MECHTRON 2MD3

Data Structures and Algorithms for Mechatronics

Winter 2022

05 C++ Class Relations

Department of Computing and Software

Instructor:

Omid Isfahanialamdari

January 24, 2022

Administration

- Midterm 1: at lecture time on Monday February 14, 2022
- It is advisable to start your assignment early

Friends of a Class

- In some cases, information-hiding is too prohibitive.
  - Only public members of a class are accessible by non-members of the class
- "friend" keyword
  - To give nonmembers of a class access to the nonpublic members of the class
- Friend
  - Functions
  - Classes – we skip this!
    - Poor class structure design

```
class Point{
    public:
        Point(double x1, double y1);
        friend ostream& operator<<(ostream &out, const Point &p1);
    private:
        double x, y;
};

Point::Point(double x1, double y1){
    x = x1;
    y = y1;
}
```

```
ostream& operator<<(ostream &out, const Point &p1){
    out << "P(" << p1.x << ", " << p1.y << ")";
    return out;
}
```

```
Point p(2.0, 4.0);
cout << p << endl;
```

Output:
P(2, 4)

Friends of a Class

- In some cases, information-hiding is too prohibitive.
  - Only public members of a class are accessible by non-members of the class
- “friend” keyword
  - To give nonmembers of a class access to the nonpublic members of the class
- Friend
  - Functions
  - Classes – we skip this!
    - Poor class structure design

```
void rectangle::setLT(point pt) {
  leftTop.set(pt.x, pt.y);
}
```

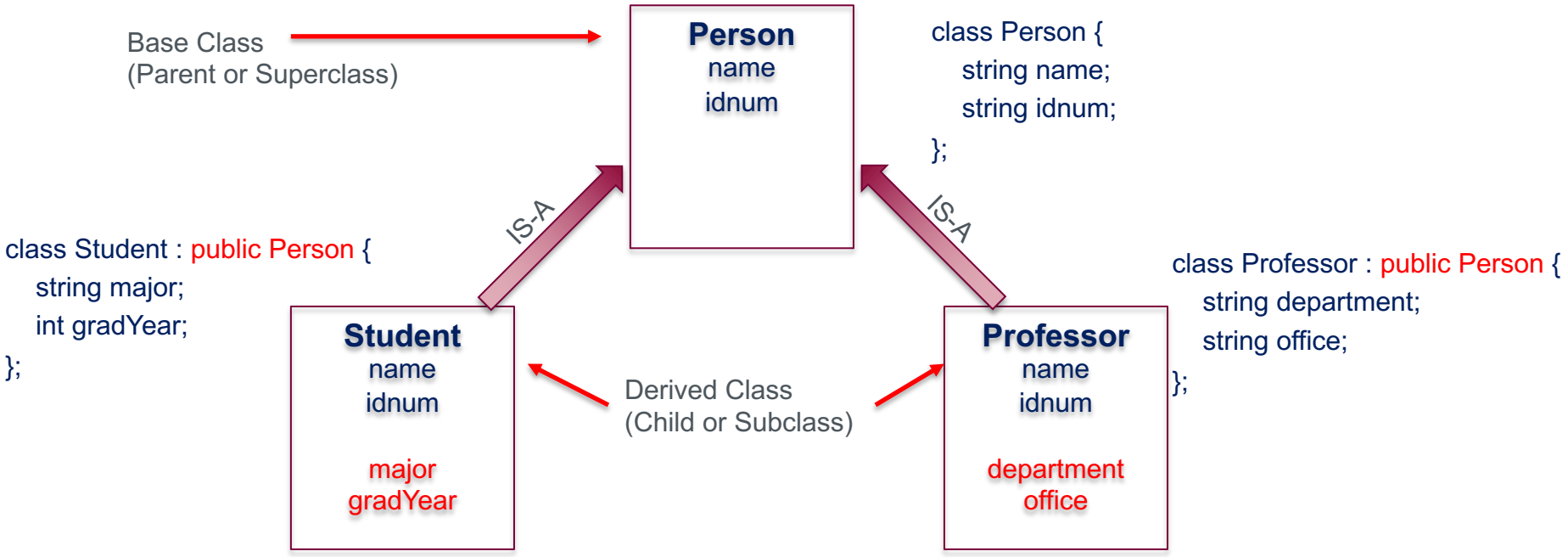
```
class Point{
    public:
        Point(double x1, double y1);
        void set(int a, int b);
    private:
        friend class rectangle;
        double x, y;
};

class rectangle {

    public:
        void setLT(point pt);
    private:
        Point leftTop, rightBottom;
};
```

