PD2 Lecture 0 - Course Overview

Generated by DALL.E

Program Design II @ CSIE NCKU 2024

Instructor Biosketch - 成功大學資訊工程學系 莊坤達 Kun-Ta Chuang 副教授

Current Position

- 成功大學數位生活科技研發中心 主任 2020-Now
- 成功大學健康服務暨數位創新推動辦公室 副執行長 2023-Now
- 行政院 科技會報辦公室 科技政策專家室智慧資通群組領域專家 2020-Now
- 台南市 研考委員 2022-Now
- 教育部 大數據分析人才培育中心 主持人

Research Interest

- Data Science/Machine Learning/Blockchain
- Applied Domain
  - Semiconductor Automation
  - Intelligent Energy
  - Intelligent Healthcare

產學合作單位

東京電力	**伊藤忠商社**	**Google**	**Microsoft**
NXP	**Infineon**	**台積電**	**帆宣科技**
台灣電力	**凱基人壽**	**智邦科技**	**奇美醫院**

In the past

- Startup -- Youthwant (1999-2001)
- Startup -- UniPattern (2002-2004)
- Join Synopsys (2006 – 2011)
  - Served as a Senior Engineer (國防役)
  - Synopsys is the world leader in EDA
  - Division of Synopsys Design-Rule Check (DRC)

- >40 engineers and totally ~100 engineers worldwide (Taipei, MV, RTP)

Prerequisite

在學習 PD2 之前，需具備以下先備知識：

- 基本的電腦操作技能
- 熟悉至少一種程式語言的基本概念 (請先修過PD1@CSIE)
- 了解變數、資料型態和基本的程式邏輯
- 熟悉基本的算術和邏輯運算
- `有程式設計的基礎`

> ➡️ Before learning PD2, it is necessary to have the following prerequisite knowledge:
>
> - Basic computer operation skills
> - Familiarity with basic concepts of at least one programming language (Please complete PD1@CSIE first)
> - Understanding of variables, data types, and basic programming logic
> - Familiarity with basic arithmetic and logical operations
> - `Basic understanding of programming concepts and logical thinking`

PD2 課程主軸

我們不教怎麼寫程式
我們教怎麼開發軟體

> 從 「程序式程式(Procedural Programming)設計」 進化到「物件導向程式(Object-Oriented Programming)設計」

- 一直寫C，你會寫程式而難以看到軟體的全貌、服務層(Service Layer)架構、系統分析(System Alanysis)及系統設計(System Design)等任務的重要性
- 太高階的R或Matlab等高階程式語言，屬於混合Procedural、OO概念，在使用上朝上Low-Code概念
- C++/Python/Java等物件導向程式，能協助你邁入架構化、抽象化、一般化的程式思維

誤入歧途當碼農

從「直覺化寫程式」進化到利用「資料結構與演算法」基礎

- 是否你看到程式題目第一步就是開一個10000 x 10000的array呢？
- 你的程式是不是一直都是3層迴圈跟一堆if/else呢
- 思考 程式效能 及 資源消耗
- 程式思維從土炮(straightforward)到優雅(graceful)

決定當個高階碼農

從「程式撰寫」提升到 「軟體開發」

- 從 個人思考 進階到 團隊思考 ；從 思考程式效能進階到 平台效能 ；從 能跑就好 進階到 系統穩定
- 軟體要成功，程式能力很重要，但更重要的事，根本不是程式，不然我國應該早就是國際知名軟體大國了!
- 你知道矽谷現在都在裁工程師嗎?
- 協作思維及ToolChain概念

叫別人當碼農

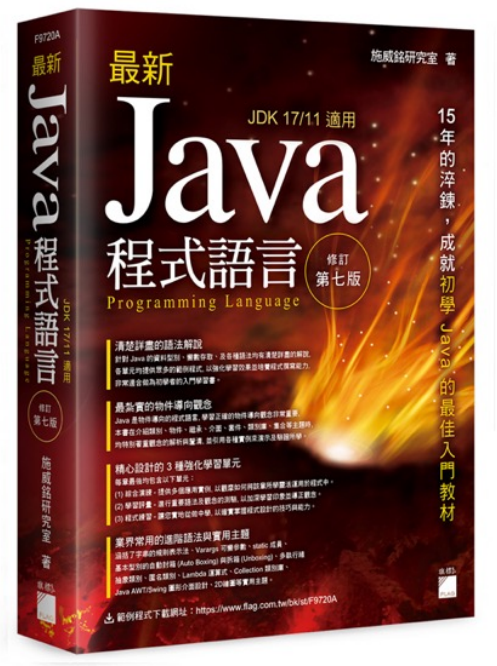
PD2 Syllabus

週次	內容	Task
第1週 2/20	課程簡介與Java介紹	
第2週 2/27	Java基本語法介紹 I	**HW1**
第3週 3/5	Java物件導向程式設計 I	
第4週 3/12	Java物件導向程式設計 II	**HW2**

