dept :   JJL, birth date : (2348- 4-19) , GPA : 36.02]
Student[ st_id : 27421, name :   Uvpva, dept :  ZBCO, birth date : (1031- 5- 9) , GPA : 32.90]
Student[ st_id : 13902, name : Rvystmw, dept :  GGXR, birth date : (1288-11-11) , GPA : 28.59]
Student[ st_id : 28716, name : Bnpljvr, dept :  OEYL, birth date : (1209- 3-11) , GPA :  1.91]

Sorting studentArray by BirthDate :
StudentArray (size: 10)
Student[ st_id : 27421, name :   Uvpva, dept :  ZBCO, birth date : (1031- 5- 9) , GPA : 32.90]
Student[ st_id : 10292, name : Wsrenzk, dept :   KKA, birth date : (1118- 7-21) , GPA : 73.76]
Student[ st_id : 28716, name : Bnpljvr, dept :  OEYL, birth date : (1209- 3-11) , GPA :  1.91]
Student[ st_id : 13902, name : Rvystmw, dept :  GGXR, birth date : (1288-11-11) , GPA : 28.59]
Student[ st_id : 15447, name :    Sfzk, dept :  CBXC, birth date : (1893- 1- 9) , GPA : 79.38]
Student[ st_id : 22382, name :   Sfadp, dept :  TLSG, birth date : (1924- 5- 9) , GPA : 99.30]
Student[ st_id : 24604, name : Ozvsrtk, dept :  UXWF, birth date : (2141- 8-13) , GPA : 62.99]
Student[ st_id : 29895, name :  Vaowux, dept :   JJL, birth date : (2348- 4-19) , GPA : 36.02]
Student[ st_id : 29718, name :    Qnqr, dept :   MYE, birth date : (2410-12- 4) , GPA : 64.13]
Student[ st_id : 10153, name :   Ikeff, dept :   CQP, birth date : (2548-10- 7) , GPA : 58.90]
```

Oral Test 5

5.1 객체 지향형 프로그래밍에서 상속 (inheritance) 개념을 사용하는 장점은 무엇인가? 예를 들어 설명하라.

5.2 C++ 클래스에서 접근 지정자 "protected"와 "private"의 차이점에 대하여 설명하라.

5.3 상속받은 자식 클래스의 생성자 (constructor method of child class) 에서 상속 받는 부모 클래스의 생성자 (constructor method of parent class)를 호출하는 방법에 대하여 예를 들어 설명하라.

5.4 상속을 사용하는 C++ 클래스에서 상속이 되는 멤버함수와 상속이 되지 않는 멤버함수들을 구분하여 각각 예를 들어 설명하라.