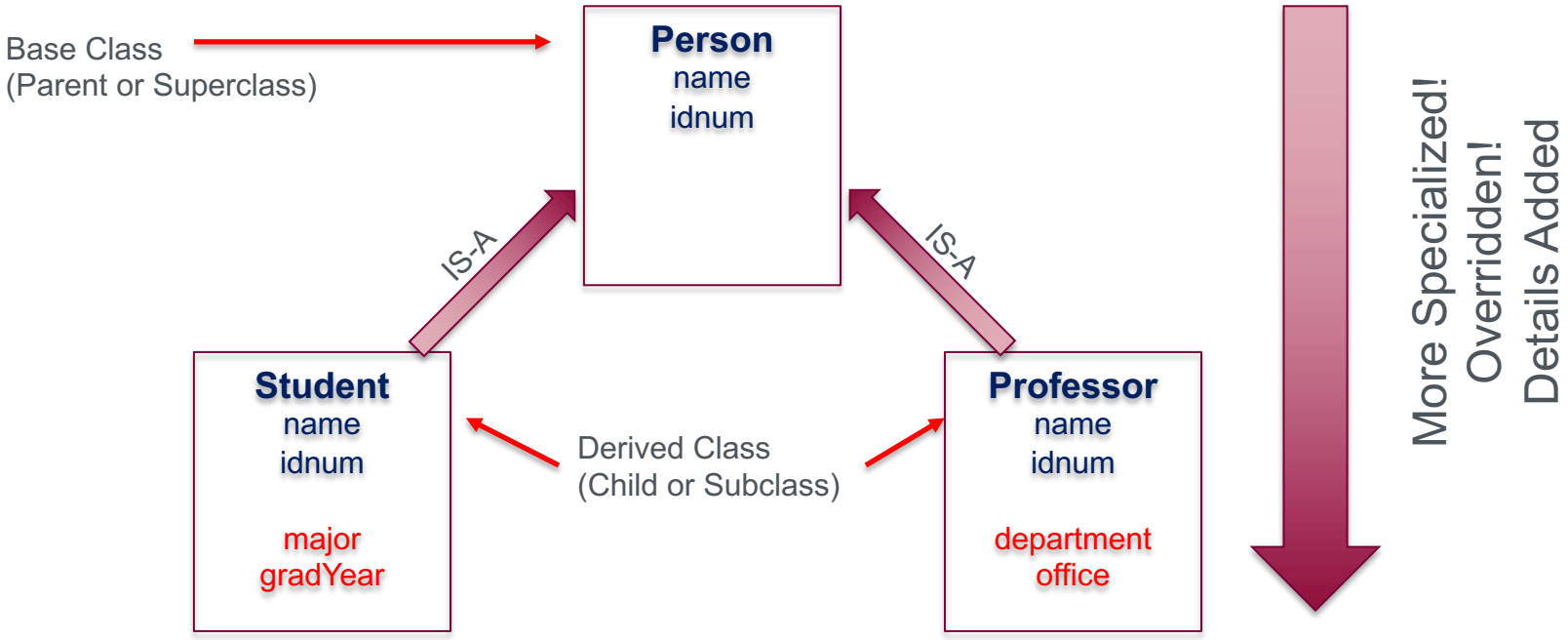
Inheritance

- Subclassing: define a class based on another class
  - Another class is parent class (or superclass)
  - New class is child class (subclass)
  - Hierarchical classification in a tree form
  - A way of "polymorphism" – we will discuss later!

