Object-Oriented Programming

物件導向程式設計

Instructor: 劉興民

e-Mail: damon@computer.org

Office: 工學院A館403室

Lectures: Tue, Thu 13:15-14:30

工學院A館001教室

Webpage: http://www.cs.ccu.edu.tw/~damon/html/oop.html

- **Course Repository** 課業倉儲

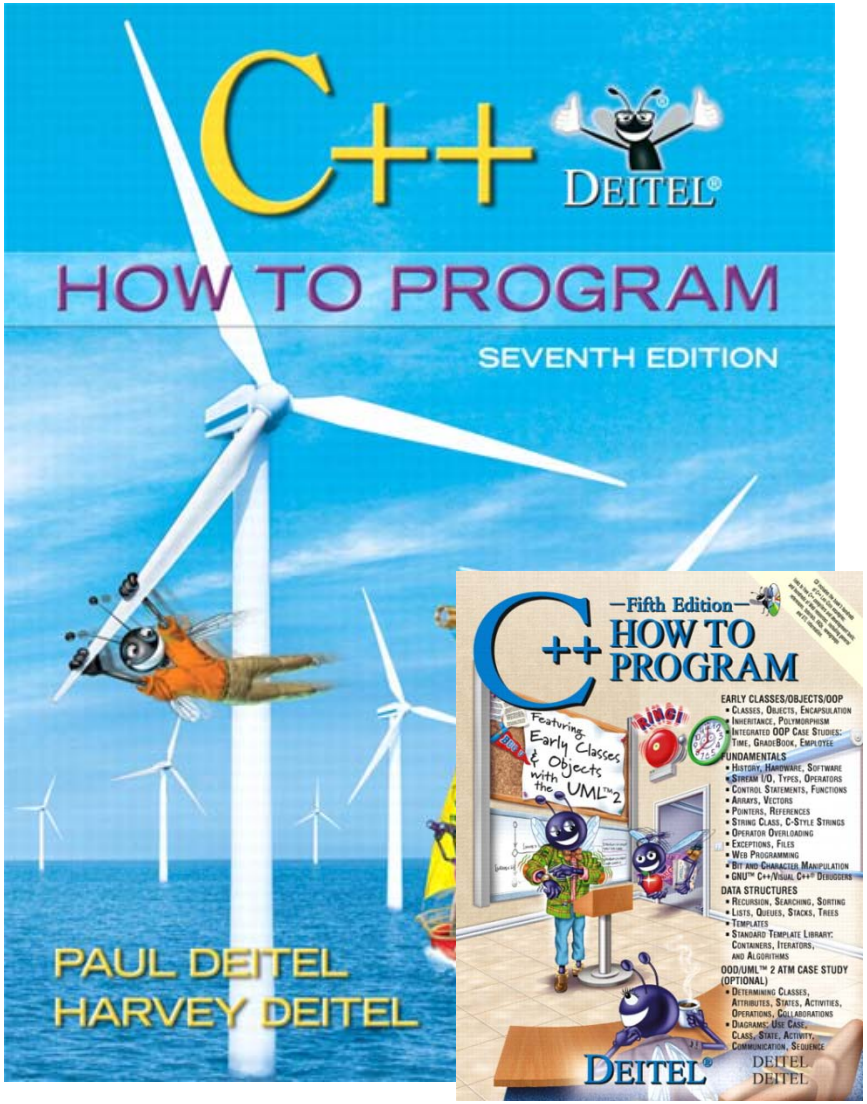
http://www.cs.ccu.edu.tw/~damon/secure/course-wk.html

