Computer System Design & Application

计算机系统设计与应用A

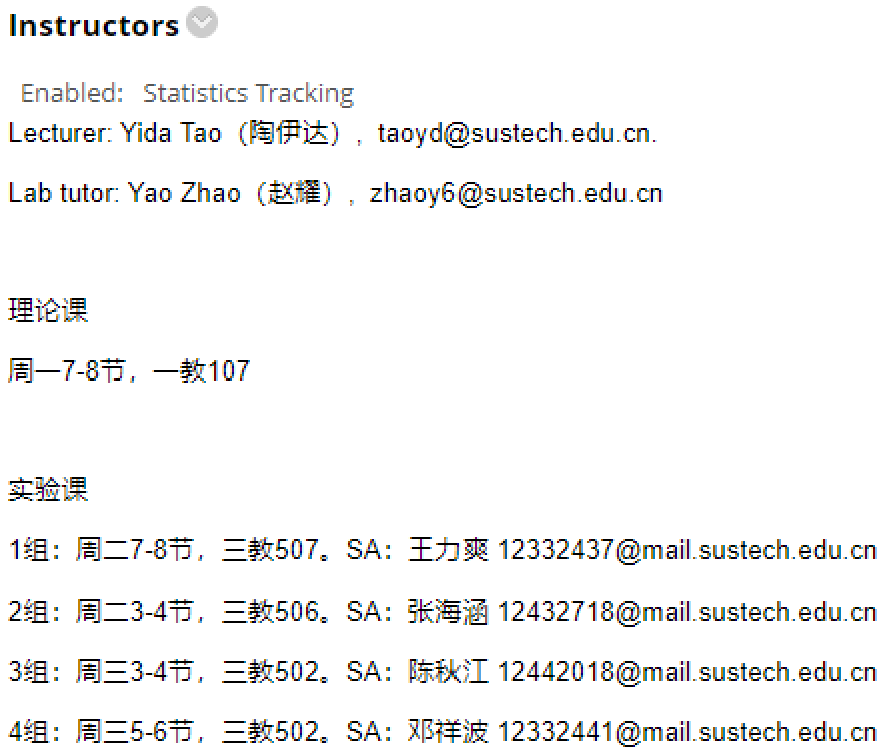
陶伊达 （TAO Yida）

taoyd@sustech.edu.cn

Course Logistics

- Course website: Blackboard https://bb.sustech.edu.cn/
- Slides and other resources will all be uploaded here.

- Office hours: Wednesday 14:00 – 16:00 pm, CoE South Building, 411B



Instructors

Enabled: Statistics Tracking

Lecturer: Yida Tao（陶伊达）, taoyd@sustech.edu.cn.

Lab tutor: Yao Zhao（赵耀）, zhaoy6@sustech.edu.cn

理论课

周一7-8节，一教107

实验课

1组：周二7-8节，三教507。SA：王力爽 12332437@mail.sustech.edu.cn

2组：周二3-4节，三教506。SA：张海涵 12432718@mail.sustech.edu.cn

3组：周三3-4节，三教502。SA：陈秋江 12442018@mail.sustech.edu.cn

4组：周三5-6节，三教502。SA：邓祥波 12332441@mail.sustech.edu.cn

Topics covered

Principles

- OOP, Functional programming
- Software design concepts
- JVM architecture
- Testing

……

Utilities

- Exception handling
- Generic collections
- Lambdas & Streams
- Annotation
- Reflection

……

Functionalities

- File I/O
- Networking
- Multithreading
- Web development
- GUI

……

Applications

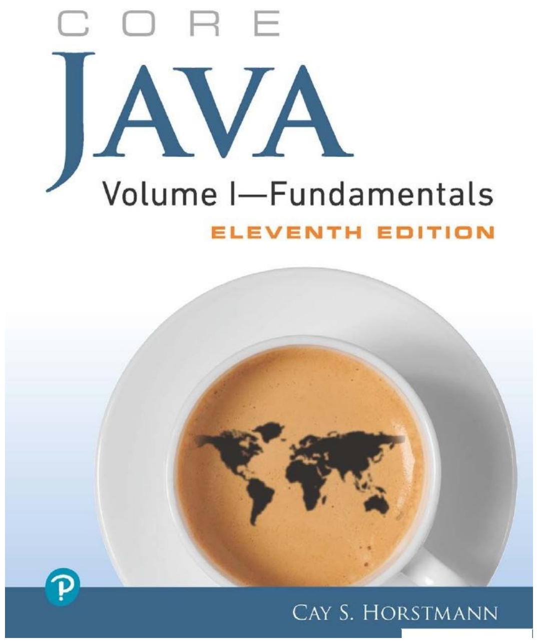
- Text scraping and processing
- Data analytics and visualization
- C/S applications
- Web applications & services

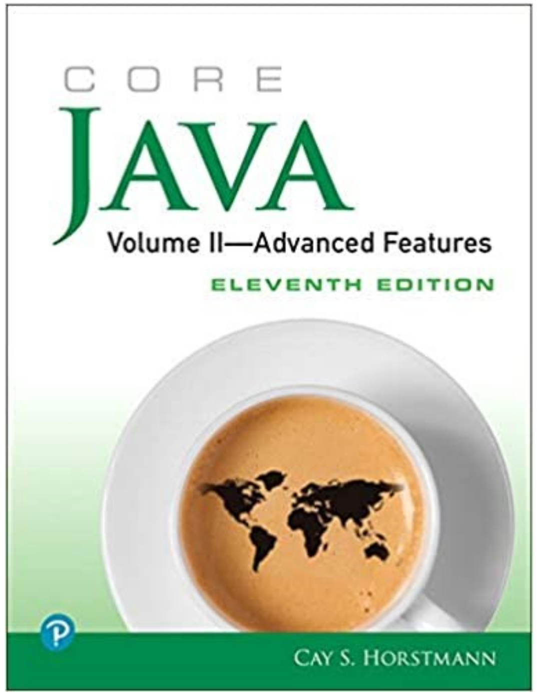
……

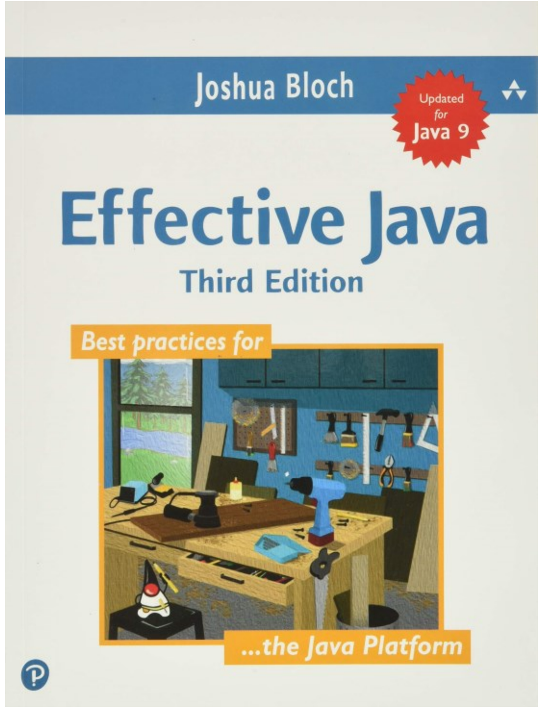
Syllabus
(Negotiable)