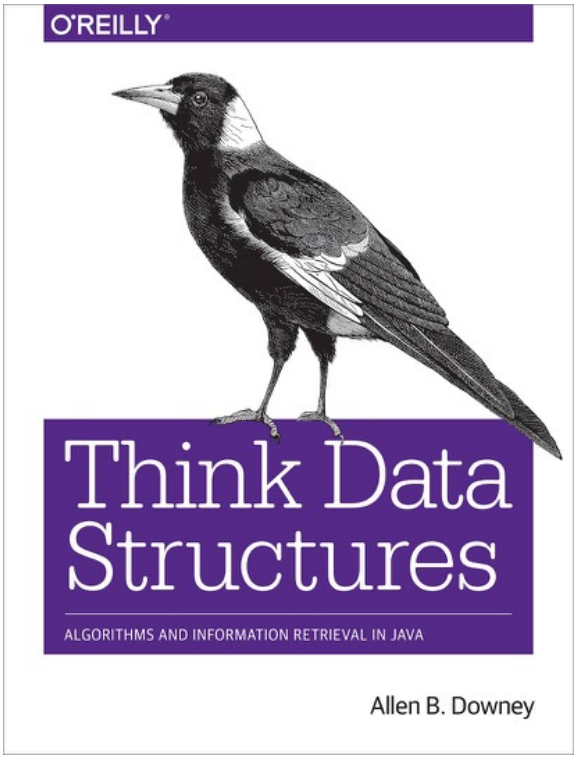
週次	內容	Task
第5週 3/19	Java物件導向程式設計 III	
第6週 3/26	**Practice for Data Structure/Algorithm I: List, Trees, Sorting, DP, Matching, Encoding**	
第7週 4/2	**Practice for Data Structure/Algorithm II: List, Trees, Sorting, DP, Matching, Encoding**	**HW3**
`第8週 4/9`	**School Holiday**	
第9週 4/16	**Practice for Data Structure/Algorithm III: List, Trees, Sorting, DP, Matching, Encoding**	**HW4 (TBD)**
第10週 4/23	**Practice for Data Structure/Algorithm IV: List, Trees, Sorting, DP, Matching, Encoding**	
第11週 4/30	`Midterm Exam`	
第12週 5/7	**Practice for Data Structure/Algorithm V: List, Trees, Sorting, DP, Matching, Encoding**	**Final Project Announcement**
第13週 5/14	**Advanced Skill: Software Development Toolchain**	**HW5 (TBD)**
第14週 5/21	**Advanced Skill I: ToolChain, CI, Refactoring, Design Pattern**	
第15週 5/28	**Advanced Skill I: ToolChain, CI, Refactoring, Design Pattern**	
第16週 6/4	**Advanced Skill II: ToolChain, CI, Refactoring, Design Pattern**	
第17週 6/11	**Advanced Skill III: ToolChain, CI, Refactoring, Design Pattern**	

週次	內容	Task
第18週 6/18	**Team-based Final Project Presentation**	**Team-based Final Project Presentation**

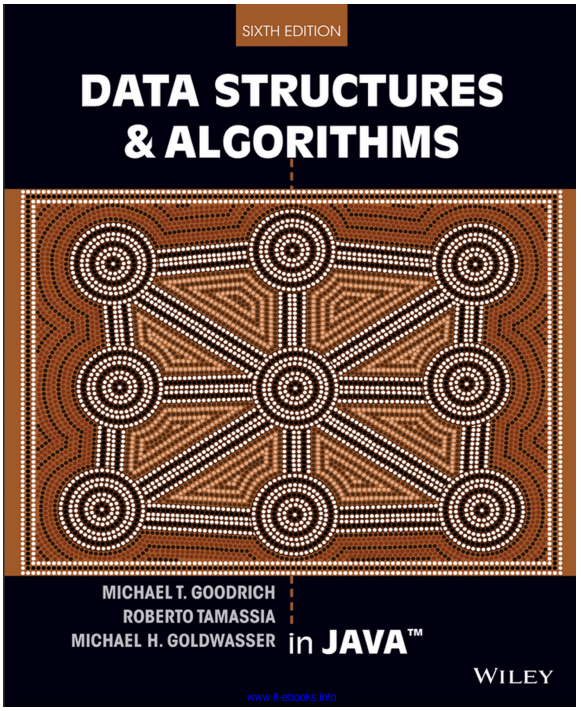
Reference Books



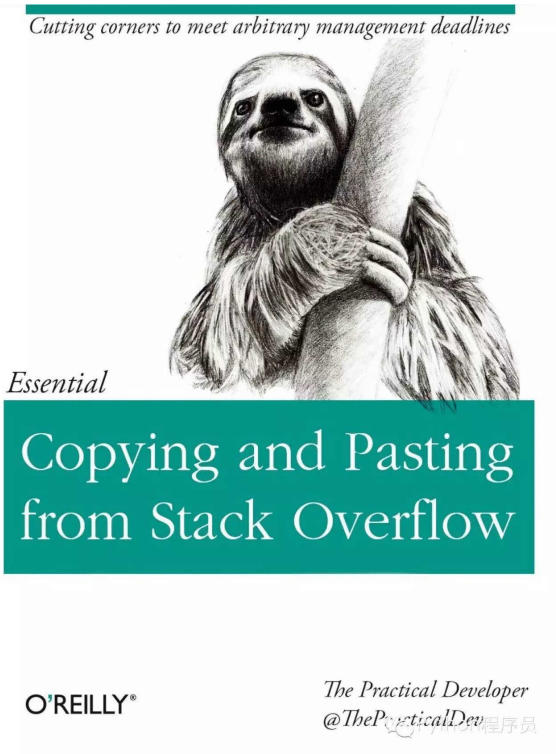
最新 Java 程式語言(修訂第七版)



