Object-Oriented Programming
(in C++)

Course Introduction

Professor Yi-Ping You (游逸平)
Department of Computer Science
http://www.cs.nctu.edu.tw/~ypyou/

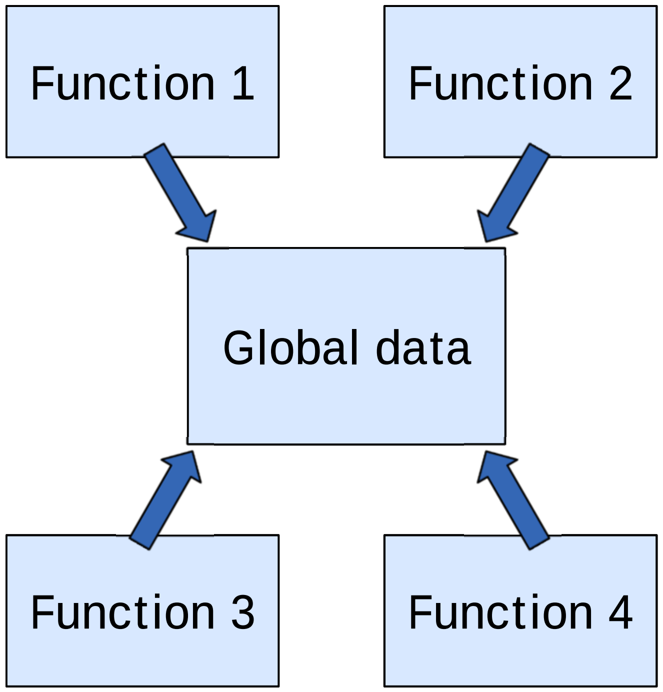
What is Object-Oriented Programming?

- A programming paradigm

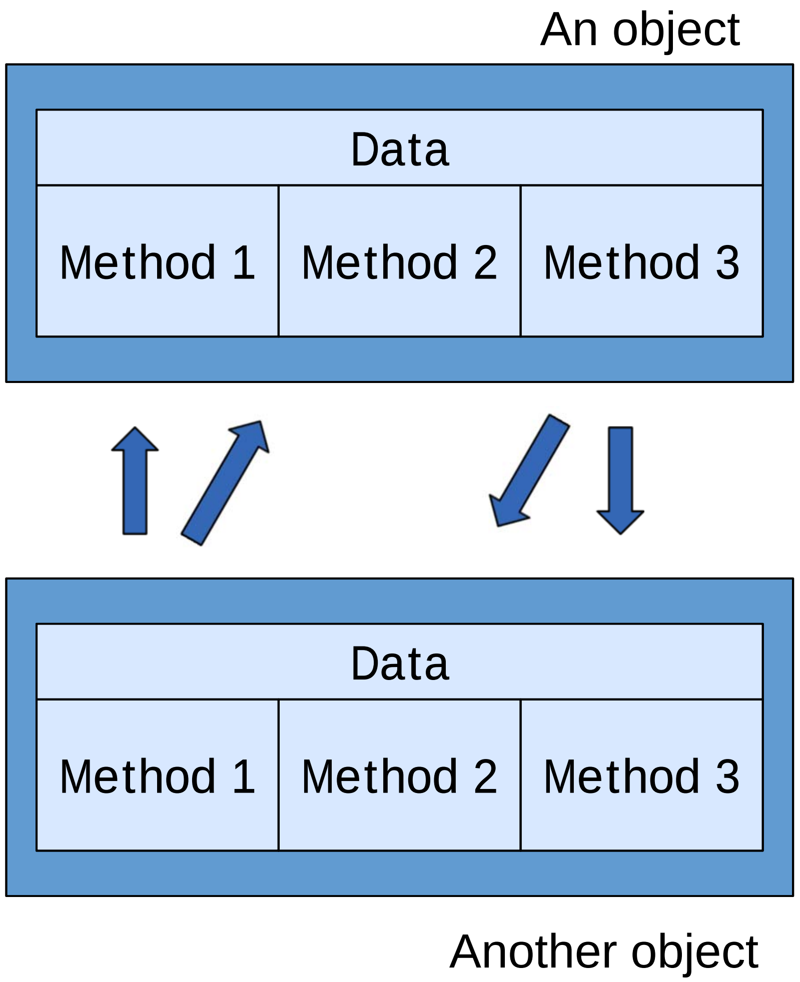
- Other programming paradigms
  - Procedural programming
    - C
  - Functional programming
    - Scheme
  - Logical programming
    - Prolog