- Lecture 1: Computing overview, JVM, OOP, Design Principles
- Lecture 2: Generics, ADT, Collections
- Lecture 3: Functional programming, Lambda
- Lecture 4: Java 8 Stream API
- Lecture 5: I/O Streams, Encoding
- Lecture 6: Serialization, File I/O, Exception Handling
- Lecture 7: Concurrency, Multithreading
- Lecture 8: Network Programming
- Lecture 9: Reflection, Annotation
- Lecture 10: GUI Intro, JavaFX
- Lecture 11: Java EE, Servlet
- Lecture 12: The Spring Framework
- Lecture 13: Spring Boot
- Lecture 14: Logging, JUnit Testing
- Lecture 15: Project Presentation, Course Review

Reference Books







Core Java Volume I II
Cay S. Horstmann

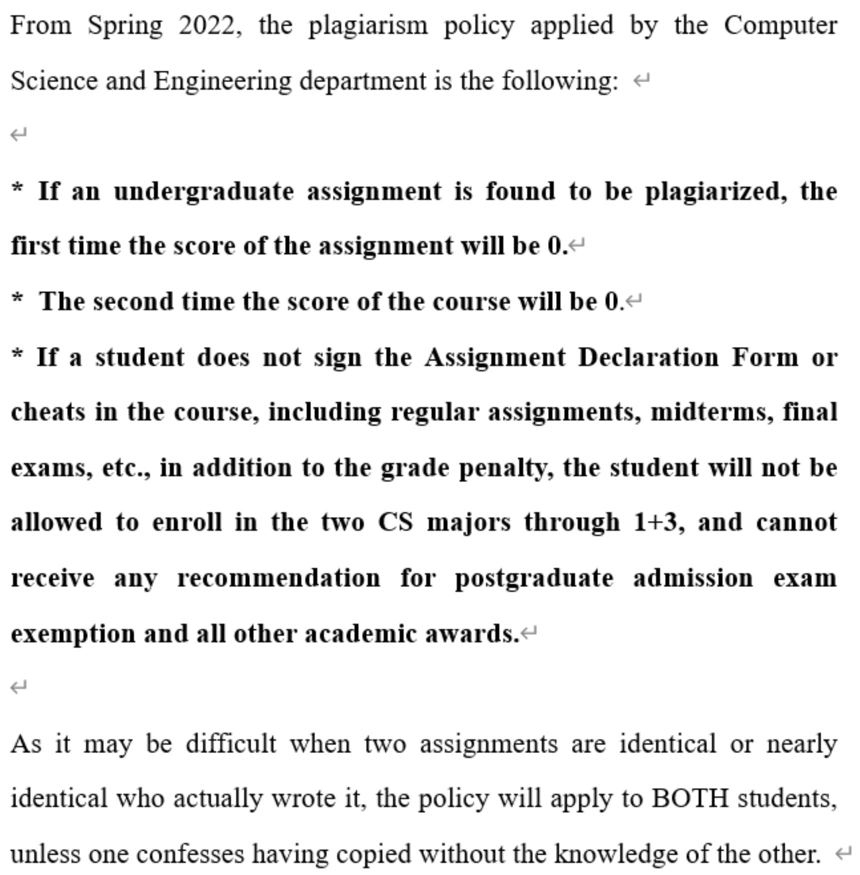
Effective Java
Joshua Bloch

Coursework & Grading Policy

	Score	Description
Assignments	25%	2 assignments Assignment 1: release at week 4 and due at week 7 Assignment 2: release at week 8 and due at week 11
Project	20%	Released at around week 10 Team: 2 people +0.5 for submitting the final project at week 15 +1 (max) for presenting at week 16 lecture
Labs	15%	Attendance Lab practices (+0.1 for finishing onsite. max +1)
Quiz	10%	Quizzes, exercises, participation during lectures
Final Exam	30%	Close-book (Two pieces of A4 cheat sheets allowed) No electronic device

Labs start from the 1st week!

Academic Integrity



From Spring 2022, the plagiarism policy applied by the Computer Science and Engineering department is the following: ↵

↵

*** If an undergraduate assignment is found to be plagiarized, the first time the score of the assignment will be 0.**↵

*** The second time the score of the course will be 0.**↵

*** If a student does not sign the Assignment Declaration Form or cheats in the course, including regular assignments, midterms, final exams, etc., in addition to the grade penalty, the student will not be allowed to enroll in the two CS majors through 1+3, and cannot receive any recommendation for postgraduate admission exam exemption and all other academic awards.**↵

↵

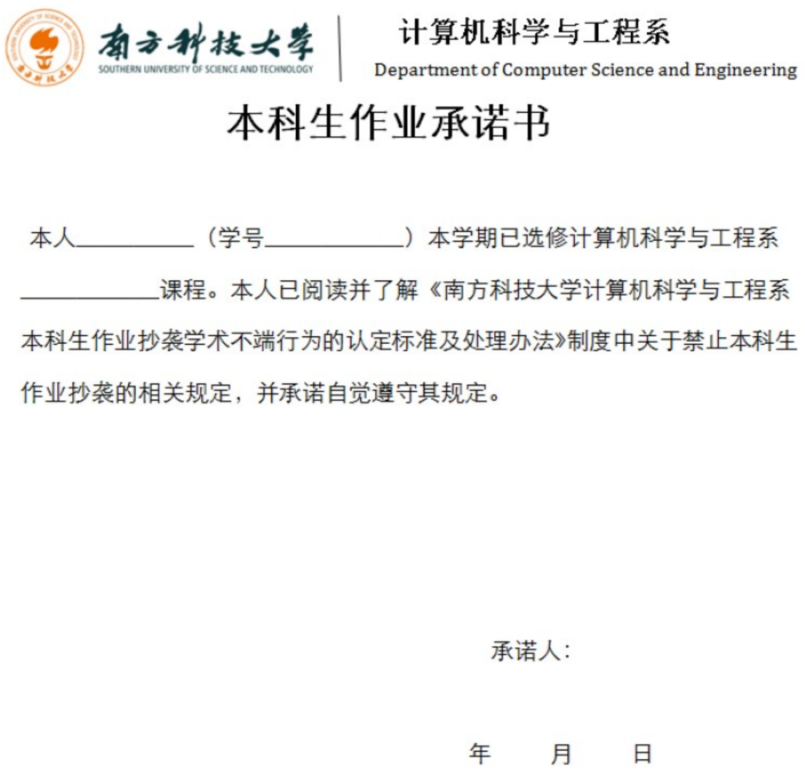
As it may be difficult when two assignments are identical or nearly identical who actually wrote it, the policy will apply to BOTH students, unless one confesses having copied without the knowledge of the other. ↵

- It's OK to work on an assignment with a friend, and think together about the program structure, share ideas and even the global logic. At the time of actually writing the code, you should write it alone.
- It's OK to use in an assignment a piece of code found on the web, as long as you indicate in a comment where it was found and don't claim it as your own work.
- It's OK to help friends debug their programs (you'll probably learn a lot yourself by doing so).
- It's OK to show your code to friends to explain the logic, as long as the friends write their code on their own later.