- **Discussion Board** 課程討論區

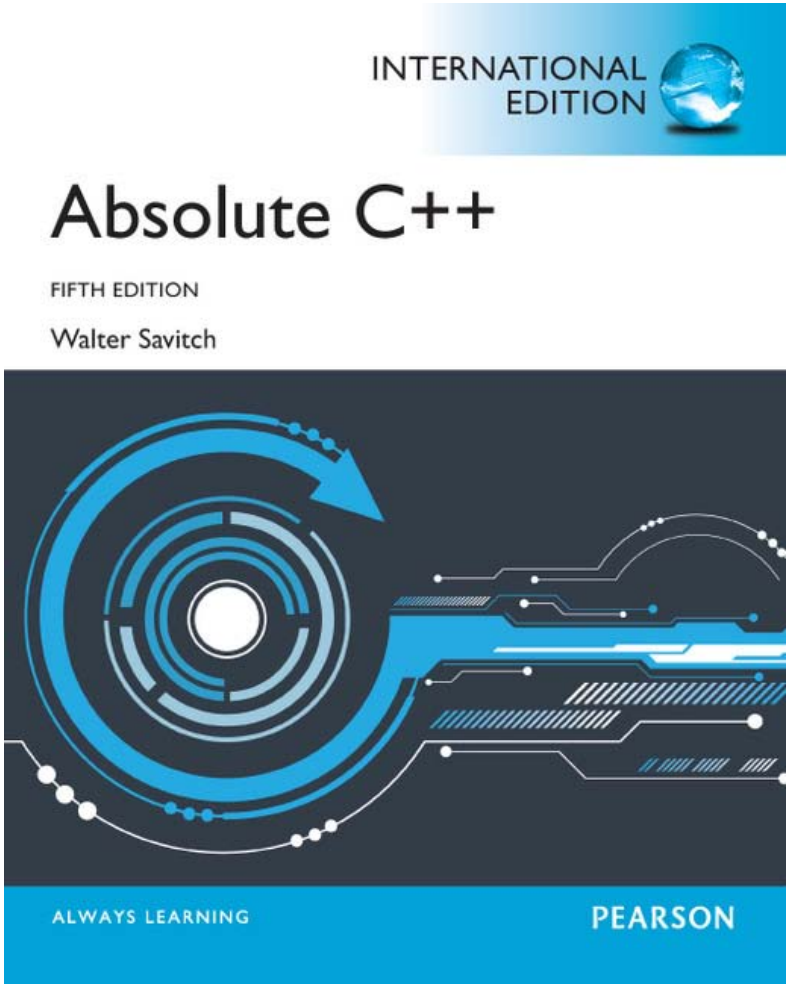
http://140.123.101.97/login.htm

Required Text

C++: How to Program,
Deitel and Deitel,
7th Edition,
Prentice Hall,
ISBN: 0-13-611726-0.



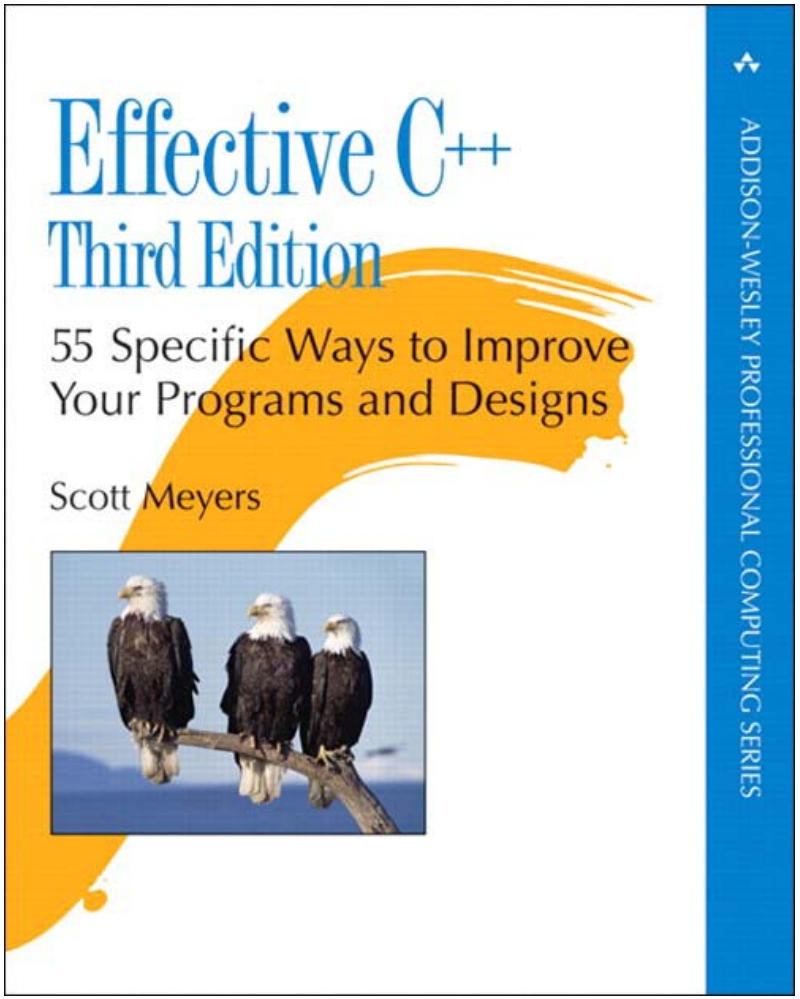
Reference Text



Absolute C++,
Walter Savitch,
5th Edition,
Pearson,
ISBN: 0-273-76932-4.