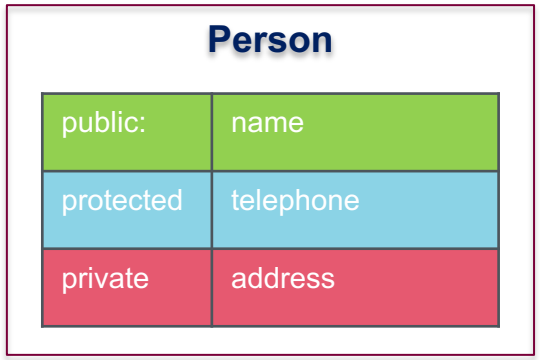


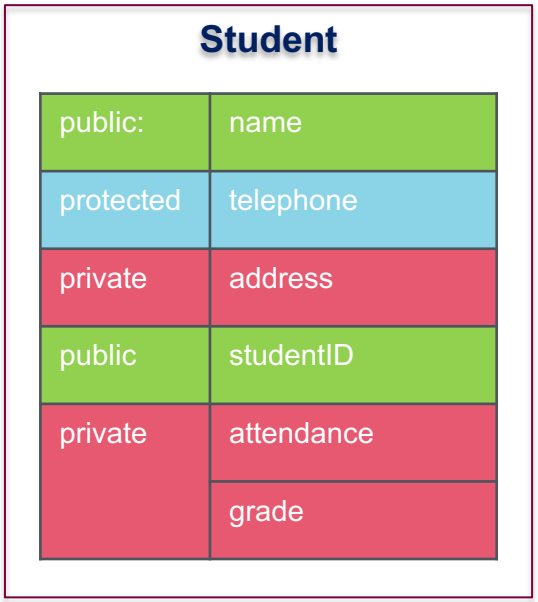
Inheritance

- Subclassing: define a class based on another class
  - Another class is parent class (or superclass)
  - New class is child class (subclass)
  - Hierarchical classification in a tree form
  - A way of "polymorphism" – we will discuss later!



Public Derivation

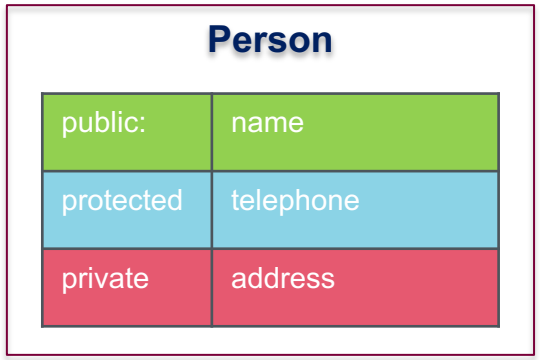


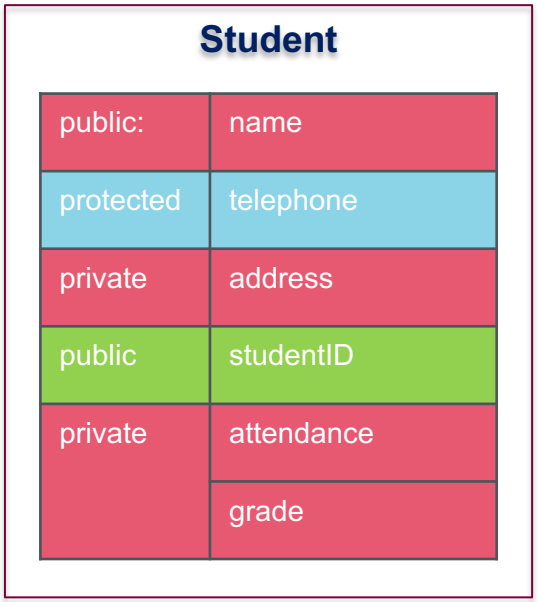



```
class Person {
  public:
    string name;
  protected:
    string telephone;
  private:
    string address;
};
```

```
class Student: public Person {
  public:
    int studentID;
  private:
    int attendance;
    double grade;
};
```

Private Derivation






```
class Person {
  public:
    string name;
  protected:
    string telephone;
  private:
    string address;
};
```

```
class Student: private Person {
  public:
    int studentID;
  private:
    int attendance;
    double grade;
};
```

Inheritance: A Mechanism for Reuse

```
class Person {  // Person (base class)
    private:
        string name;   // name
        string idNum;  // university ID number

    public:
        Person(const string& nm, const string& id);
        void print();        // print information
        string getName();  // retrieve name
};
```

```
class Student : public Person {  // Student (derived from Person)
    private:
        string major;  // major subject
        int gradYear;  // graduation year

    public:
        Student(const string& nm, const string& id, const string& maj, int year);
        void print();                                  // print information
        void changeMajor(const string& newMajor);  // change major

};
```

shared print()