Think Data Structures Algorithms and Information Retrieval in Java, pdf download

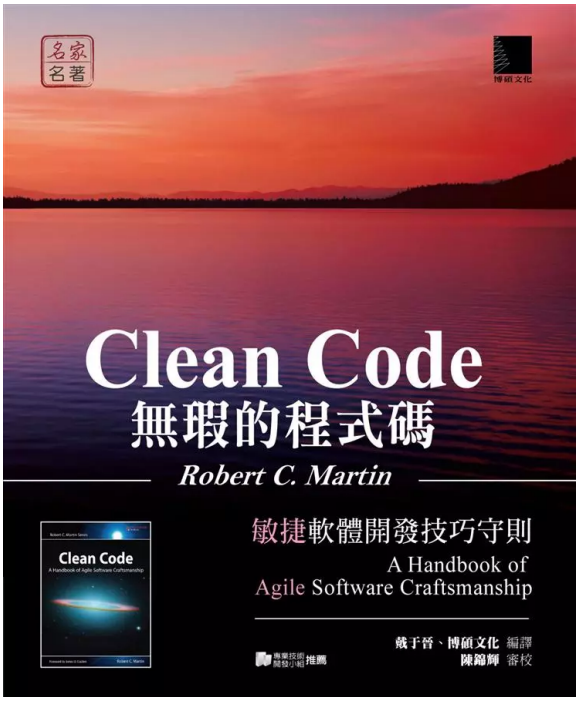


Data Structures and Algorithms in Java, 6th Edition, pdf download

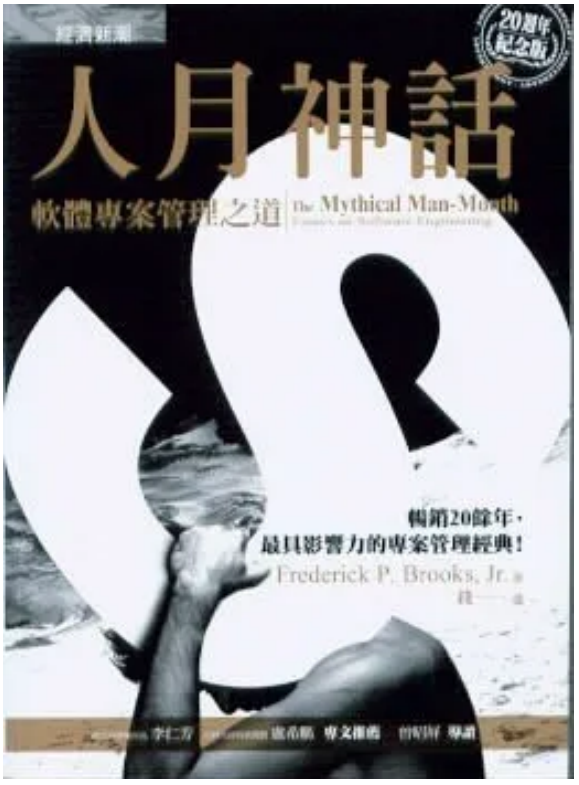


工程師必備技能

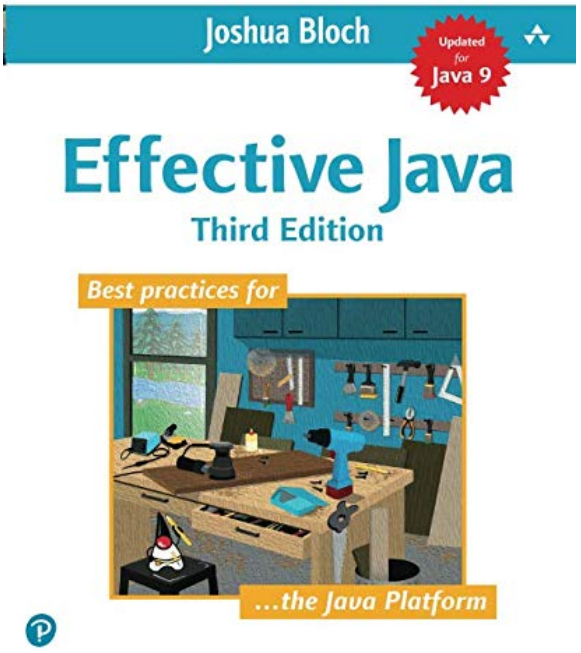
Bible



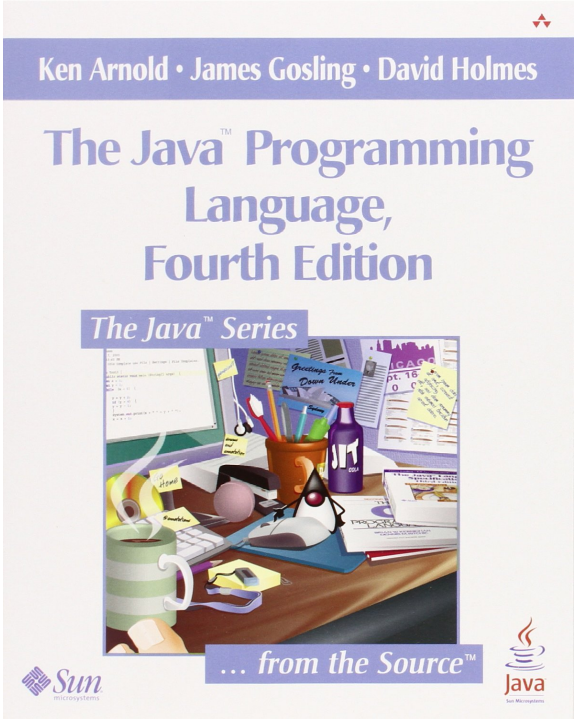
Robert C. Martin, “Clean Code 無瑕的程式碼: 敏捷軟體開發技巧守則”



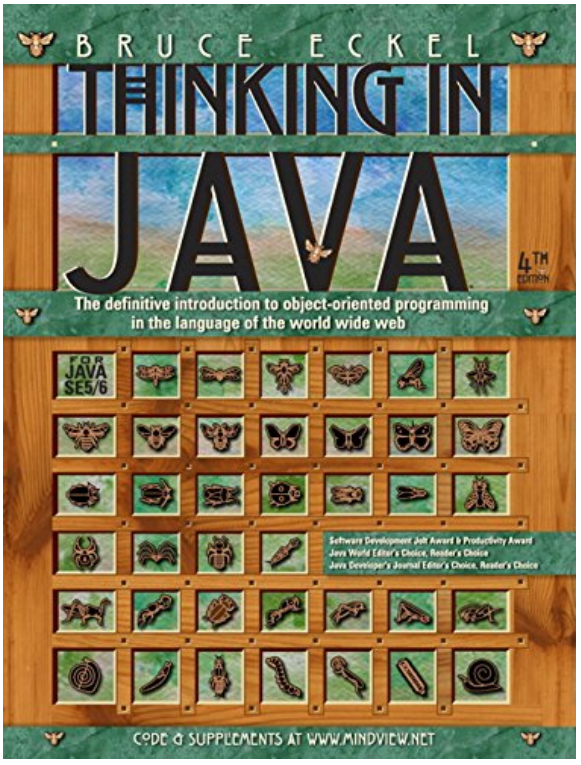
Fred Brooks, “人月神話：軟體專案管理之道, The Mythical Man-Month: Essays on Software Engineering”



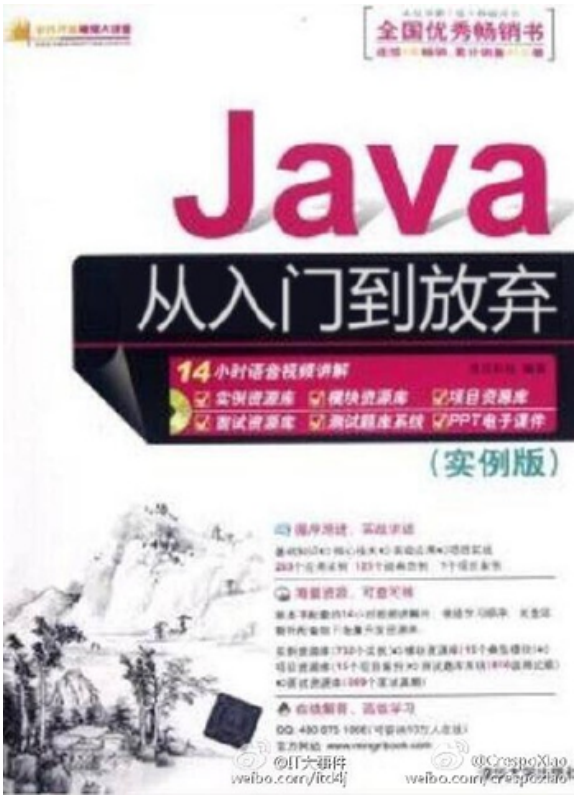
Joshua Bloch, "Effective Java", 3/e



Ken Arnold, James Gosling*, David Holmes, "The Java Programming Language", 4/e



Bruce Eckel, Thinking in Java, 4/e



買到送我一本

Other good web references:

- w3schools Java tutorial https://www.w3schools.com/java/default.asp
- Paul N. Hilfinger, “A Java Reference: Assorted Java Reference Material”, UC Berkeley download
- Increment Java, CS of U. Maryland, https://www.cs.umd.edu/~clin/MoreJava/
- https://openhome.cc/zh-tw/java/

Grading 成績

Item	Score
Programming Homework 1~5	`11% each, 55% in total`
Midterm Exam	`20%`
Final Project	`25%`
Attendance (possible bonus)	`1% each, 2% at most`

Offline Study

- 除期中考試/Final Project Presentation外，你不一定需要出席
- 我們每次上課後，均會提供錄影供你參考(於Moodle上提供)