- It's NOT OK to take the code of a friend, make a few cosmetic changes (comments, some variable names) and pass it as your own work.

Academic Integrity

Please submit the form before the end of the course selection & drop period!



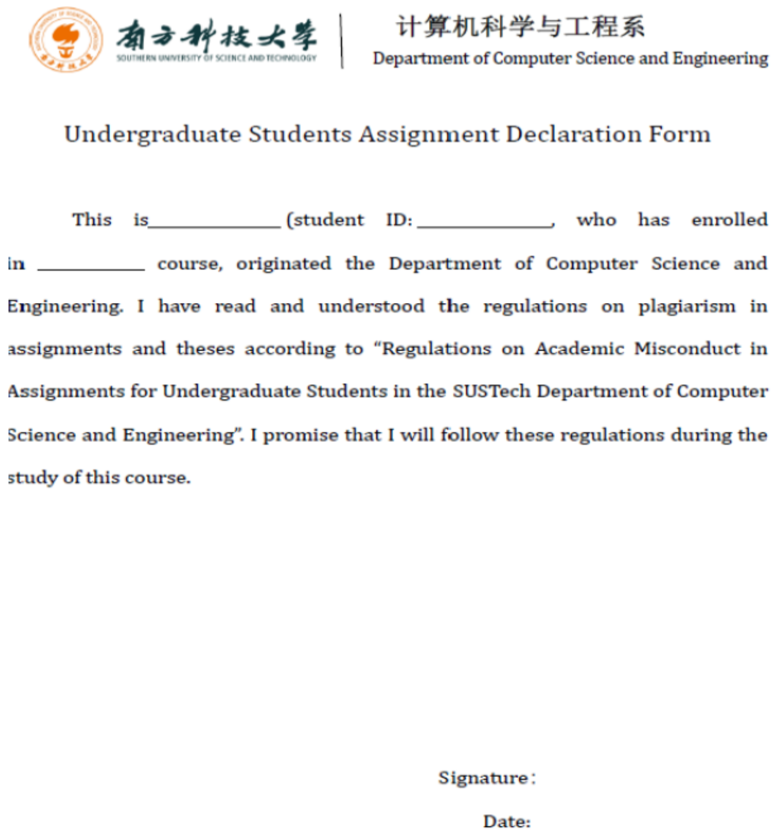
南方科技大学 SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY | 计算机科学与工程系 Department of Computer Science and Engineering

本科生作业承诺书

本人__________（学号___________）本学期已选修计算机科学与工程系___________课程。本人已阅读并了解《南方科技大学计算机科学与工程系本科生作业抄袭学术不端行为的认定标准及处理办法》制度中关于禁止本科生作业抄袭的相关规定，并承诺自觉遵守其规定。

承诺人：

年　　月　　日



南方科技大学 SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY | 计算机科学与工程系 Department of Computer Science and Engineering

Undergraduate Students Assignment Declaration Form

This is___________(student ID: ___________, who has enrolled in _________ course, originated the Department of Computer Science and Engineering. I have read and understood the regulations on plagiarism in assignments and theses according to “Regulations on Academic Misconduct in Assignments for Undergraduate Students in the SUSTech Department of Computer Science and Engineering”. I promise that I will follow these regulations during the study of this course.

Signature：

Date:

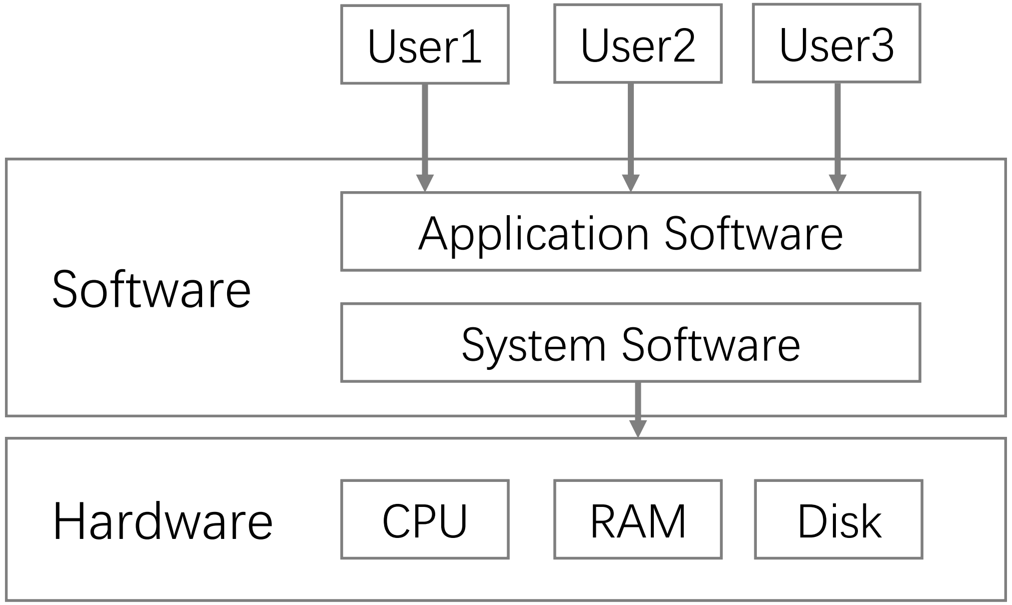
Lecture 1

- Course introduction
- Computer system & programs
- Java review and JVM
- Software design principles & OOP concepts

Computer System

- Hardware
  - The physical parts: CPU, keyboard, disks
- Software
  - System software: a set of programs that control & manage the operations of hardware, e.g., OS
  - Application software: a set of programs for end users to perform specific tasks, e.g., browser, media player

What is a program?



