

Java Technicalities

From a C/C++ Programmer's View

yen3

長庚大學資訊工程學系

April 8, 2009

1 About

About Author

- Computer Science Student
- Blog: [No title, no thinking, no meaning](#)
- E-mail: yen3rc 在 gmail 答康
- 隨手書寫生活
- C, C++, Java, Haskell, L^AT_EX

About Slide

- 專題程式開發介紹相關簡報第二彈，陸續有其他介紹
- 感謝[李春良老師](#)專題指導，使得這一系列簡報得以誕生
- 感謝 Josh Ko ([Joshsoft](#)) 在主題與技術上的指導與協助(還有英文) XD

2 Introduction

But ...

- 想了很多天還是不知道自己要講什麼 XD
- 亂講好了 XD

Josh Ko and yen3's Talking

- yen3: 要講 Java 耶, 我已經不知道要說什麼了 ...Orz
- Josh Ko: XD
- yen3: 不然我講怎麼用 Java 寫 Functional Programming 好了 XD
- Josh Ko: 不太好, Java 太 OO 了 XD

Programming

By Compiler

- Compiled Programming language
- Interpreted Programming language (Script Programming Language)

By Machine

- Imperative Programming (Turing Machine)
- Functional Programming (λ -calculus)

By Model

- Procedural Programming
- Object-Based Programming (Abstract Data Type)
- Object-Oriented Programming
- Generic Programming

TIOBE Programming Community Index for March 2009

Picture

About Java

- James Gosling, 1995, Sun Microsystems
- a pure Object-Oriented Programming Language
- Everything is an object.
- Java Virtual Machine and Just-In-Time Compiler XD
- Java SE, Java EE, Java ME
- The language has already been developed for over 10 years and become a complete programming language.
- So, What do we know from Java ?

3 Library and Programming

Library

- Multi-Threading Programming – `java.util.concurrent.*`
- GUI Programming – `javax.swing.*`
- Socket Programming – `java.net.*`
- Generic Container – `java.util.*` (just for some classes) ¹
- I/O – `java.io.*` (It is a good OO design example.)²
- balabala

¹In C++, Generic Container is more interesting than Java.

²In C++, I/O Library is also designed by OO paradigm.

Java I/O

- In Unix/Unix-like, everything is a file.
- Java I/O Library is a pure OO paradigm example, and C++ is too.
- You can get an overview by `java.io.*` hierarchy tree.³ It will help you to understand how to write I/O codes.
- Be as possible as to use super classes to write codes.

For Example – class `java.io.OutputStream` and class `java.io.Writer`

- class `java.io.OutputStream`
 - class `java.io.ByteArrayOutputStream`
 - class `java.io.FileOutputStream`
 - class `java.io.FilterOutputStream`
 - * class `java.io.BufferedOutputStream`
 - * class `java.io.DataOutputStream` (implements `java.io.DataOutput`)
 - * class `java.io.PrintStream`
- class `java.io.Writer`
 - class `java.io.BufferedWriter`
 - class `java.io.CharArrayWriter`
 - class `java.io.FilterWriter`
 - class `java.io.OutputStreamWriter`
 - * class `java.io.FileWriter`
 - class `java.io.PipedWriter`
 - class `java.io.PrintWriter`
 - class `java.io.StringWriter`

Socket Programming

- Socket Programming is the same as writing to files because we consider socket as a file stream.
- use “Buffered I/O” (ex: class `java.io.BufferedWriter`, class `java.io.BufferedReader`)
- We have to be care about the message’s format.(fixed format or XML format)
- Please don’t use “\n” as a message’s end!
- Please remember “Base64” format when you send binary messages.⁴
- Reference: yen3’s blog – 有關 File I/O 的兩三事 (1) 不算開始的開始, (2) Binary File, (3) XML

4 Type

Type

Java has two different types.

- primitive type: int, char, double, balabala
- pointer type: class type(user-defined type)

We know that Java doesn’t have “pointers”

- Java has “new”, but no “delete” keyword.
- What is Garbage Collection?
- What is Java’s memory management mechanism? (It’s an important factor about Java’s program efficiency.)

³<http://java.sun.com/j2se/1.4.2/docs/api/java/io/package-tree.html>

⁴<http://en.wikipedia.org/wiki/Base64>

String Type

- `String` has build-in type's interface with pointer type's implementation.
- `String` vs. `StringBuilder`
- What does the language do in background?
- Java's `String` uses UTF-8 as default encoding.
- Let's talk about an example.

Type-Casting

- Java is strongly-typed programming language.
- `void*` vs. `java.lang.Object`
- In Java, every class inherits from `java.lang.Object`. It means we can cast an object to `java.lang.Object` type and cast again to other arbitrary class type .
- Compiler doesn't check the situation because it seems legal from the syntactical perspective.
- You have to check the situation by yourself.

5 Class and Object

Thinking about The Problems

- What is a "type"?
- What is a "object"?
- OO is claimed to be closer to human thoughts. However, is it true ?
- `Object` vs `java.lang.Object`
- Why doesn't Java support "Operator Overloading"? "Operator Overloading is syntax sugar. You don't know how efficient you cost in using."

Class and Object

A Class has ...

- data member (Data) – Record States
- member function(Method)

A Class has three modes

- `public` – interface
- `private` – implements
- `protected` – for some reasons XD

We have to pay attention to two keywords, but the slide doesn't mention.

- `static`
- `final`

Class

There are several kinds of classes.

- Normal Class – There’s nothing to say. XD
- Interface – Think about multiple-inheritance.
- anonymous Class – Think about the relationship about function pointer and call-back function.

Back to C — Function Pointer

- Von Neumann’s Model says program can be saved in memory.
- If we ruled the function’s interface, we can show difference functionality by replacing functions rely on function pointer.