Homeworks

📖 Homework Guideline:

- 每一個作業依規模大小，有至少兩個星期的實作時間
- 一律不準遲交，除非事前以合理理由申請
- 一律以在class server繳交原始碼為依據

📖 Homework Guideline:

- Each homework, depending on its size, has at least two weeks of implementation time
- No late submissions are allowed unless previously asked with a reasonable reason
- All are based on submitting the source code on the class server

Final Project形式

```
%%{init: {"quadrantChart": {
quadrantChart
  x-axis Signle/Small Team -
  y-axis Good Collaboration
  quadrant-1 "適合當公務員"
  quadrant-2 "不知道適合做什麼"
  quadrant-3 "適合去創業"
  quadrant-4 Excellent
```

在你選課之前請先確認你是否有這些先備能力跟認知

Math Expression

請先確認你知道這是什麼意思 (support a = 5)

Please confirm that you understand what this means (support a = 5)

- a = a +1
- a += 1
- `a *= a^2`

```
public class App {
    public static void main(String[] args) {
        int a=5;
        a *= a^2;
        System.out.println(a);

    }
}
```

- a = b % c
- a++ vs. ++a
- a-- vs. --a
- `&` vs. `&&`

Basic Programming Concept

- `NULL`是什麼？
- 你是否有"Debug"的概念?

Homework Expectation

- 5個HW程式碼基本均不會少於500行，大則超過千行
- 部分HW需處理到100MB大小的檔案

Basic Computer Engineering Skill

- 當你遇到問題時，請詢問 'Google'、'GPT' 或 'Stack Overflow' 來找出解決方案。
  - 助教將幫助你進行問題排查，但確保你能提出聰明的問題，將操作的基本問題交給 Google。
- `我們將使用 'Linux' 來提交作業。請確保你不怕在CLI命令行模式下操作。`
  - 我們會提供所有應該操作的cmd給大家，但如果是這種情況下你還是必須問助教step-by-step的操作，那應該思考本堂課會給你修課上龐大的壓力
- 你了解文字編碼問題。知道如何使用 'UTF-8' 編碼你的code。

📖 Basic Engineering Skill

- When you face a question, ask ‘google’, ‘GPT’, or ‘Stack Overflow’ to figure out the solution.
  - TA will help you to troubleshoot, but make sure you can ask smart questions, leaving essential issue of operations to google.
- `We will use ‘Linux’ for Homework submission. Please make sure that you are not afraid to operate in the CLI console mode.`
  - We will provide all the cmd operations you should perform, but if you still need to ask the teaching assistant for step-by-step operations under this circumstance, you should consider that this course may put a lot of pressure on you.
- You know the text encoding issue. Know use ‘UTF-8’ encoding your source code.

Why use Java in PD2?

According to Penetration Rate in Open Source community

➡️ According to the largest community in OpenSource World https://projects.apache.org/projects.html, the usage numbers of various programming languages in Apache projects:

Programming Language	Number of Projects
C	249
C++	187
Java	329
JavaScript	130
Perl	115
PHP	56
Python	306
Ruby	28

Generated by DALL.E

APACHE介紹

Apache軟體基金會（Apache Software Foundation）是一個非營利的開源組織，致力於支持和發展開源軟體專案。該組織於1999年創立，目前管理著超過350個開源專案，涵蓋各個領域，包括Web伺服器、大數據、人工智慧、雲端計算等。

Apache以其開放和合作的精神，吸引了全球眾多開發者和貢獻者參與其中。該組織提供了一個開放和民主的決策架構，讓每個專案的發展都能夠獲得社群的共識和支持。

開源組織APACHE的目標是促進開源軟體的創新和發展，並提供一個半個穩定和可靠的平台供開發者使用。通過共享代碼和知識，APACHE不僅推動了開源軟體的普及，也推動了技術的進步和社群的成長。

更多關於Apache軟體基金會的資訊，請參閱官方網站：https://www.apache.org/

➡️ The Apache Software Foundation (ASF) is a non-profit organization that supports and develops open-source software projects. It was founded in 1999 and currently manages over 350 open-source projects in various domains, including web servers, big data, artificial intelligence, and cloud computing.

The ASF attracts developers and contributors from around the world with its open and collaborative spirit. It provides an open and democratic decision-making framework that allows each project to gain consensus and support from the community.

The goal of the Apache Software Foundation is to foster innovation and development in open-source software and provide a stable and reliable platform for developers. By sharing code and knowledge, the ASF not only promotes the adoption of open-source software but also drives technological advancements and community growth.