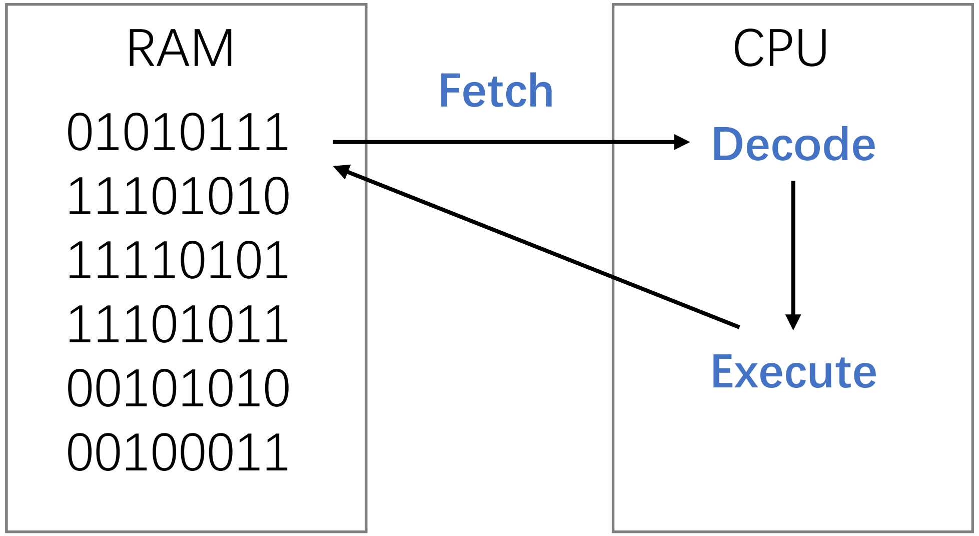
Programs

- A sequence of instructions that specifies how to perform a computation

Fetch-Decode-Execute Cycle

- **Fetch**: Get the next instruction from memory
- **Decode**: Interpret the instruction
- **Execute**: Pass the decoded info as a sequence of control signals to relevant CPU units to perform the action

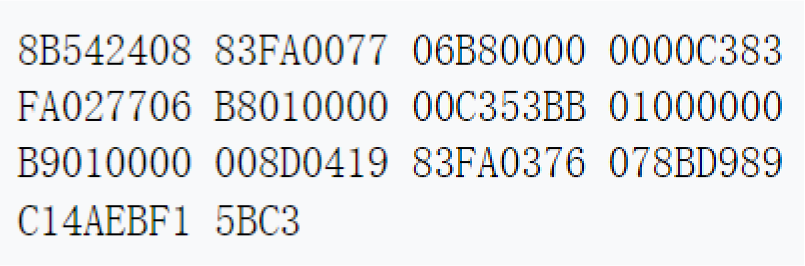
The fetch-execute cycle was first proposed by **John von Neumann**, who is famous for the **Von Neumann architecture**, which is being followed by most computers today



Programs

- A sequence of instructions that specifies how to perform a computation

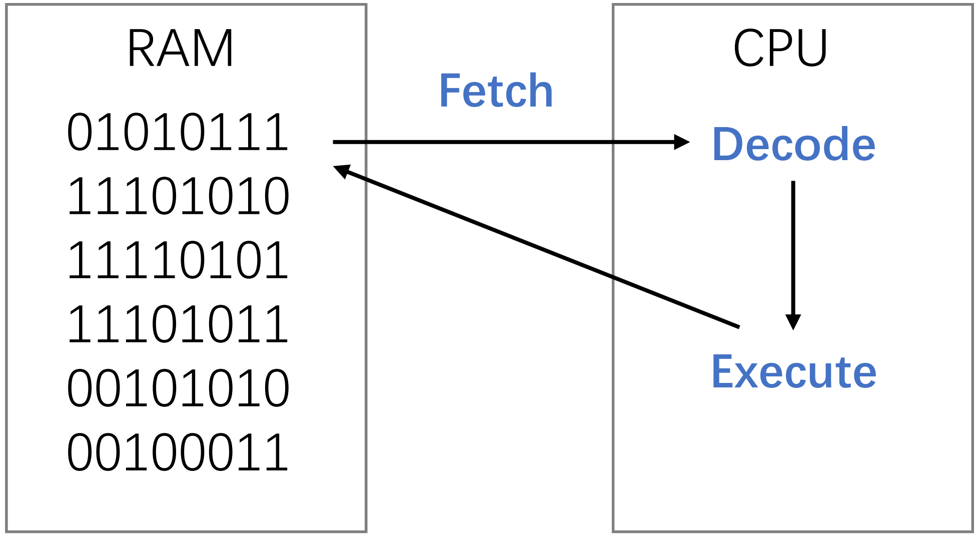
☹ **Machine-language instructions are hard to read & write for human.**



```
8B542408 83FA0077 06B80000 0000C383
FA027706 B8010000 00C353BB 01000000
B9010000 008D0419 83FA0376 078BD989
C14AEBF1 5BC3
```

A function in hexadecimal (十六进制) to calculate Fibonacci number

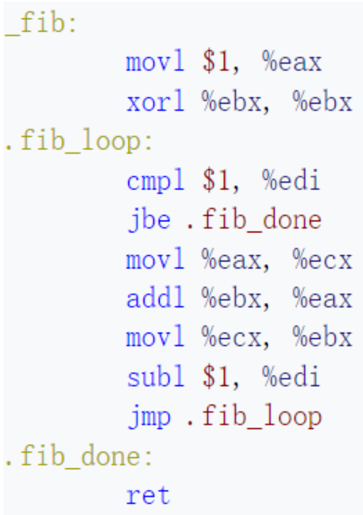
Source: https://en.wikipedia.org/wiki/Low-level_programming_language



Programs

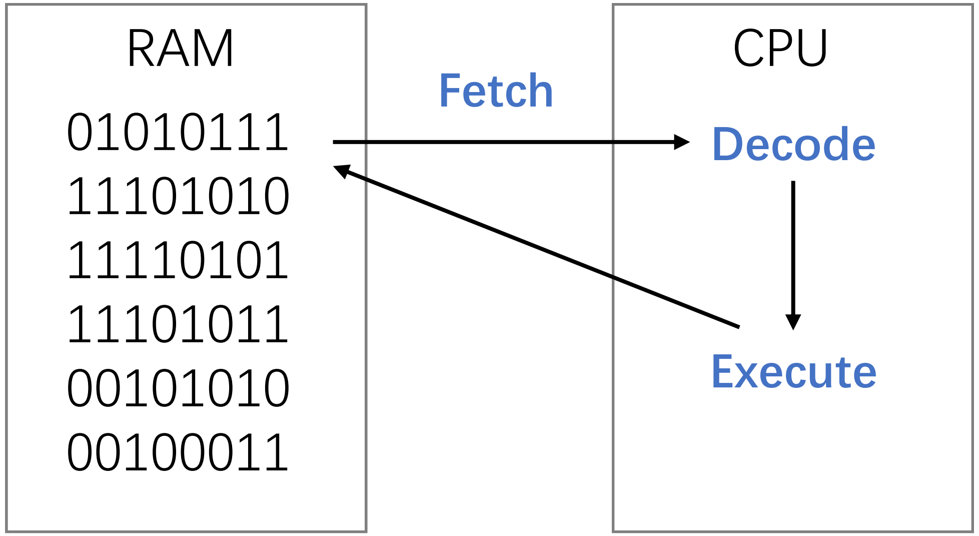
- A sequence of instructions that specifies how to perform a computation

