CISC181 Introduction to Computer Science

Dr. McCoy

Lecture 27
December 8, 2009

Object Oriented Programming

- Classes categorize entities that occur in applications.
- Class teacher captures commonalities of teachers.
- See
- ~/Class/cisc181/examples/stu-teach.h
- ~/Class/cisc181/examples/stu-teach.cc
- ~/Class/cisc181/examples/stu-teach-drive.cc

What is a class?

- Class is similar to a category – the notion of categorization is common in conventional knowledge representation capturing is-a relationships between levels.

Fig. 9.3 Inheritance hierarchy for Shapes.

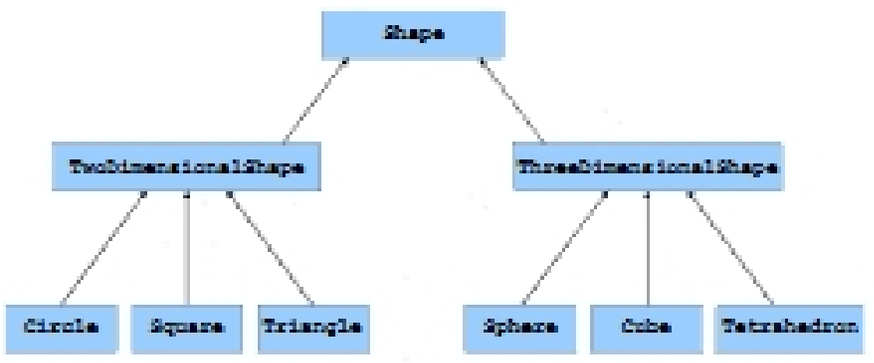
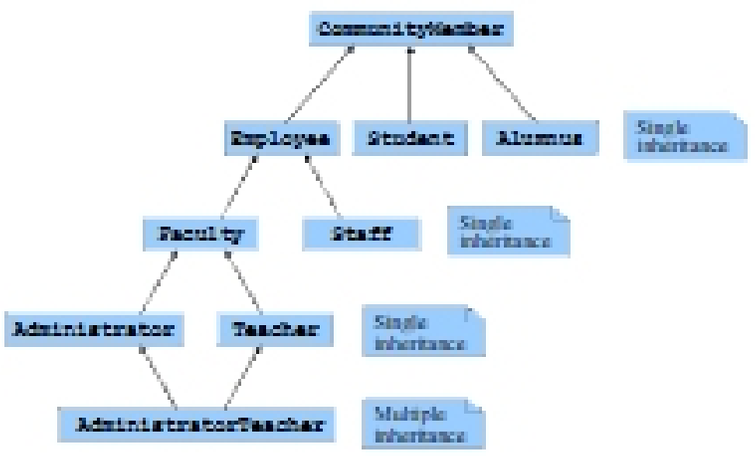


Fig. 9.2 Inheritance hierarchy for university CommunityMembers.



Extending a Hierarchy

- Once a hierarchy is established, it is easy to extend.
- To describe a new concept, not necessary to describe all of its features, it suffices to describe it differences from a concept existing in the hierarchy.
- Note: new cat, already know quite a bit!
- Same concept can be used with programming.

Inheritance Basics

- New class inherited from another class
- Base class
  - "General" class from which others derive
- Derived class
  - New class
  - Automatically has base class's:
    - Member variables
    - Member functions
  - Can then add additional member functions and variables

Commonality Abstracted

- What is common about student and teacher?
- Can we describe that, and then make a student and a teacher under that concept (inheriting some behavior from the "base" concept)

Example -- Person

- ~/Class/cisc181/examples/person-stu-teach.h
- ~/Class/cisc181/examples/person-stu-teach.cc
- ~/Class/cisc181/examples/person-stu-teach-drive.cc

9.1 Introduction

- **Inheritance**
  - Software reusability
  - Create new class from existing class
    - Absorb existing class's data and behaviors
    - Enhance with new capabilities
  - Derived class inherits from base class
    - Derived class
      - More specialized group of objects
      - Behaviors inherited from base class
        - Can customize
      - Additional behaviors

9.1 Introduction

- **Three types of inheritance**
  - **`public`**
    - Every object of derived class also object of base class
      - Base-class objects not objects of derived classes
      - Example: All cars vehicles, but not all vehicles cars
    - Can access non-**`private`** members of base class
      - Derived class can effect change to **`private`** base-class members
        - Through inherited non-**`private`** member functions
  - **`private`**
    - Alternative to composition
    - Chapter 17
  - **`protected`**
    - Rarely used

Derived Classes

- Consider example:
  Class of "Employees"
- Composed of:
  - Salaried employees
  - Hourly employees
- Each is "subset" of employees
  - Another might be those paid fixed wage each month or week

Derived Classes

- Don't "need" type of generic "employee"
  - Since no one's just an "employee"
- General concept of employee helpful!
  - All have names
  - All have social security numbers
  - Associated functions for these "basics" are same among all employees
- So "general" class can contain all these "things" about employees

Employee Class

- Many members of "employee" class apply to all types of employees
  - Accessor functions
  - Mutator functions
  - Most data items:
    - SSN
    - Name
    - Pay
- We won't have "objects" of this class, however

Employee Class

- Consider printCheck() function:
  - Will always be "redefined" in derived classes
  - So different employee types can have different checks
  - Makes no sense really for "undifferentiated" employee
  - So function printCheck() in Employee class says just that:
    - Error message stating "printCheck called for undifferentiated employee!! Aborting..."

Display 14.3 Interface for the Derived Class HourlyEmployee (1 of 2)

Display 14.3 Interface for the Derived Class HourlyEmployee

```
1
2  //This is the header file hourlyemployee.h.
3  //This is the interface for the class HourlyEmployee.
4  #ifndef HOURLYEMPLOYEE_H
5  #define HOURLYEMPLOYEE_H
6
6  #include <string>
7  #include "employee.h"
8
8  using std::string;
9
9  namespace SavitchEmployees
10 {
```


Display 14.3 Interface for the Derived Class HourlyEmployee (2 of 2)

```
11     class HourlyEmployee : public Employee
12     {
13     public:
14         HourlyEmployee( );
15         HourlyEmployee(string theName, string theSsn,
16                        double theWageRate, double theHours);
17         void setRate(double newWageRate);
18         double getRate( ) const;
19         void setHours(double hoursWorked);
20         double getHours( ) const;
21         void printCheck( );
22     private:
23         double wageRate;
24         double hours;
25     };

26 }//SavitchEmployees

27 #endif //HOURLYEMPLOYEE_H
```

You only list the declaration of an inherited member function if you want to change the definition of the function.

HourlyEmployee Class Interface

- Note definition begins same as any other
  - #ifndef structure
  - Includes required libraries
  - Also includes employee.h!
- And, the heading:
  class HourlyEmployee : public Employee
  { ...
  - Specifies "publicly inherited" from Employee class