Inheritance

```
class Person {  // Person (base class)
    private:
        string name;   // name
        string idNum;  // university ID number

    public:
        Person(const string& nm, const string& id);
        void print();      // print information
        string getName();  // retrieve name
};
```

```
Person::Person(const string& nm, const string& id)
    : name(nm), idNum(id)  // initialize name and id
    { }

void Person::print() { // definition of Person print
    cout << "Name " << name << ", " << "IDnum " << idNum << endl;
}

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

Inheritance

```
class Person {  // Person (base class)
    private:
        string name;   // name
        string idNum;  // university ID number

    public:
        Person(const string& nm, const string& id);
        void print();      // print information
        string getName();  // retrieve name
};
```

```
class Student : public Person {  // Student (derived from Person)
    private:
        string major;  // major subject
        int gradYear;  // graduation year

    public:
        Student(const string& nm, const string& id, const string& maj, int year);
        void print();                              // print information
        void changeMajor(const string& newMajor);  // change major

};
```

Inheritance

```
class Student : public Person {  // Student (derived from Person)
    private:
        string major;  // major subject
        int gradYear;  // graduation year

    public:
        Student(const string& nm, const string& id, const string& maj, int year);
        void print();                                  // print information
        void changeMajor(const string& newMajor);  // change major

};
```

Derived class must call base class's constructor

Base class's constructor must be in the initialization list!

```
Student::Student(const string& nm, const string& id, const string& maj, int year)
    : Person(nm, id), major(maj), gradYear(year)
    { }

void Student::print() {   // definition of Student print
    Person::print();      // first print Person information
    cout << "Major " << major << ", Year " << gradYear << endl;  // then student-specific info
}

void Student::changeMajor(const string& newMajor) {  // definition of Student print
    major = newMajor;
}
```

Inheritance

```
class Student : public Person {  // Student (derived from Person)
    private:
        string major;  // major subject
        int gradYear;  // graduation year

    public:
        Student(const string& nm, const string& id, const string& maj, int year);
        void print();                                  // print information
        void changeMajor(const string& newMajor);  // change major

};
```

Constructor order:
- base class => derived class

Destructor order:
- derived class => base class

```
Student::Student(const string& nm, const string& id, const string& maj, int year)
    : Person(nm, id), major(maj), gradYear(year)
    { }

void Student::print() {   // definition of Student print
    Person::print();      // first print Person information
    cout << "Major " << major << ", Year " << gradYear << endl;  // then student-specific info
}

void Student::changeMajor(const string& newMajor) {  // definition of Student print
    major = newMajor;
}
```

Inheritance

```
class Student : public Person {  // Student (derived from Person)
    private:
        string major;  // major subject
        int gradYear;  // graduation year

    public:
        Student(const string& nm, const string& id, const string& maj, int year);
        void print();                                  // print information
        void changeMajor(const string& newMajor);  // change major

};
```

calling parent's print()

```
Student::Student(const string& nm, const string& id, const string& maj, int year)
    : Person(nm, id), major(maj), gradYear(year)
    { }

void Student::print() {    // definition of Student print
    Person::print();       // first print Person information
    cout << "Major " << major << ", Year " << gradYear << endl;  // then student-specific info
}

void Student::changeMajor(const string& newMajor) {  // definition of Student print
    major = newMajor;
}
```

Inheritance

- Testing Inheritance

```
int main() {
    Person person("Mary", "12-345");  // declare a Person
    Student student("Bob", "98-764", "Math", 2012);  // declare a Student
    cout << student.getName() << endl;  // invokes Person::getName()
    person.print();   // invokes Person::print()
    student.Person::print();  // invokes studnet's parent's print() !!
    student.print();  // invokes Student::print()
    //person.changeMajor("Physics");   // ERROR!
    student.changeMajor("English");  // Okay
    student.print();

    return EXIT_SUCCESS;
}
```

```
Bob
Name Mary, IDnum 12-345
Name Bob, IDnum 98-764
Name Bob, IDnum 98-764
Major Math, Year 2012
Name Bob, IDnum 98-764
Major English, Year 2012
```

Questions?