Procedural vs Object-Oriented

- Procedural program
  - Passive data



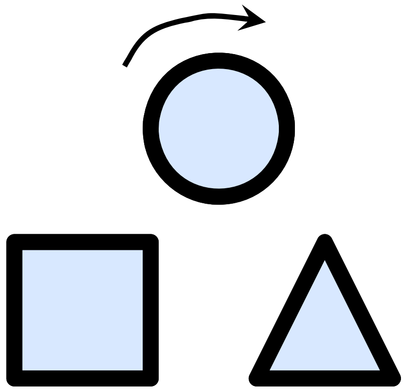
- Object-oriented program
  - Active data



Procedural vs Object-Oriented: Examples

- Given a specification:

There will be shapes on a GUI: a square, a circle, and a triangle. When the user clicks on a shape, the shape will rotate clockwise 360 degrees and play a MIDI sound file specific to that particular shape.



- Procedural solution?
- Object-oriented solution?

Procedural vs Object-Oriented: Examples

- Procedural

```
rotate(shapeNum) {
  //use shapeNum to look up
  //which shape to rotate
}
playSound(shapeNum) {
  //use shapeNum to look up
  //which MIDI to play
  //and play it
}
```

- Object-oriented

Square

```
rotate() {
  //rotate a square
}
playSo
  //pl
  //fo
}
```

Circle

```
rotate() {
  //rotate a circle
}
playS
  //p
  //f
}
```

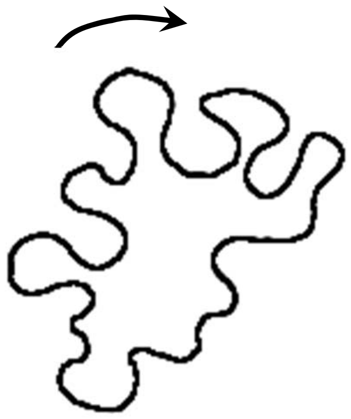
Triangle

```
rotate() {
  //rotate a triangle
}
playSound() {
  //play the MIDI
  //for the triangle
}
```

Procedural vs Object-Oriented: Examples

- Then comes a change to the specification:

There will be an amoeba shape on the screen, with the others. When the user clicks on the amoeba, it will rotate like the others, and play a MP3 sound file.



- Procedural solution?
- Object-oriented solution?

Procedural vs Object-Oriented: Examples

- Procedural
  - playSound() has to change

```
rotate(shapeNum) {
  //use shapeNum to look up
  //which shape to rotate
}
playSound(shapeNum) {
// if the shape is not
amoeba
  //use shapeNum to look up
  //...which MIDI to play
  //and play it
// else
  //play amoeba.mp3 sound
}
```

- Object-oriented
  - Class Amoeba is added

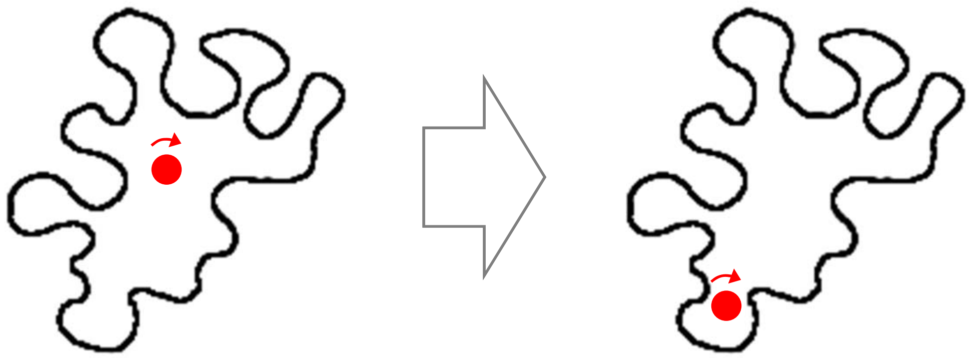
Amoeba

```
rotate() {
  //rotate an amoeba
}
playSound() {
  //play the MP3
  //for the amoeba
}
```

Procedural vs Object-Oriented: Examples

- Then comes another change to the specification:

Change the rotate point



- Procedural solution?
- Object-oriented solution?

