More on Object Oriented Programming

Hsuan-Tien Lin

Dept. of CSIE, NTU

OOP Class, February 21, 2013

Done: Different Programming Paradigms

- imperative programming (compuation as commands for data manipulation)
  - procedure-oriented
- object-oriented (computation as interactions of "parts")
- functional (computation as math process)
- logical (computation as theorem proving)

Why do we introduce those paradigms?

程式輸入員v.s. 程式設計師

- 輸入員：basic language skill
- 設計師：good design skill + good language skill
  - what's the purpose of the program?
  - what's the specialty of the language?
  - what's the current need of the program?
  - what's the future need of the program?

—different answers call for different paradigm/language/style/...

設計師：
願意用現在的專業付出，來節省未來的時間和心力！

不要只想寫出

用後即丟的程式

Noodle-Oriented Programming

- whatever ingredients you put in, edible noodles are good noodles
- too salty? add more water
- no vegetables? get whatever is in your refrigerator
- spaghetti code: a program flow that looks like a bowl of spaghetti, i.e. twisted and tangled

NOP: generate whatever code that works for now

An Example of NOP

See `OOPLotteryV0.java`.

Applications of Noodle-Oriented Programming

Homework

Exam

Online Judge

讓人看不懂的(Virus, Internet Program....)

Final Project

折磨教授

From NOP to Procedure Oriented Programming

- organize the code
- identify the purpose of procedures (what a block of code can do)
- isolate (modularize) the procedures (as individual functions)
- reuse the procedures (by function calls)

You basically learned those in the C class.

An Example of POP

See `OOPLotteryV1.java`.

Object Oriented Programming 101-1

- group related data together in design

```c
/* C */
typedef struct{
  char dept[100];
  char ID[100];
  char name[100];
} Record;
```

```java
/* Java */
class Record{
  String dept;
  String ID;
  String name;
}
```

Object Oriented Programming 101-2

- use the struct/class to generate objects

```c
/* C */
Record r;
Record* rp=(Record*)malloc(sizeof(Record));
strcpy(r.dept, "CSIE");
strcpy(rp->name, "HTLIN");
free(rp);
```

```java
/* Java */
Record r = new Record();
r.dept = "CSIE";
r.name = "HTLIN";
```

Object Oriented Programming 101-3

- don't "do something on" the object; let the object "take some action"

```
1  /* Java */
2  PrintStream ps = System.out;
3  ps.println("CSIE");
4  String s = "a,b,c";
5  tokens = s.split(",");
```

From POP to Object Oriented Programming

- use a procedure (`strtok`) to manipulate some representation of data (`str`)

```
1 /* C */
2 char* str ="to␣be␣splitted.'';
3 char␣*␣p;
4 p␣=␣strtok(str,"␣");
5 while␣(p␣!=␣NULL){
6 ␣␣printf("%s\n",␣p)
7 ␣␣p␣=␣strtok(NULL,␣"␣");
8 }
```

- ask an object (str) to perform an action (`split`)

```
1 /* Java */
2 String str ="to␣be␣splitted.'';
3 String[]␣res␣=␣str.split("␣");
4 for(int␣i=0;i<res.length;i++)
5 ␣␣System.out.println(res[i]);
```

similar, but possibly easier if you can **think from objects**

An Example of OOP

See `OOPLotteryV2.java`.

From Noodle to Procedural to Object

- NOP: spaghetti code + (possibly spaghetti) data
  - You can write NOP with virtually ANY languages
  - Some easier to NOP (e.g. assembly), some harder
- POP: organized CODE + (possibly organized) data
  - using procedures as the basic module
    –maintain, reuse
  - action as separated procedures from data
    (do on the data)
  - C, Pascal
- OOP: organized DATA + organized code (ACTION) grouped together
  - using classes as the basic module
  - action are closely coupled with data
    (data do something)
  - Java, C++ (?), C#

From Noodle to Procedural to Object

- OOP: organized DATA + organized code (ACTION)
  - using classes as the basic module
  - action are closely coupled with data
    (data do something)
  - Java, C++ (?), C#
- You can write virtually any-OP with any language
- OO design: think in the OO way
- OO language: help (force) you to think in the OO way

An OO Trip with Java `String` (see: Sec 1.3)

```
1  import java.lang.*;
2
3  public class HelloWorld{
4    public static void main(String[] argv){
5      String s = "Hello World";
6      //imagine: s 'holds' all characters instead of pointing to H
7      //(note: an inaccurate description)
8
9      String t = "Hello world";
10     System.out.println(s.length());
11     //action for probing obj status
12
13     System.out.println(s.charAt(3));
14     //action for probing obj status
15   }
16 }
```

More OO Trip with Java `String`

```
1  import java.lang.*;
2
3  public class HelloWorld{
4    public static void main(String[] argv){
5      String s = "Hello World";
6      String t = "Hello world";
7
8      System.out.println(s.equalsIgnoreCase(t));
9      //action concerning one obj and the other
10
11     System.out.println(s.replace('o', 'a'));
12     //action for 'manipulating' the object
13     //(note: an inaccurate description)
14
15     System.out.println(s.concat(t));
16     System.out.println(s.concat(" lala"));
17     System.out.println(s + " lala");
18     //action for 'manipulating' the object
19     //(note: an inaccurate description)
20   }
21 }
```

Three Levels of OO

- Object-Oriented Analysis (OOA):
  what the system does
  - from (customer/system) needs to software models
  - an important topic in Software Engineering class
- Object-Oriented Design (OOD):
  how the system does it
  - from software model to class diagrams
  - an important topic in Design Pattern class, some in this class
- Object-Oriented Programming (OOP):
  how to implement such a system
  - from class diagrams to class implementations
  - learn from this class

this class: called software design but actually **programming**

Three Levels of OO

- Object-Oriented Analysis (OOA):
  what the system does
- Object-Oriented Design (OOD):
  how the system does it
- Object-Oriented Programming (OOP):
  how to implement such a system

not necessarily separate levels
—**connect the dots** from your different classes