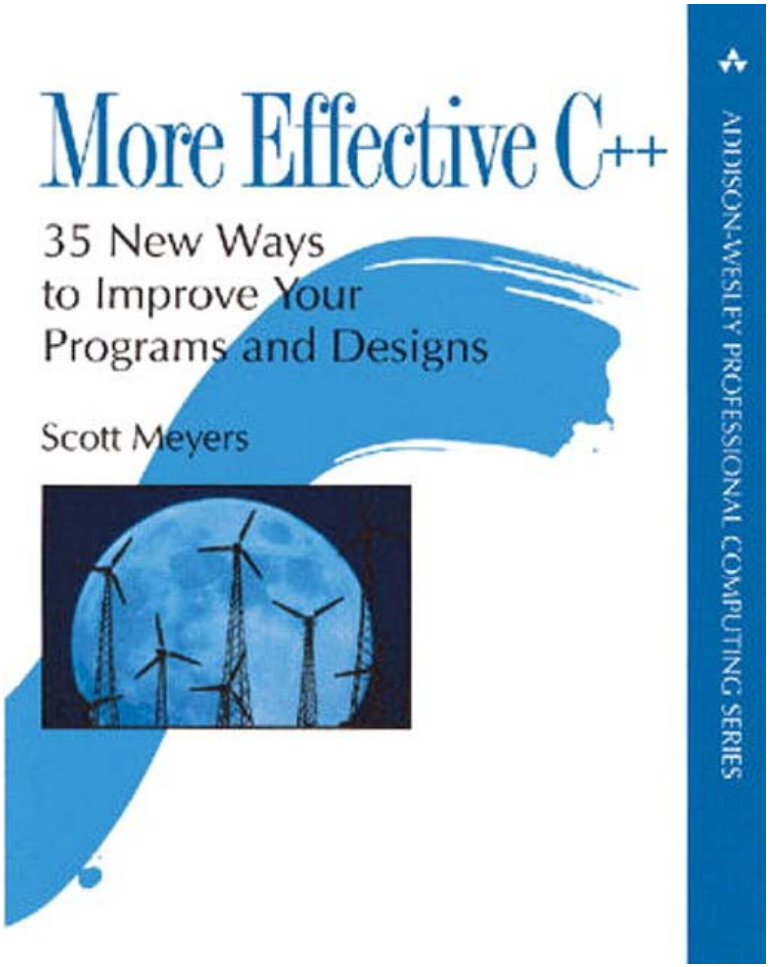
Readings

Effective C++:
55 Specific Ways to Improve Your Programs and Designs,
Scott Meyers,
3rd Edition,
Addison Wesley,
ISBN: 0-321-33487-6.