Procedural vs Object-Oriented: Examples

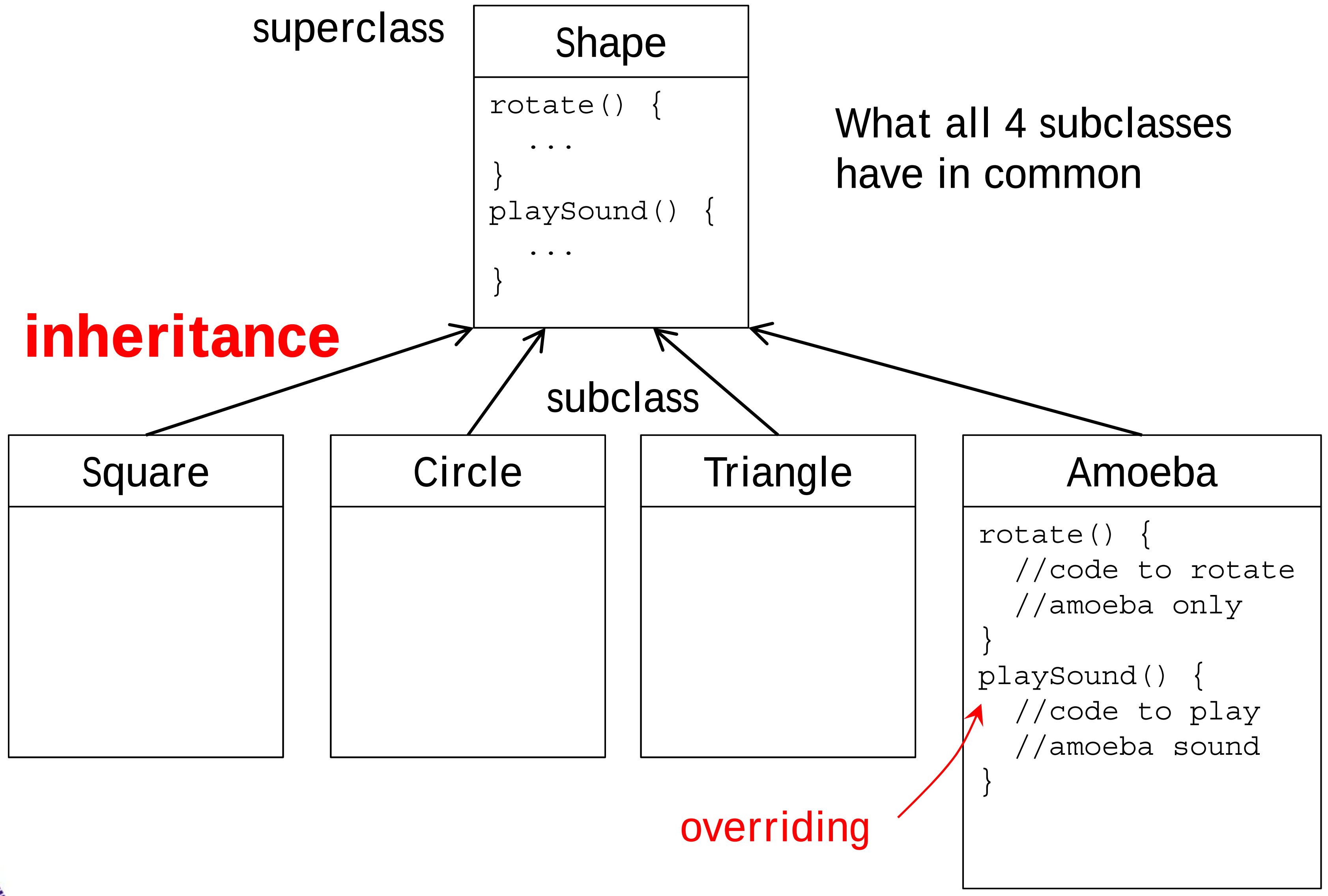
- Procedural
  - rotate() is modified
  - So is ALL the related code
- Object-oriented
  - Class Amoeba is changed
  - The rest is NOT affected

```
rotate(shapeNum,xPt,yPt) {
// if the shape is not amoeba
   //calculate center point
   //based on a rectangle
   //then rotate
// else
   //use xPt,yPt as
   //the rotation point offset
   //and then rotate
}
playSound(shapeNum) {
   ...
}
```

