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.

Object Oriented Programming: A New Way of Modularizing

- a running computer in front of you: a computing “component”
- an (interactive) program: a computing “component”
- program A interacting with program B: another computing “component”
- sub-program A.1 interacting with sub-program A.2: two mini-computing “components”

What does a computing “component” need?

Computing Component

- code: input $\rightarrow$ output
- associated data

Google Search Engine

- code: "search words" $\rightarrow$ "ranked list"
- associated data: user profile, web snapshot, etc

String Processor

- code: "string processing request" $\rightarrow$ "processed result"
- associated data: string content

Object-Oriented Programming: Program by Designing (Mini-)Computing Components

Object Oriented Programming

Computing Component

- associated data
- code: input $\rightarrow$ output

Object

- properties
- actions (methods): message $\rightarrow$ return value (and status change)

Object-Oriented Programming: Program by Letting Objects Interact with Each Other

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 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

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
- Object-Oriented Programming (OOP):
  how to implement such a system
  - from class diagrams to class implementations
  - learn from this class

this class: just **fundamental** OOP