Low-level language provides a level of abstraction on top of machine code



```
_fib:
        movl $1, %eax
        xorl %ebx, %ebx
.fib_loop:
        cmpl $1, %edi
        jbe .fib_done
        movl %eax, %ecx
        addl %ebx, %eax
        movl %ecx, %ebx
        subl $1, %edi
        jmp .fib_loop
.fib_done:
        ret
```

A function in assembly (汇编) to calculate Fibonacci number

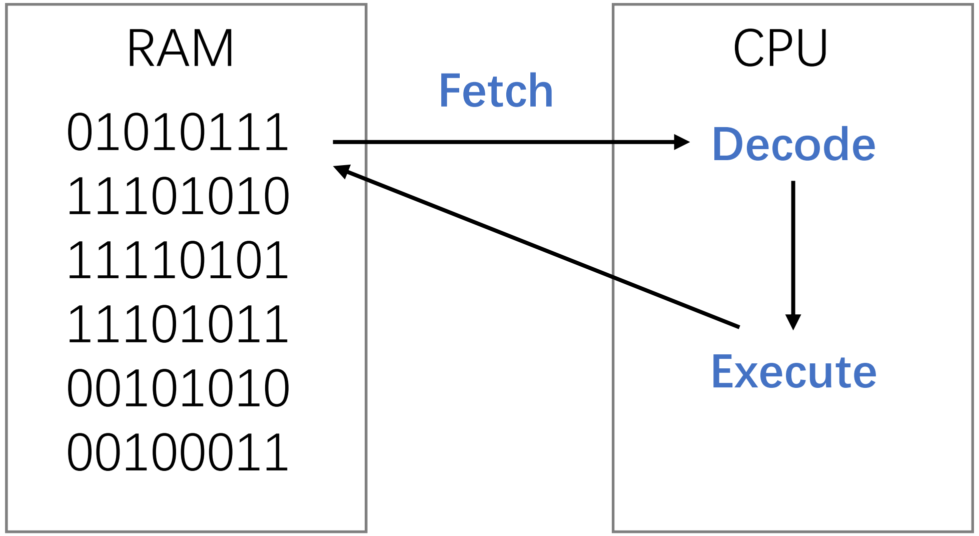


Source: https://en.wikipedia.org/wiki/Low-level_programming_language

Programs

- A sequence of instructions that specifies how to perform a computation

Low-level language provides a level of abstraction on top of machine code



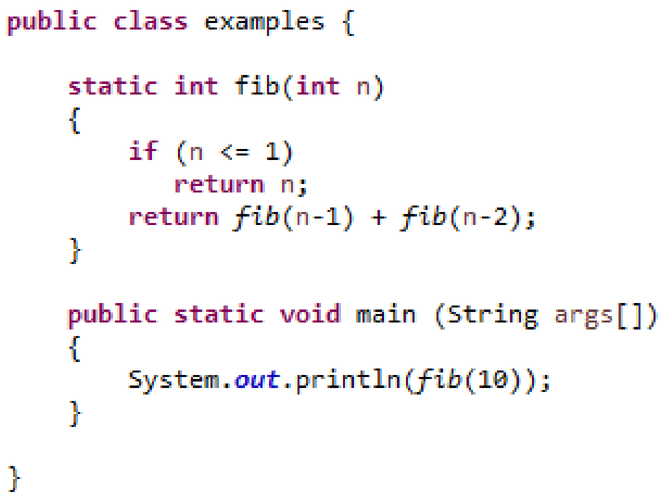
A video game written in assembly

Source: https://en.wikipedia.org/wiki/Prince_of_Persia_(1989_video_game)

Programs

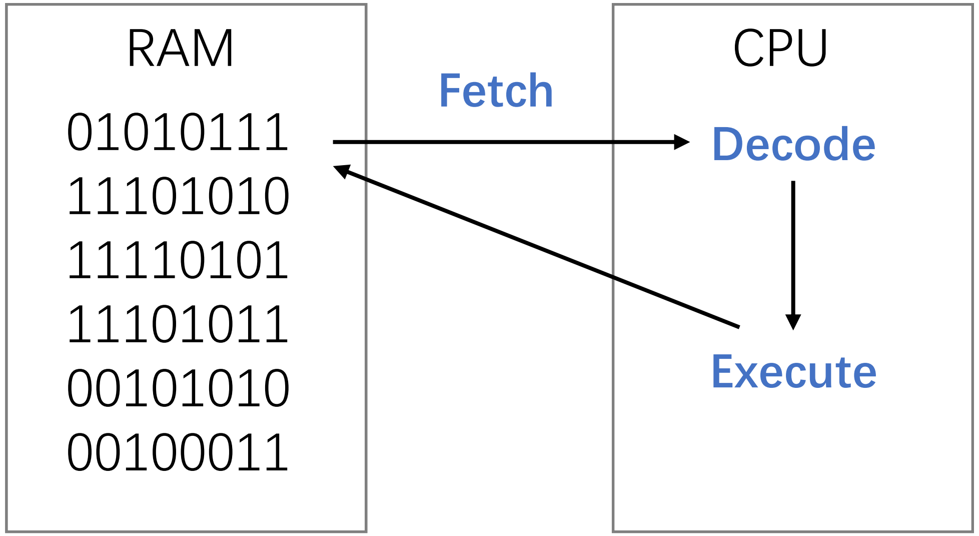
- A sequence of instructions that specifies how to perform a computation

High-level language (e.g., C++, Java, Python, etc.) provides stronger abstraction and resembles more of natural language



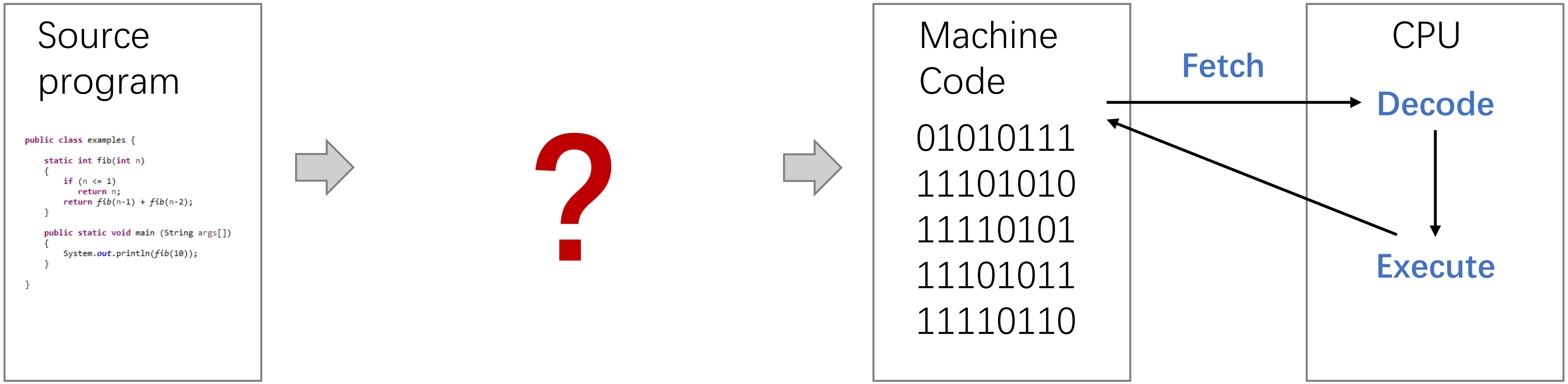
```
public class examples {

    static int fib(int n)
    {
        if (n <= 1)
           return n;
        return fib(n-1) + fib(n-2);
    }

    public static void main (String args[])
    {
        System.out.println(fib(10));
    }

}
```