Amoeba

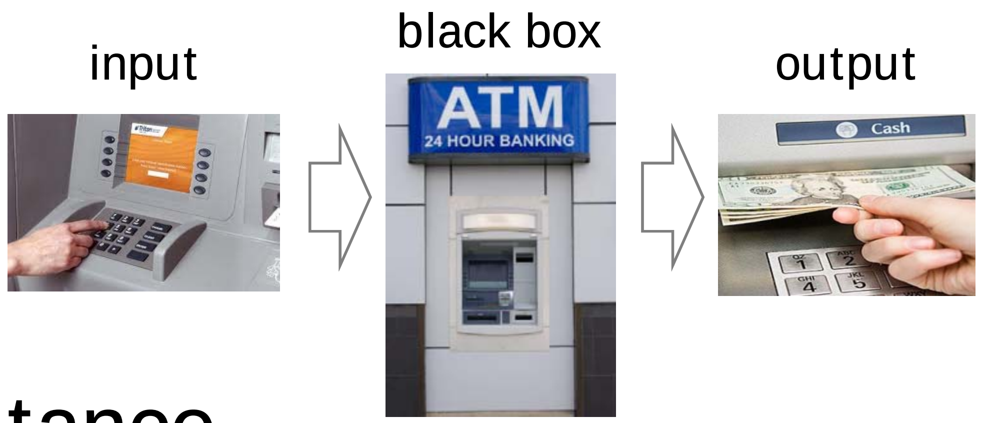
```
int xPoint
int yPoint
rotate() {
   //rotate an amoeba
   //using xPoint,yPoint
}
playSound() {
   //play the MP3
   //for the amoeba
}
```

OOP Solution



Important OO Concepts: P.I.E.

- Encapsulation (abstract data type)
  - “Black box” – information hiding



- Inheritance
  - Related classes share implementation and/or interface, allowing reuse of codes
- Polymorphism
  - Ability to use a class without knowing its type

Why C++?

- C++: extends C
  - Upwardly-compatible
- Popular and relevant (used in nearly every application domain):
  - End-user applications (Word, Excel, PowerPoint, Photoshop, Acrobat, Quicken, Google Chromium, Mozilla)
  - Operating systems (Windows 9x, NT, XP; IBM's K42; some Apple OS X)
  - Large-scale web servers/apps (Amazon, Google)
  - Central database control (Israel's census bureau; Amadeus; Morgan-Stanley financial modeling)
  - Communications (Alcatel; Nokia; 800 telephone numbers; major transmission nodes in Germany and France)
  - Numerical computation / graphics (Maya)
  - Device drivers under real-time constraints

Prerequisites

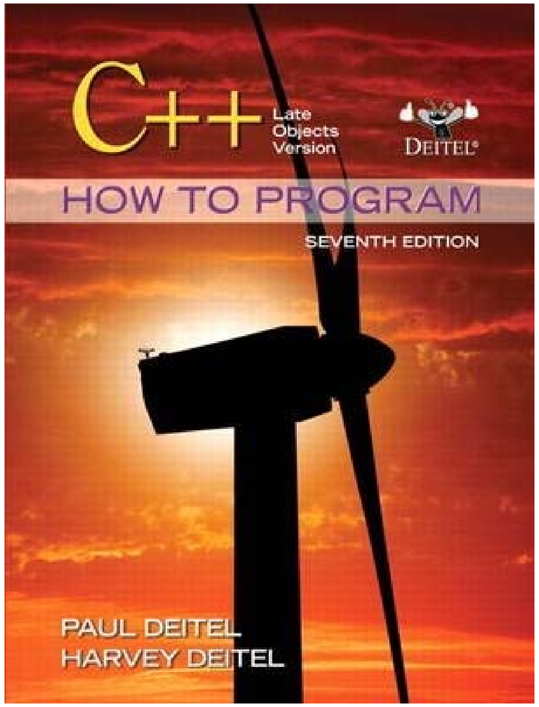
- Introduction to Computers and Programming (計算機概論與程式設計)
- C Programming experience

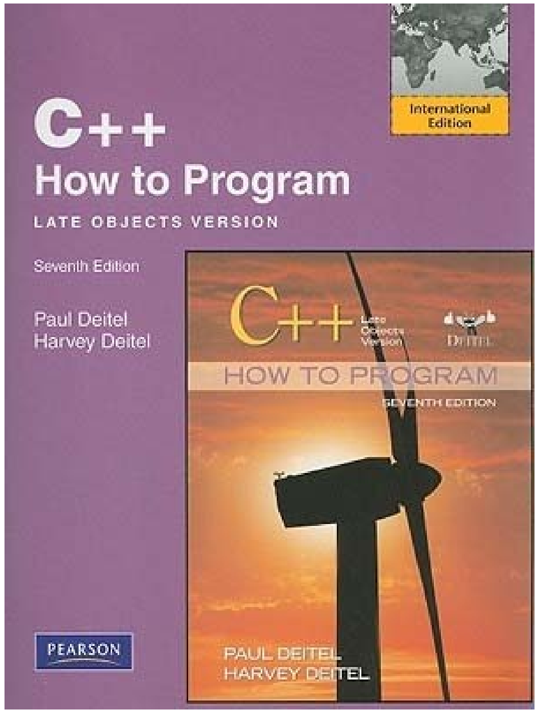
Course Objectives

- Understand the concept of object-oriented programming and be able to discuss the differences between procedural and object oriented languages
- Be able to program using important C++ techniques, such as composition of objects, operator overloads, dynamic memory allocation, inheritance and polymorphism, file I/O, exception handling, templates, preprocessor directives, and basic data structures.