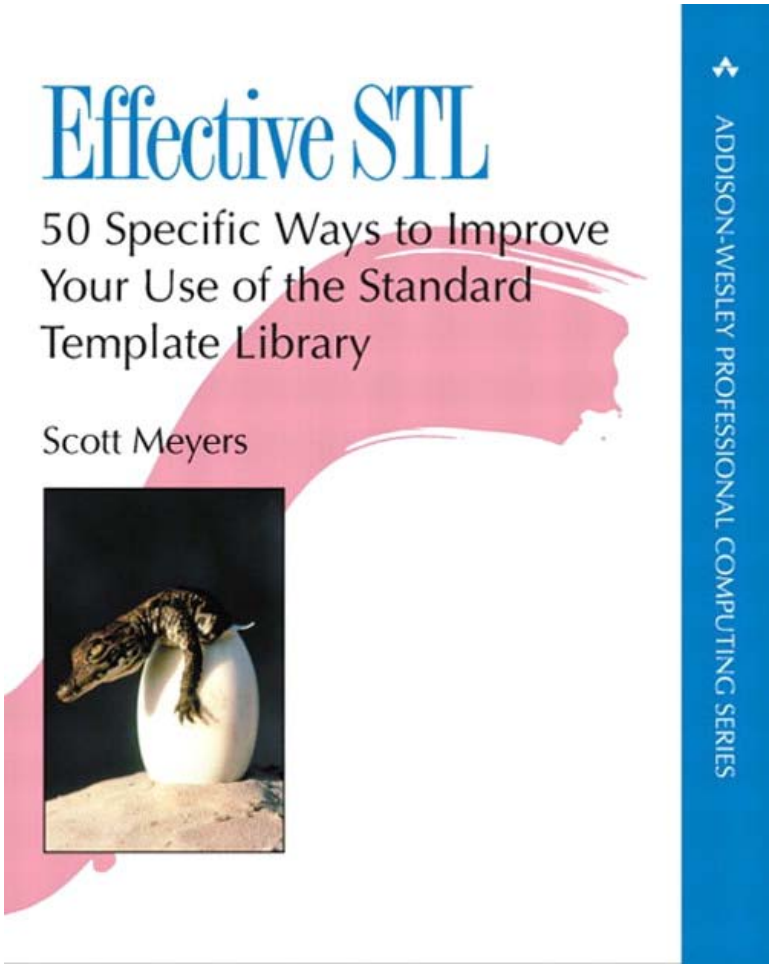
Supplementary Readings

More Effective C++:
35 New Ways to Improve Your Programs and Designs,
Scott Meyers,
Addison Wesley,
ISBN: 0-201-63371-X.



Extra Readings

Effective STL:
50 Specific Ways to Improve Your Use of the Standard Template Library,
Scott Meyers,
Addison Wesley,
ISBN: 0-201-74962-9.

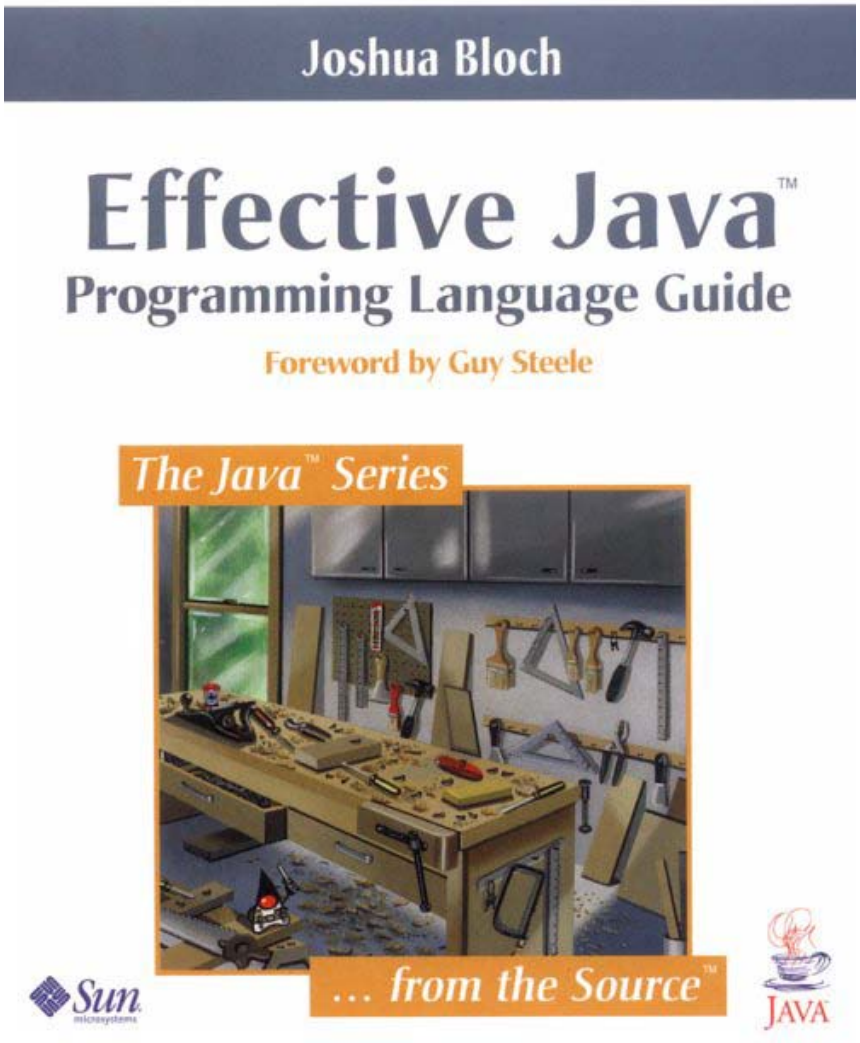


More Readings

Java: How to Program,
Deitel and Deitel,
6th Edition,
Prentice Hall,
ISBN: 0-13-148398-6.