For more information about the Apache Software Foundation, please visit their official website: https://www.apache.org/

According to Popularity in Worldwide Searching

Check worldwide search index - https://pypl.github.io/PYPL.html

https://pypl.github.io/PYPL.html

Good Discussion on Internet

- https://buzzorange.com/techorange/2020/11/03/python-exceeds-java/

My two cents

💡 Java的強大的社群及JVM能力:

- Java Garbage Collection：Java的Garbage Collection機制為程式開發提供了一個更安全、更高效的環境，使得開發人員可以更專注於業務邏輯的實現，而不必擔心記憶體管理的細節。這不僅提高了開發效率，也提升了程式的穩定性和安全性。
  - `缺點` 你會不知道什麼是memery leak、dangling pointers、buffer overflow等系統核心概念，如果你有志做為Architect等級的工程人員，這些概念是如喝水般的直覺
- Java的運用泛用性：Java的泛用性和強大的套件支持源於其穩定的跨平台性能、豐富的API、以及一個龐大的開發者和貢獻者社群。這使得Java成為了一種適合於各種應用領域的編程語言，從企業應用到Mobile computing、Multimedia、Network、AI、Database、Cloud、Security等，Java都能提供高效、穩定的解決方案。
- Java的廣大使用群及優雅內涵的Design Pattern能提供很好/易懂的coding reference典範：
  - Python常為求快速應用及low code，多數範例較為直覺式的coding style，接近`指令`思維而非`架構`思維
  - C/C++常為求效能及低階優化，多數範例較不易理解，過多memory轉換及platform-depend的structure
- Java的高階應用彈性：Java具易讀性、可維護性和高效性為特點，這使得Java編程更加容易入手。Java語言也支持`跨平台`操作，可運行於任何支持JVM的平台上，這意味著Java程序員不必對底層電腦進行過多關注，因此更適合用於大型企業用服務、Internet服務等
  - `缺點` Java易讀且求跨平台一致性的本質，對coding思維形成了一定的局限，從某種程度上來說Java更注重開發者“便捷、安全、快速構建應用”的目標。但同時容易讓一些“底層操作技術水平較高“的任務，難以被初階的工程師理解。
- Java十分特別的語言火力：`Dynamic Binding`及`Serialization`

A Good Example to Observe Java/Python/C++

💡 請讀取一個純文字檔，裡頭有包含超過千篇英文文章。請寫一個java程式，將此文字檔中所有出現的英文單字，各自出現的次數統計出現。最後依出現次數，從最高到最低列出其對應的英文單字

```
graph LR
    A[開始] --> B[打開檔案]
    B --> C{檔案讀取完畢？}
    C -- 是 --> D[結束]
    C -- 否 --> E[讀取下一個單詞]
    E --> F[標準化單詞]
```

```
    F --> G[統計單詞出現次數]
    G --> C
    D --> H[將結果按出現次數排序]
    H --> I[輸出結果]
```

Java

```
import java.io.*;
import java.util.*;

public class WordFrequencyCounter {
    public static void main(String[] args) {
        // 讀取檔案並統計單字頻率
        Map<String, Integer> wordCounts = new HashMap<>();
        try {
            File file = new File("textfile.txt");
            Scanner scanner = new Scanner(file);
            while (scanner.hasNext()) {
                String word = scanner.next().toLowerCase(); /
                word = word.replaceAll("[^a-zA-Z]", ""); // 移
                if (!word.isEmpty()) {
                    wordCounts.put(word, wordCounts.getOrDefa
                }
            }
            scanner.close();
        } catch (FileNotFoundException e) {
            System.err.println("檔案未找到: " + e.getMessage())
            return;
        }

        // 將map入口轉換成列表，並根據次數排序
        List<Map.Entry<String, Integer>> sortedWords = new Ar
        sortedWords.sort(Map.Entry.<String, Integer>comparing

        // 打印結果
        for (Map.Entry<String, Integer> entry : sortedWords)
            System.out.println(entry.getKey() + ": " + entry.
```

```
        }
    }
}
```

Python

```
from collections import Counter
import re

def count_word_frequencies(file_path):
    with open(file_path, 'r') as file:
        words = re.findall(r'\b[a-zA-Z]+\b', file.read().lo
wer())
    word_counts = Counter(words)
    for word, count in word_counts.most_common():
        print(word, ':', count)

# 替換成你的檔案路徑
file_path = 'textfile.txt'
count_word_frequencies(file_path)
```

C++

```
#include <iostream>
#include <fstream>
#include <map>
#include <vector>
#include <algorithm>
#include <cctype>
#include <string>

// 用於排序的比較函數
bool compare(const std::pair<std::string, int>& a, const std:
    return a.second > b.second;
}
```

```
int main() {
    std::string filename = "textfile.txt"; // 更換為你的文件名
    std::ifstream file(filename);
    std::map<std::string, int> wordCount;
    std::string word;

    if (!file.is_open()) {
        std::cerr << "無法打開文件: " << filename << std::endl;
        return 1;
    }

    while (file >> word) {
        // 轉換為小寫
        std::transform(word.begin(), word.end(), word.begin()
                       [](unsigned char c){ return std::tolow
        // 移除非字母字符
        word.erase(std::remove_if(word.begin(), word.end(), [

        if (!word.empty()) {
            wordCount[word]++;
        }
    }

    file.close();

    // 將map轉移到vector以便排序
    std::vector<std::pair<std::string, int>> vec(wordCount.be
    std::sort(vec.begin(), vec.end(), compare);

    // 打印結果
    for (const auto& pair : vec) {
        std::cout << pair.first << ": " << pair.second << std
    }

    return 0;
}
```

💡 Discussions:

- C++演算法的generic style (如std::sort) 跟Java class內含的List.sort()之別
- Python你寫太長的程式的話，你還叫Python工程師嗎
- C++的memory usage思維需十分明確

Object-Oriented Programming vs. Procedure Programming

Feature	Object-Oriented Programming	Functional/Procedure Programming
Basic Concept	Centers around classes and objects, emphasizing encapsulation, inheritance, and polymorphism.	Focuses on functions as the basic unit, emphasizing immutability and side-effect-free functions.
Data Handling	Operates on data through object methods.	Processes data using pure functions, avoiding data sharing between functions as much as possible.
State Management	Objects can have persistent state.	Prefers using immutable data structures, function output depends solely on input parameters.
Code Organization	Encapsulates data and the methods that operate on the data together into classes.	More flexible code organization, functions can exist independently of data.
Reusability & Modularity	Reuses code through inheritance and composition.	Reuses code through higher-order functions (functions as parameters or return values).
Design Philosophy	Emphasizes the identity of objects, as collections of data and behaviors.	Emphasizes data immutability and the mathematical properties of functions, focusing on data

Feature	Object-Oriented Programming	Functional/Procedure Programming
		flows and pipelines of function processing.
Code Understanding & Maintenance	Code complexity might increase due to inheritance and polymorphism.	Code can be more concise and clear, but highly abstract function compositions may require a deeper understanding of functional programming concepts.

IDE for JAVA

Visual Studio Code

Installation Guide - https://code.visualstudio.com/docs/languages/java

中文Reference

https://www.dropbox.com/scl/fi/j8p3lfer7d6iaza4fb0zk/F9720_bonusA.pdf?rlkey=cfhckozisxi4wn0tx42uasnkj&dl=0

Eclipse

https://www.eclipse.org/

Installation Reference (in Windows, thanks to the sharing ecosystem)- https://medium.com/@zera.tseng888/eclipse-安裝與設定-4ddf245772f1

VS Code與Eclipse比較

工具	優點
Eclipse	• 強大的Java開發環境，提供豐富的功能和工具 • 具有強大的代碼編輯和自動完成功能 • 支持多種插件和擴展，可滿足不同的開發需求 • 內置的調試器和測試工具，方便進行代碼調試和測試 • 良好的項目管理和版本控制功能

工具	優點

Visual Studio Code	• 輕量級的開發環境，啟動速度快且占用系統資源少 • 具有豐富的擴展和插件生態系統，可根據需要進行擴展和定制 • 內置的Git支持，方便進行代碼版本控制 • 強大的代碼編輯功能，支持多種語言和自動完成 • 具有豐富的調試功能，支持多種語言的調試

In Windows, alternative good IDE - Visual Studio

- Visual Studio是一個全功能的集成開發環境（IDE），專注於多種語言和平台的應用程序開發。它提供了豐富的功能和工具，如代碼編輯、調試、測試、版本控制和項目管理等。它是一個較重量級的開發環境，並且需要較高的系統資源。
- Visual Studio Code是一個輕量級的代碼編輯器，適用於多種語言和平台的開發。它具有豐富的擴展和插件生態系統，可以根據需要進行擴展和定制。它具有良好的代碼編輯功能，如語法高亮、代碼補全和格式化等。它還具有內置的Git支持和豐富的調試功能。
- Visual Studio主要用於開發大型和複雜的應用程序，如桌面應用程序、網絡應用程序和移動應用程序等。它提供了更多的功能和工具，可以加快開發速度和提高開發效率。
- Visual Studio Code主要用於輕量級的代碼編輯和腳本開發。它適用於快速編輯和測試小型代碼片段，並且具有輕量級和快速的特點。
- Visual Studio提供了更豐富的調試和測試功能，包括可視化調試、性能分析和代碼覆蓋測試等。
- Visual Studio Code具有更靈活和輕量級的插件生態系統，可以根據需要添加各種插件和擴展，從而滿足不同的開發需求。

總的來說，Visual Studio適用於大型項目和專業開發者，提供了更多的功能和工具。而Visual Studio Code適用於輕量級的開發和快速編輯，具有輕量級和靈活的特點。

Good Reference for IDE Comparison - https://pypl.github.io/

https://pypl.github.io/IDE.html

Java程序執行流程

Java程序的執行流程可以分為以下幾個主要步驟：

1. 編寫Java Source Code

- 開發者使用Java語言編寫源代碼，保存為以`.java`為副檔名的文件。

2. 編譯Source Code

- 使用Java編譯器（`javac`）將`.java`文件編譯成Java binary code/bytecode（`.class`文件）。
- bytecode是一種中間代碼，獨立於機器語言，可在任何JVM上運行。

3. Load bytecode

- Java虛擬機（JVM）load這些`.class`文件。
- JVM是運行Java bytecode的環境，提供了bytecode與特定機器Machine Code之間的轉換。

4. 執行bytecode

- JVM解釋bytecode或通過即時編譯器（JIT）將bytecode編譯成特定平台的Machine Code。
  - Good Reference for JIT https://afan0918.github.io/jit/
- 這一步驟使Java程序能夠在不同的操作系統和平台上執行，實現“一次編寫，到處運行”。