A function in Java to calculate Fibonacci number



Programs

- A sequence of instructions that specifies how to perform a computation



```
public class examples {

    static int fib(int n)
    {
        if (n <= 1)
           return n;
        return fib(n-1) + fib(n-2);
    }

    public static void main (String args[])
    {
        System.out.println(fib(10));
    }

}
```

```
01010111
11101010
11110101
11101011
11110110
```

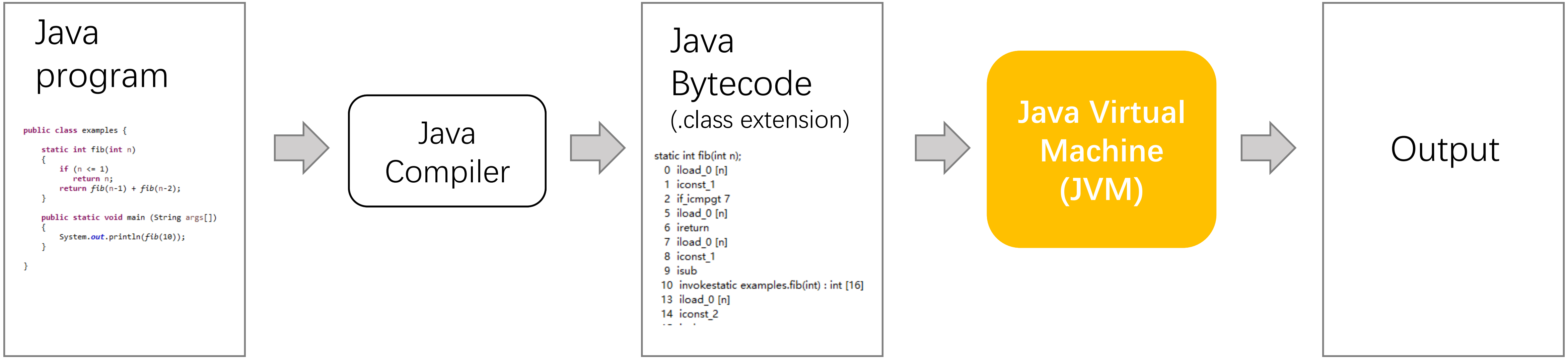
CS202. Computer Organization

Lecture 1

- Course introduction
- Computer system & programs
- Java review and JVM
- Software design principles & OOP concepts

How is a Java program executed?

- Same principle: high-level source → low-level/machine code

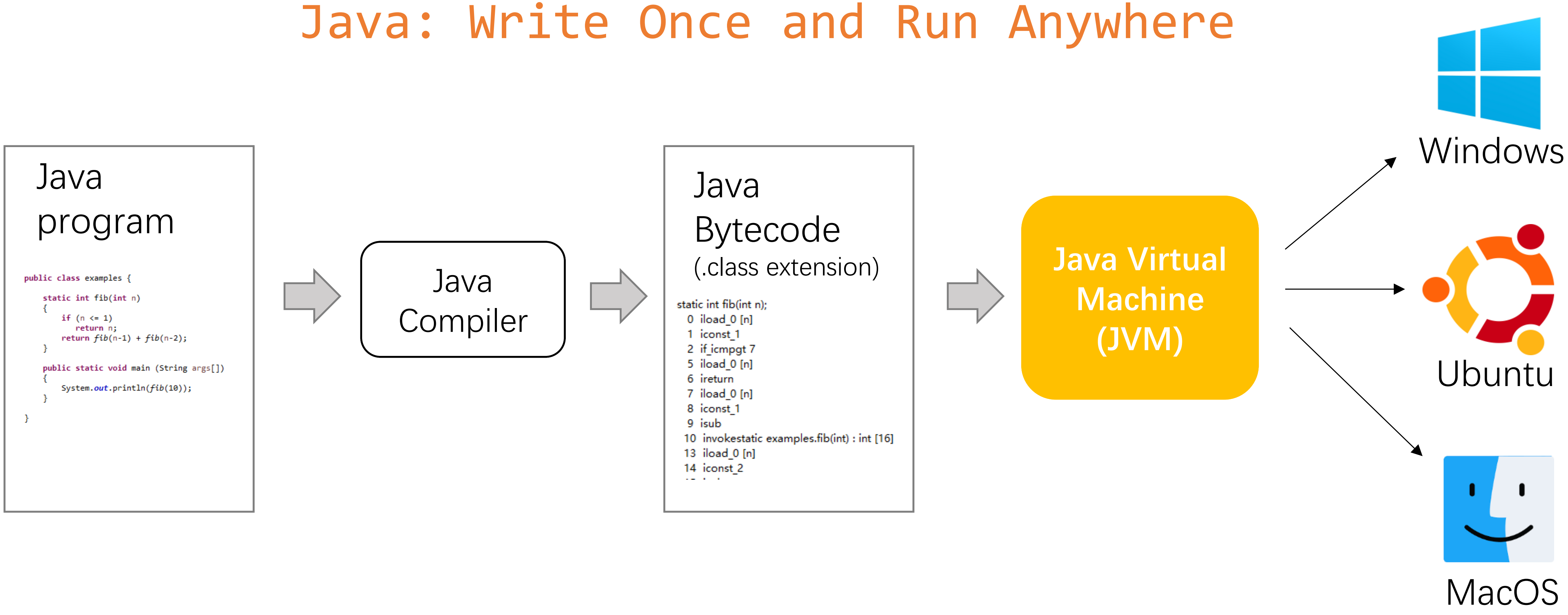


```java
public class examples {

    static int fib(int n)
    {
        if (n <= 1)
            return n;
        return fib(n-1) + fib(n-2);
    }

    public static void main (String args[])
    {
        System.out.println(fib(10));
    }

}
```

```
static int fib(int n);
   0  iload_0 [n]
   1  iconst_1
   2  if_icmpgt 7
   5  iload_0 [n]
   6  ireturn
   7  iload_0 [n]
   8  iconst_1
   9  isub
  10  invokestatic examples.fib(int) : int [16]
  13  iload_0 [n]
  14  iconst_2
```

Java Virtual Machine (JVM)

Java: Write Once and Run Anywhere



Java program

```java
public class examples {

    static int fib(int n)
    {
        if (n <= 1)
           return n;
        return fib(n-1) + fib(n-2);
    }

    public static void main (String args[])
    {
        System.out.println(fib(10));
    }

}
```