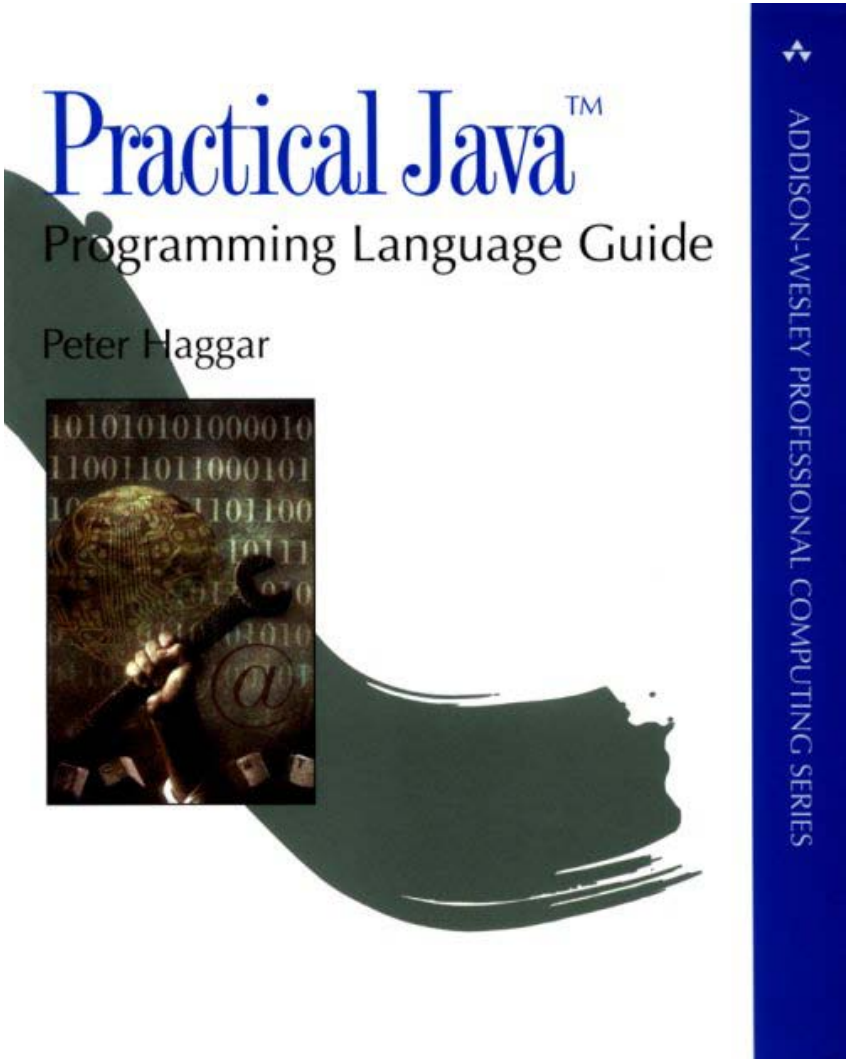
And More Readings

Effective Java Programming Language Guide,
Joshua Bloch,
Addison Wesley,
ISBN: 0-201-31005-8.



Even More Readings

Practical Java Programming Language Guide, Peter Haggar, Addison Wesley, ISBN: 0-201-61646-7.



