Advanced Exercise on Computer and Information Science B

A survey of Object-Oriented Programming

Purpose of the Lecture

- Object-Oriented Software Design
- Object-Oriented Programming
- Required Technology
  - Experience of Programming
  - Basic skills of Programming(e.g., loop and branch)
  - Understanding Object-Oriented concept

Assumption/Related Lecture

- Assumption
  - Basic Java Programming
    - Understanding basic syntax and semantics
    - Do not need object-oriented style
- Related Lecture
  - Advanced Programming
    - Java Language
  - Software Engineering2
    - Put in practice what we have learned in SE2

Evaluation

- Object-Oriented Programming(50%)
  - Personal Exercises
- Object-Oriented Software Design(50%)
  - Output implemented by group exercise
- NO examinations

Exercise Environment

- Linux (Unix)
  - No more Windows!
- Environment
  - 1st half
    - Emacs + Javac/Java
  - 2nd half
    - Eclipse

Preparation

- Study beforehand
  - This lecture is not so easy for you to listen to normally
- Not only brank spaces
  - You should do it from scratch
- This is not a kind of EASY lectures

Mechanisms of computer

- Turing machine
  - The theory of computations
  - Infinite memory cells and the move of the head
- Von Neumman architecture
  - CPU
  - Register
  - Memory, HDD

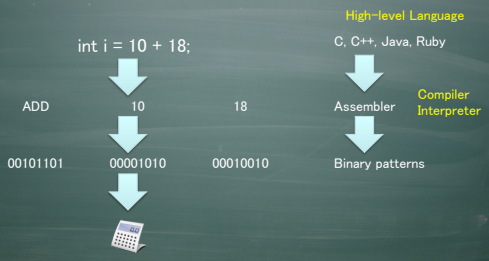
Executions of a program

1. Load a program from HDD to memory
2. Load an address of a program stored in main memory to a register
3. Load data specified by the value of a register to another register
4. The values are computed by CPU
5. Store the result to a register
6. Store the result to main memory
7. Store the result to HDD if necessary
8. Repeat from 1 to 7
9. Exit and release the main memory

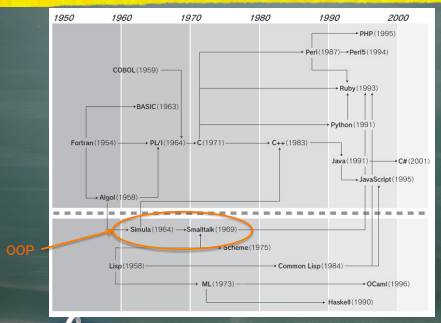
Executions of a program

- All operations are defined by binary patterns
  - Difficult for human being to understand
- Assembly Language
  - One operation corresponds to one binary pattern
  - Load, Jump (Goto in C language)
- High-level programming language
  - Structured programming

From programming to its execution



The history of programming language



Why do we need Object-Oriented Programming?

- Large-scaled software development
  - Resuse the resources
  - Encapsulation
  - Modularity
  - Divide specifications and its implementations

History of OOP

- The concept of OOP was in the second half of 1960's, especially in Simula.
  - 1970's → Smalltalk
  - 1980's → C++, Objective-C, Eiffel
  - Each one is programming language

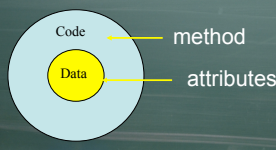
The features of Object-oriented

- Object = Data + Operations
- Message Passing
- Features
  - Encapsulation
  - Inheritance
  - Polymorphism

Object=Data+Code

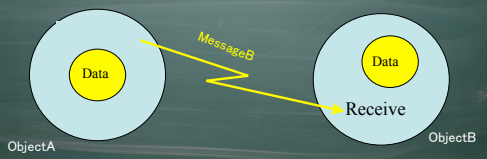
- Object
  - Data
  - Code that operates data

} a real entity



Message Passing

- ObjectA sends a message to ObjectB in order to progress a operation
  - Basically, the method accesses certain data included ObjectB.



Object-Oriented Concept and Language

- Ideally, everything is Object
  - Smalltalk is pure OOP Language
- But, Java introduces:
  - New keyword (it's not message)
  - Primitive type
  - Interface
    - Because of multiple inheritance

Java Language

- History of Java

Release Name	Virsion
JDK1.1x/JRE1.1x	1.1
Java2 Platform, Standard Edition, v 1.2	1.2
Java2 Platform, Standard Edition, v 1.3	1.3
Java2 Platform, Standard Edition, v 1.4	1.4
Java2 Platform, Standard Edition, v 5.0	1.5
Java Platform, Standard Edntion 6	1.6
Java Platform, Standard Edntion 7	1.7

Java Programming

- Class
  - Data
    - Member variables
  - Behavior
    - Method
- Instance
  - new ClassName

An example of Java program

```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello World");
    }
}
```

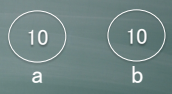
HelloWorld.java

```
int main(int argc, char *argv[]) {
    printf("Hello World");
    return 0;
}
```

helloworld.c

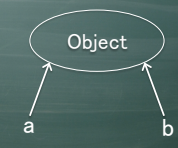
Primitive vs Object

```
int a = 10;
int b = a;
```



Note! comparative operator

```
Object a = new Object();
Object b = new Object();
Object b = a;
```



Difference of extend and implements

- Extends
  - Means extension
  - Can refer member variables and methods in parent class
- Implements
  - Means implementation
  - Has responsibility to implement methods defined in the interface

Java Environment/Compile and Execution

- JAVA_HOME
  - echo $JAVA_HOME
- Compile
  - javac HelloWorld.java
    - HelloWorld.class is created
- Execution
  - java HelloWorld
    - DON'T need .class
- CLASSPATH
  - is required to compile and execute
- CLASSPATH configuration
  - Define by Enviromnent
  - Define at runtime(-cp path)
    - java -cp somewhere HelloWorld

ClassNotFound if configuration is not completed

```
Exception in thread "main" java.lang.NoClassDefFoundError: HelloWorld
Caused by: java.lang.ClassNotFoundException: HelloWorld
        at java.net.URLClassLoader$1.run(URLClassLoader.java:202)
        at java.security.AccessController.doPrivileged(Native Method)
        at java.net.URLClassLoader.findClass(URLClassLoader.java:190)
        at java.lang.ClassLoader.loadClass(ClassLoader.java:307)
        at sun.misc.Launcher$AppClassLoader.loadClass(Launcher.java:301)
        at java.lang.ClassLoader.loadClass(ClassLoader.java:248)
Could not find the main class: HelloWorld.  Program will exit.
```