For Example: C’ `stdlib.h` — `qsort()` prototype

```
void qsort(void* base, size_t n, size_t size,
           int (*cmp)(const void*, const void*));
           /* function pointer */
```

Back to C++ — Function Object

- The most difference between Function Object and Function Pointer is Function Object has own states(variable).
- Implementation: Class with Operator Overloading “()”
- If we implemented function object with template. It stars the Generic Programming’s first step.

For Example: C++ — Function Object

```
template<typename T> class Less{
public:
    bool operator()(const T& x, const T& y){ return x < y;}
}
```

λ Function (Lambda Function)

- base on *lambda*-expressions from Functional Programming,.
- no side-effect function.
- an unnamed function for temporary uses
- Haskell naïve support
- C++’s support: `Boost::lambda`

For Example: C++ — `Boost::lambda`

```
std::sort(v.begin(), v.end(),
          std::greater<iterator_traits(v.begin())::value_type>());
std::sort(v.begin(), v.end(), *_1 > *_2);
```

Call-Back Function — Anonymous Class with Interface

- What is “Call-Back Function”?
- How does JavaScript support the Call-Back Function?
- Class-Based vs Prototype-Based
- Java uses anonymous class and interface(Runnable) to support call-back property.

For Example: Java — anonymous class with interface

```
Thread newTask = new Thread(new Runnable(){
    public void run(){ System.out.println("Hello World!"); });
```

Interface and Abstract Class

- `interface` and `abstract` are keywords.
- Interface and abstract classes provide a more structured way to separate interface from implementation.
- In begin, we always have no idea to use them.
- It's show time to use “Refactoring”.
- We can extract to super class, abstract class, or interface from the same functionality classes.
- Button-Up Design vs Top-Down Design

Object-Oriented Class

- For efficient reason, C++ default member function mode is `non-virtual`, but Java is not.
- Please read the relational books to get more details about Java's Class.
- The feature is implemented by function pointer table.
- Think about the relation of RTTI and function pointer table.
- How to Design Class? Traditional OO Design vs eXtreme Programming⁵

6 Generic in Java (igonred)

Generic in Java

- Let's ignore the topic.
- We can discuss the topic if we have a basic knowledge for C++ Generic Programming.
- C++'s STL is made from some Functional Programming concepts.
- Java's Generic is made by Java OO method.
- In fact, Java's Generic is still a OO paradigm.
- You can play the paradigm using Haskell. It's more interesting than C++ and Java. XD

⁵We will discuss the issue next week.

7 Exception

Exception

- What is a “Exception”?
- “GOTO Considered Harmful.”⁶
- Think about the relation between “goto” and “exceptions”.
- We have to know program execute the exception block means low efficiency.
- But we couldn't ignore writing exception when using some library(ex: I/O, Socket, Thread)

8 Conclusion

Conclusion

- We go through Java from different parts, especially from C/C++.
- Java is a noun-kingdom XD XD. (What do you think about when seeing “`ArrayIndexOutOfBoundsException`” first time?)
- Please love “Eclipse IDE” when developing Java programs. XD
- Let's expect the interesting issue next week . XD

9 Reference

Reference — Website

- *Java SE Documentation* – <http://java.sun.com/javase/downloads/index.jsp>
- *Eclipse.org* – <http://www.eclipse.org>
- *C++ Boost Library* – <http://www.boost.org>
- *SGI STL* – <http://www.sgi.com/tech/stl/>
- *TIOBE Software: Tiobe Index* – <http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html>

Reference — Java

- *Thinking in Java 4/e*, Bruce Eckel, ISBN: 0131872486
- *The Java Programming Language 4/e*, Ken Arnold, James Gosling, David Holmes , ISBN: 0321349806
- *Effective Java 2/e*, Joshua Bloch, ISBN: 0321356683
- *Practical Java(TM) Programming Language Guide*, Peter Hagggar, ISBN: 0201616467
- *The Art of Java*, Herbert Schildt, James Holmes, ISBN: 0072229713
- *Java Network Programming 3/e*, Elliotte Rusty Harold, ISBN: 0596007213
- *Java Swing 2/e*, James Elliott, Robert Eckstein, Marc Loy, David Wood, Brian Cole, ISBN: 0596004087

⁶Edsger W. Dijkstra, Turing Award 1972

Reference — Object Oriented Analysis/ Design/ Programming

- *Object-Oriented Analysis and Design with Applications 3/e*, Grady Booch, Robert A. Maksimchuk, Michael W. Engel, Bobbi J. Young, ISBN: 020189551
- *Design Patterns: Elements of Reusable Object-Oriented Software*, Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides, ISBN: 0201663612
- *Refactoring: Improving the Design of Existing Code*, Martin Fowler, Kent Beck, John Brant, William Opdyke, don Roberts, ISBN: 0201485672
- *Refactroing to Pattens*, Joshua Kerievsky, ISBN: 0321213351
- *Agile Software Development, Principles, Patterns, and Practices*, Robert C. Martin

Reference — C/C++ and Other

- *Real World Haskell*, Bryan O’Sullivan, John Goerzen, Don Stewart, ISBN: 0596514980
- *The C++ Programming Language 3/e*, Bjarne Stroustrup, ISBN: 0201700735
- *The C Programming Language 2/e*, Brian W. Kernighan, Dennis M. Ritchie, ISBN: 0131193716
- *C++ Primer 4/e*, Stanley B. Lippman, Josée Lajoie, and Barbara E. Moo, ISBN: 0201721481
- *Effective C++ 3/e: 55 Specific Ways to Improve Your Programs and Designs*, Scott Meyers, ISBN: 0321334876
- *Generic Programming and the STL: Using and Extending the C++ Standard Template Library*, Matthew H. Austern, ISBN: 0201309564