Textbook

- Paul Deitel and Harvey Deitel, *C++ How to Program*, *7th edition (late objects version)*.





Course Topics (1/2)

- Procedural Programming
  - Introduction to C++ programming
  - C++ Functions
  - Arrays and Vectors
  - Pointers and References
- Object-Oriented Programming
  - Classes and Objects
  - Operator Overloading
  - Inheritance
  - Polymorphism

Course Topics (2/2)

- Other C++Topics
  - Stream I/O
  - Template and STL
  - Exception Handling
  - File Processing
  - String Processing

Tentative Schedule

Week	Subject	Readings
1	Course Introduction/Introduction to C++ Programming	Chapter 2
2	C++ Functions	Chapter 5
3	Peace Memorial Day/Arrays and Vectors	Chapter 6
4	Pointers and References	Chapter 7
5	Stream Input/Output	Chapter 15
6	Classes and Objects (I)	Chapter 9
7	Classes and Objects (II)	Chapter 10
8	Spring Break/Operator Overloading	Chapter 11
9	**Midterm Exam**	
10	Inheritance	Chapter 12
11	Polymorphism (I)	Chapter 13
12	Polymorphism (II)	Chapter 13
13	Template and Standard Template Library (I)	Chapters 14 and 21
14	Standard Template Library (II)	Chapter 21
15	Exception Handling	Chapter 16
16	File Processing	Chapters 8 and 17
17	String Processing	Chapter 18
18	**Final Exam**	

Administrative Stuff

- Course information
  - Credit: 3
  - Schedule: Mondays 13:20-15:10, Thursdays 9:00-9:50
  - Place: EC114
- Lab hours
  - Schedule: Thursdays 18:30-21:20
  - Place: EC316
- Course website
  - http://www.cs.nctu.edu.tw/~ypyou/courses/OOP-s16
- Course forum
  - http://sslab.cs.nctu.edu.tw/forum/
  - Registration required!

Grading

- Grades will be assigned based on
  - 7 Homework assignments and 7 quizzes (70%)
    - Each homework assignment contributes 8%
    - Each quiz contributes 2%
  - Midterm paper exams (10%)
  - Final online exams (20%)
  - Class participation and online discussion (bonus)

Lab Hours Schedule (4IJK)

Week	**Homework**	**Quiz**
3	HW1 released	Quiz 1
4	HW1 demo	
5	HW2 released	Quiz 2
6	HW2 demo	
7	HW3 released	Quiz 3
8	HW3 demo	
9	**(Midterm Exam)**	
10	HW4 released	Quiz 4
11	HW4 demo	
12	HW5 released	Quiz 5
13	HW5 demo	
14	HW6 released	Quiz 6
15	HW6 demo	
16	HW7 released	Quiz 7
17	**(Final Exam)**	

Class Participation

- Attendance – You should be here
- In-class and online participation!
- Opportunities for participation
  - Solving class problems
  - Feedback to me about the class

Late Assignment & Honesty Policy

- Slackers beware!
  - The penalty for late homework is **15% per day**
  - Late homework will not be accepted after 3 days past the original due date
- NO PLAGIARISM!
  - Homework assignments must be individual work
  - While you are allowed (and encouraged) to work together in understanding the concepts of the course and even the assigned problems, the solutions that you hand in should be entirely your own
  - Sharing of algorithms or code is NOT ALLOWED

Additional Rules

1. Source codes must be uploaded before demo
2. Submissions after the deadline get 10% penalty if a demo is made during the regular demo hours
3. Redemo during the regular demo hours is allowed for only once (a penalty of 50% is applied to each redemo item)
4. A no-demo during the regular demo hours is considered a late homework (15% penalty per day, zero credit after 3 days)

Office Hours and TAs

- Contacting me
  - ypyou@cs.nctu.edu.tw
  - Office: EC708
  - Office hours: by appointment
  - Can also catch me right after class
- TAs
  - You are highly recommended to send your mail to OOP-s16@sslab.cs.nctu.edu.tw. All TAs will receive the mail.
    - 邱明聰- mtchiu@sslab.cs.nctu.edu.tw
    - 林天心- thlin@sslab.cs.nctu.edu.tw
    - TBA