Java Compiler

Java Bytecode

(.class extension)

```
static int fib(int n);
   0  iload_0 [n]
   1  iconst_1
   2  if_icmpgt 7
   5  iload_0 [n]
   6  ireturn
   7  iload_0 [n]
   8  iconst_1
   9  isub
  10  invokestatic examples.fib(int) : int [16]
  13  iload_0 [n]
  14  iconst_2
```

Java Virtual Machine (JVM)

Windows

Ubuntu

MacOS

.class: Platform Independence

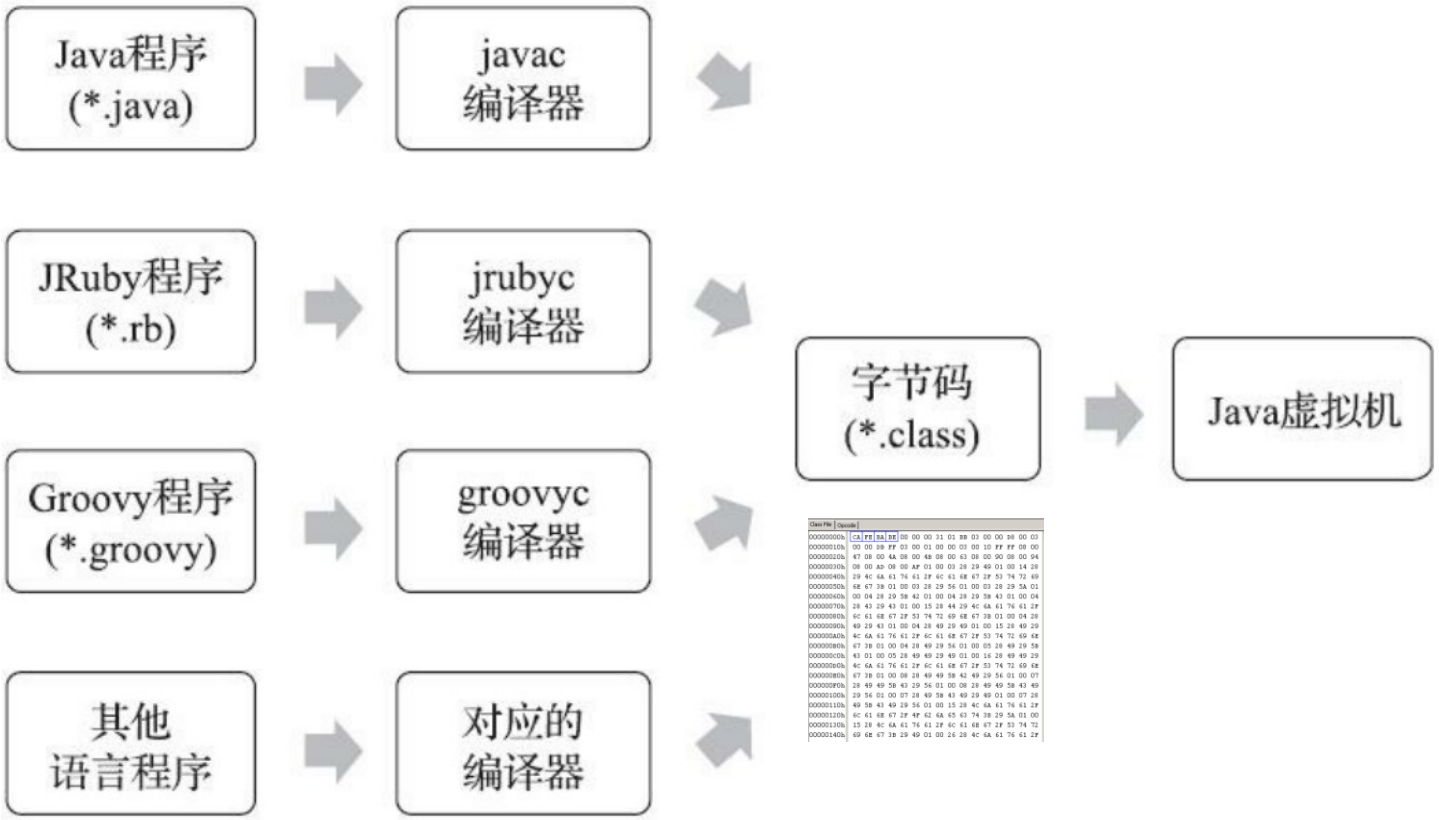
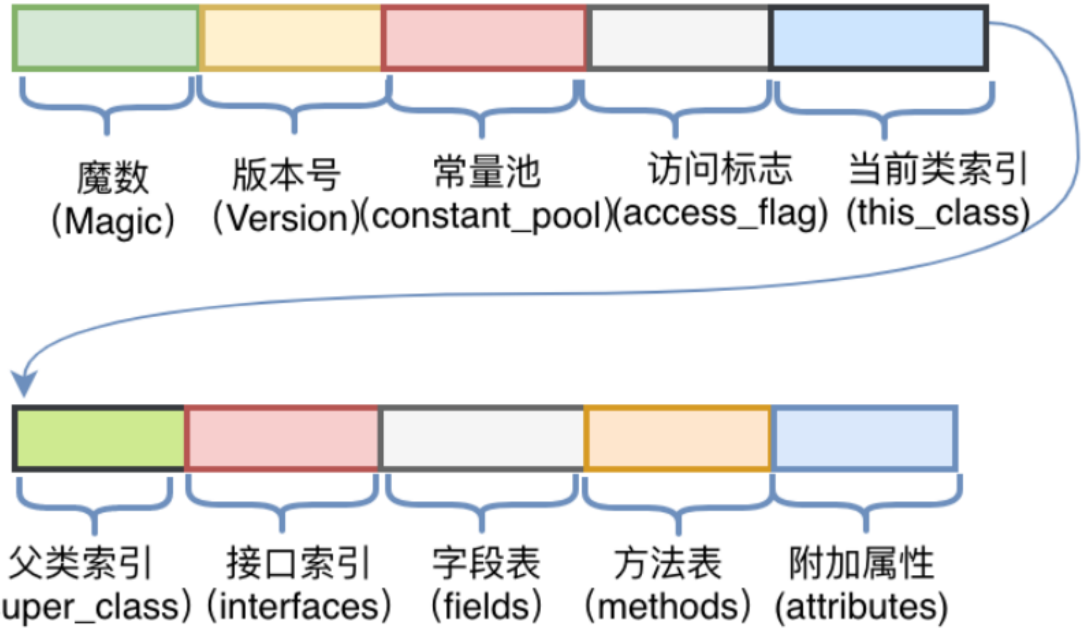


Image source: 《深入理解Java虚拟机》第三版，周志明

.class file format



魔数
(Magic)

版本号
(Version)

常量池
(constant_pool)

访问标志
(access_flag)

当前类索引
(this_class)

父类索引
(super_class)

接口索引
(interfaces)