Even Even More Readings

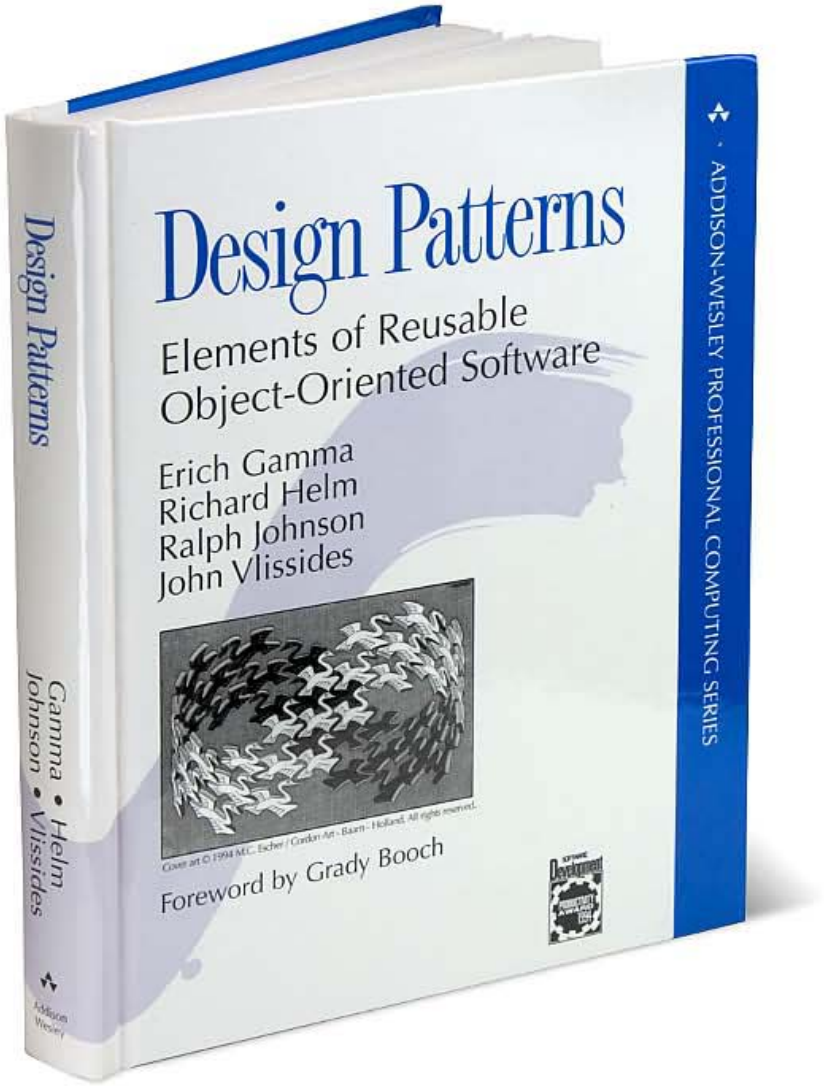
Design Patterns: Elements of Reusable Object-Oriented Software,

Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides,

Addison Wesley,

ISBN: 0-201-63361-2.

(Gang of Four)



Indicative Topics

1. Object-oriented design
2. An overview of C++
3. A crash course in Java
4. Functions & arrays
5. Pointers & strings
6. “const”-ness
7. Implementing classes

Indicative Topics (2)

8. OOP in Java
9. Operator overloading
10. Inheritance
11. Virtual functions & polymorphism
12. Templates
13. Graphical user interfaces in Java
14. Exception handling

Indicative Topics (3)

15. Multithreading in Java
16. Synchronization in Java
17. Networking for Java
18. Advanced C++
19. Effective Java
20. Design patterns

Evaluation

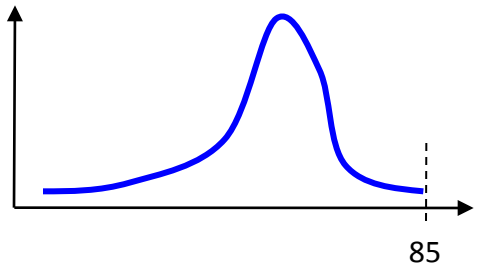
Programming projects (five – seven).....25%