跨平台執行

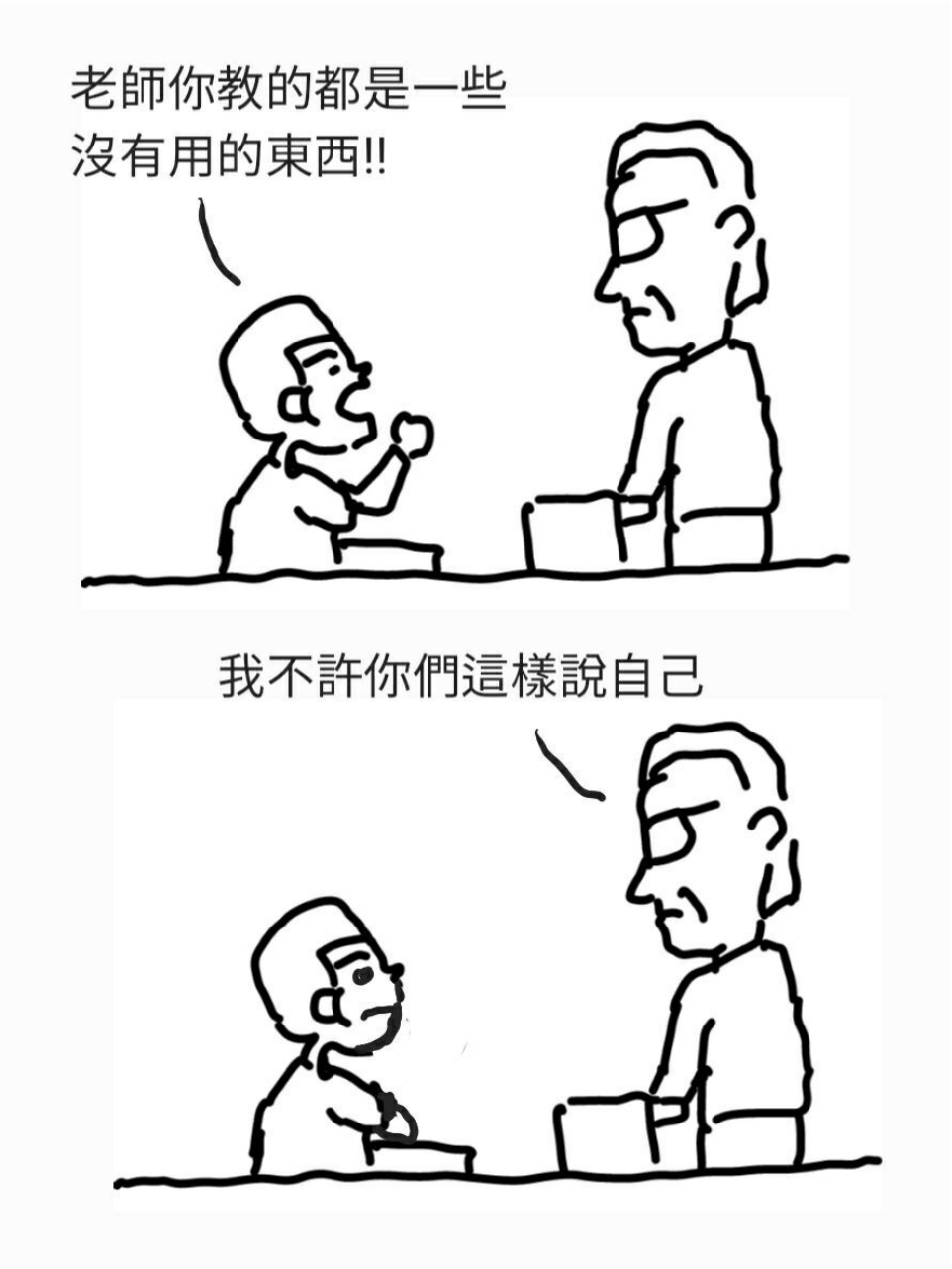
- 由於JVM可以在多種操作系統上實現，Java程序不需要為每種目標平台重新編譯。
- 使用者只需在其設備上安裝適當版本的JVM，即可運行相同的Java字節碼。

這個過程是Java高度跨平台兼容性的基礎，使得Java應用程序能夠在不同的環境中無縫運行，從而廣泛應用於Web、移動、企業和桌面應用開發中。

```
graph TD
subgraph 開發
    A[開發階段] -->|編寫源代碼| B(Java Source Code &lpar;example
    B --> |編譯| C[Java Compiler &lpar;javac example.java&rpar
    C -->|生成| D[Java Bytecode &lpar;example.class&rpar;]
end
```

```
    D --> |部署到不同作業平台| E[Java Virtual Machine &lpar;JVM&
    E -->|Windows| F[Windows上的JVM]
    E -->|macOS| G[macOS上的JVM]
    E -->|Linux| H[Linux上的JVM]
    F --> I[在Windows上執行 &lpar;java example&rpar;]
    G --> J[在macOS上執行 &lpar;java example&rpar;]
    H --> K[在Linux上執行 &lpar;java example&rpar;]
```

Q&A



Course Supplement

Functional Pointer