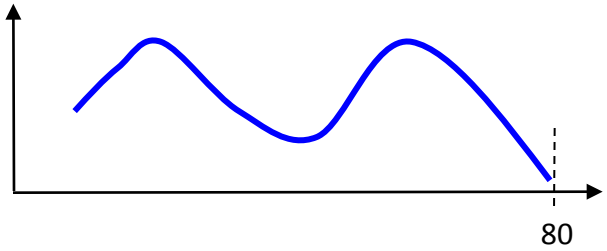
Midterm exam.....37%

Final exam.....35%

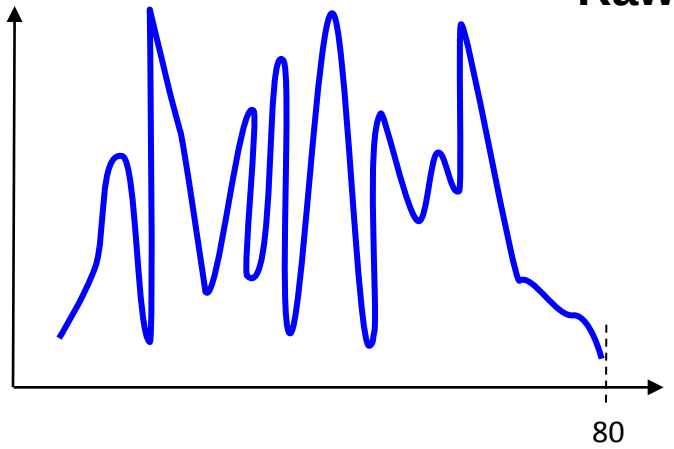
Class participation.....3%

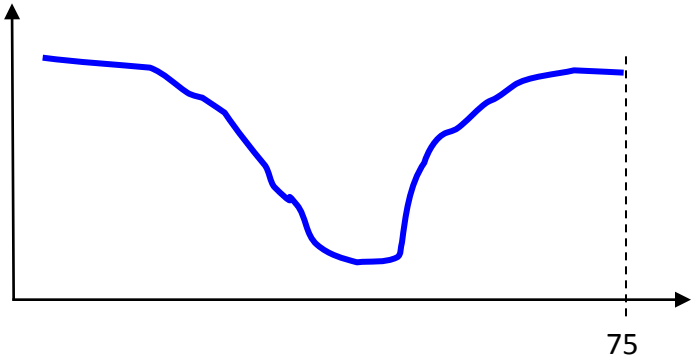
Both Java and C++ will be the implementation languages.

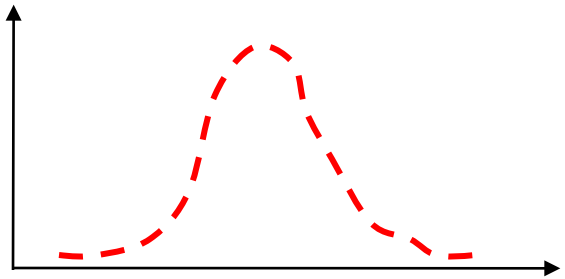




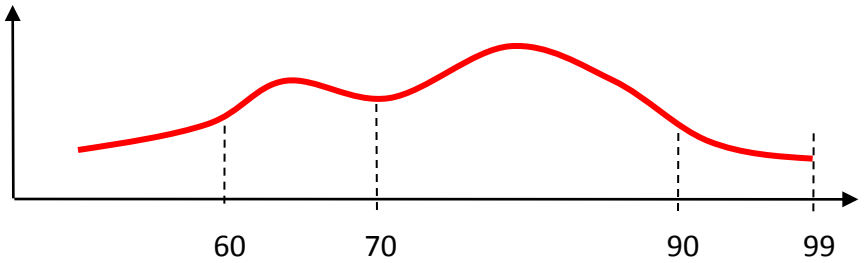
Raw scores







Non-linear scaling



千金難買早知道 **A word before is worth two behind.**

Academic Honesty Policy

- **All graded materials (examinations and programming assignments) must be strictly individual efforts.**
- **Students who cheat skew the grading curve in a class, resulting in lower grades for students who worked hard and did their own work.**
- **Cutoff for passing grades at the end of the semester will be the raw score 60 or close to the scale of 45%, depending on which one is lower.**

Academic Honesty Policy

- Programming projects are to be carried out individually. Students are forbidden from copying code, makefile, or any other material from the Internet. If you look at one, record it in your REAME. Any violation of this policy will also be considered cheating.

- Discussions with other people are also permitted and encouraged. Working with other students can be beneficial but the rules below must be followed. The actual program submitted by you must be individually yours. You must understand what the program is doing and why it is doing it.

- If two or more students work together on a programming assignment, all students are to acknowledge this in their individual program documentation (and README).

- If student A asks student B for help then it is student B's responsibility to see that A understands the problem and solution. It is A's responsibility to acknowledge B's assistance in the program documentation (and README).

- We believe that by encouraging students to submit their own work and, where possible, removing instances of copying and plagiarism of computer program submissions from the course we can better help weaker students to overcome their problems and guarantee that work actually done is correctly rewarded.

CodeSim 程式碼相似度比對工具

- **http://delphi.ktop.com.tw/board.php?cid=31&fid=79&tid=100527**

- 助教會利用程式碼相似度比對軟體來偵測可能的抄襲。一旦抄襲，該同學們 (無論誰抄襲誰) 全學期的作業成績就是零。

嚴格遵守學術倫理

- 有許多學生引用學長姐留下來的專案加以修改。造成了助教批改作業時發現有非常多的作業擁有相同格式的介面，這樣子被認為算是抄襲。

- 有下列情形者，應在readme上註明出處：
  1.參考了某某人的專案，改寫了裡面的內容，但是架構是一樣的。
  2.參考了某個網站的教學，複製了裡面的某段讀圖程式或者某段函式。
  3.只要這個專案不是你從零開始寫的。你都應該要在readme上註明出處。

- 例如:
- 我編修了602410000王天才的程式，使用了他的介面，其餘”主要主幹”還是自己撰寫。

  （O）這樣的情形你要事先告知。但助教還是會依修改程度扣分。

- 我觀看了www.example.com/howToC++/hw3.php裡面的seam教學。

  但所有程式碼都是我自己撰寫的，沒有複製裡面的函式或迴圈。

  （X）這樣的情況不是抄襲，可以不必寫。

- 我參考了www.example.com/howToC++/hw3.php裡面的seam教學。裡面有一個彩色to灰階的函式（此函數為”非”作業主幹），我是複製下來用的。

  （O）這樣的情況允許接受，但你要事先告知。

- 我參考了602410012王天才的程式，我只改了一小部分。

  （X）這樣的情況你告知了，還是０分。

The Generation of Programming Languages

First generation:

- consists of global data and subprograms.
- an error in one part of a program can have a devastating effect across the rest of the system.

The Generation of PL's (2)

Second generation (such as FORTRAN and COBOL):

- supports parameter-passing mechanisms.
- fails to address the problems of programming-in-the-large design.

The Generation of PL's (3)

Third generation (such as C):

- supports modular structures that can be compiled separately.
- each module can have its own data.
- still algorithm (not data) driven design.

Object-Oriented Programming (OOP)

A method of implementation in which programs are organized as cooperative collections of objects, each of which represents an instance of some class, and whose classes are of members of a hierarchy of classes united via inheritance relationships.

Thinking about Objects

- Reusable software components that model real-world items.
- Look all around you: people, animals, plants, cars, etc.
- Attributes: size, shape, color, weight, etc.
- Behaviors: babies cry, crawl, sleep, etc.

Elements of Object Model

- Abstraction
- Encapsulation
- Modularity
- Hierarchy

Types of Abstraction

Procedural abstraction:

- Specify what actions should be performed.
- Hide algorithms.

Data abstraction:

- Specify data objects for problem.
- Hide representation.

Abstraction & Encapsulation

Abstraction and encapsulation are complementary concepts. Abstraction focuses on the observable behavior of an object, whereas encapsulation focuses on the implementation that gives rise to this behavior.

Object-Oriented Design

- Models real-world objects.
- Models communication among objects.
- Encapsulates data (attributes) and functions or methods (behaviors):
    - information hiding.
    - communication through well-defined interfaces.

Classes

- A class is a blueprint or prototype that defines the variables and methods common to all objects of a certain kind.

- An object is an instance of a class.

Life of an Object

1. Allocation of data members.
2. Construction (of the object).
3. Usage.
4. Destruction.
5. De-allocation of data members.

Orders matter!

Object-Oriented Analysis and Design (OOAD)

- Essential for large programs.
- Analyze program requirements, then develop solution.

OO Programming Languages

- Development history:
  - **Simula (Dahl & Nygaard, 1962)**
    - **Modeling discrete event simulation**
  - **Smalltalk (Kay, 1972)**
    - **General programming**
  - **C++ (Stroustrup, 1979)**
    - **Manage complexity in huge software projects**
  - **Java (Gosling, 1991)**
    - **Designed for embedded processors**

History of Java

- Originally for intelligent consumer-electronic devices (1991 a research project to develop a C/C++ system for **set top box** called “Oak”).
- Then used for creating Web pages with dynamic content.
- Java is modeled after C++, but omits its complex parts. Java is formally announced in May 1995.

History of Java (cont)

- Java is portable (platform-independent): "**write once, run anywhere**". C++ applications written for different platforms are not portable.
- Now also used for:
  - develop large-scale enterprise applications.
  - enhance WWW server functionality.
  - provide applications for consumer devices (cell phones, etc.)

How is Java Different?

- Designed ***from the ground up*** as object-oriented.
- No pointers (hidden from the programmers).
- No memory allocation (automatic with garbage collection).
- Multithreaded.