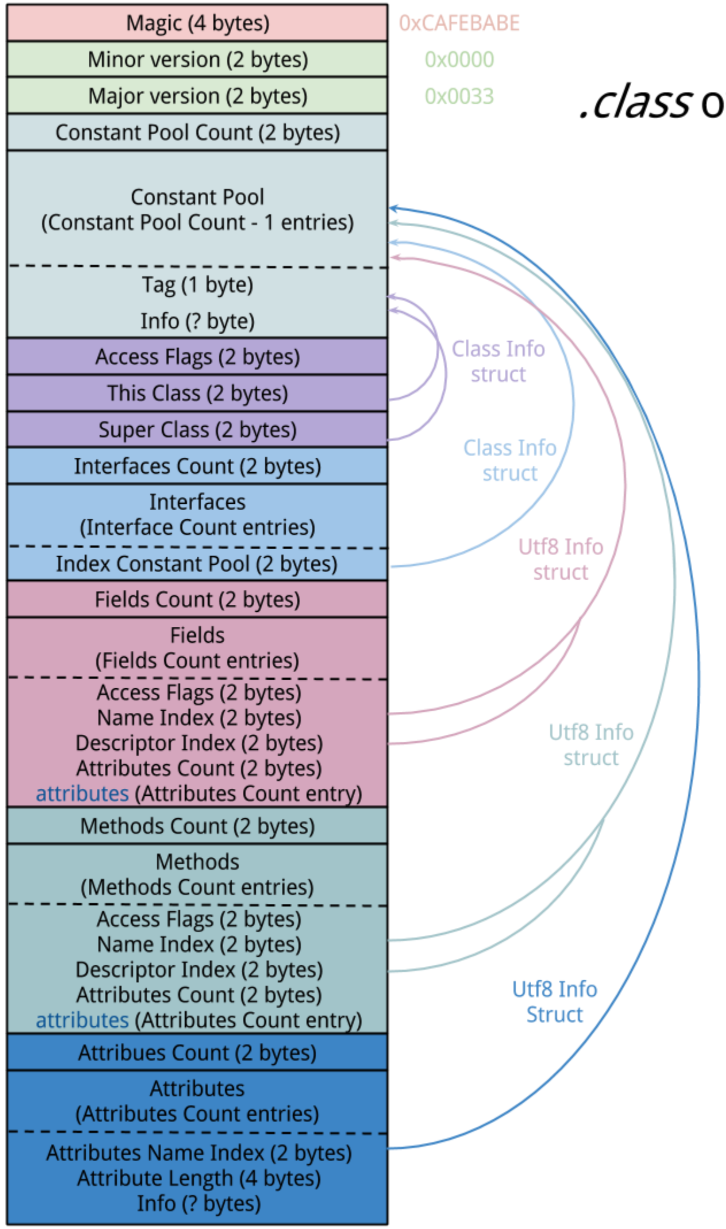
字段表
(fields)

方法表
(methods)

附加属性
(attributes)

https://github.com/likuisuper/Java-Notes/

.class overview



Magic (4 bytes) 0xCAFEBABE

Minor version (2 bytes) 0x0000

Major version (2 bytes) 0x0033

Constant Pool Count (2 bytes)

Constant Pool
(Constant Pool Count - 1 entries)

Tag (1 byte)
Info (? byte)

Access Flags (2 bytes)

This Class (2 bytes)

Super Class (2 bytes)

Interfaces Count (2 bytes)

Interfaces
(Interface Count entries)

Index Constant Pool (2 bytes)

Fields Count (2 bytes)

Fields
(Fields Count entries)

Access Flags (2 bytes)
Name Index (2 bytes)
Descriptor Index (2 bytes)
Attributes Count (2 bytes)
attributes (Attributes Count entry)

Methods Count (2 bytes)

Methods
(Methods Count entries)

Access Flags (2 bytes)
Name Index (2 bytes)
Descriptor Index (2 bytes)
Attributes Count (2 bytes)
attributes (Attributes Count entry)

Attribues Count (2 bytes)

Attributes
(Attributes Count entries)

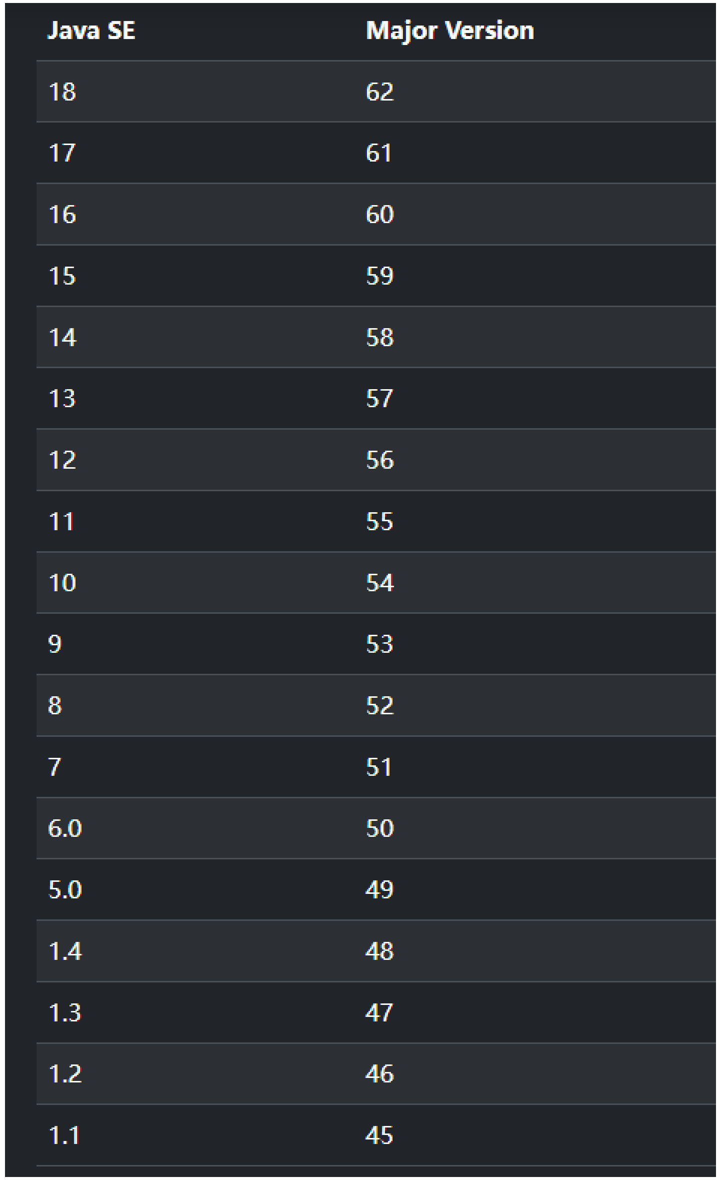
Attributes Name Index (2 bytes)
Attribute Length (4 bytes)
Info (? bytes)

https://blog.lse.epita.fr/2014/04/28/0xcafebabe-java-class-file-format-an-overview.html

Version Number

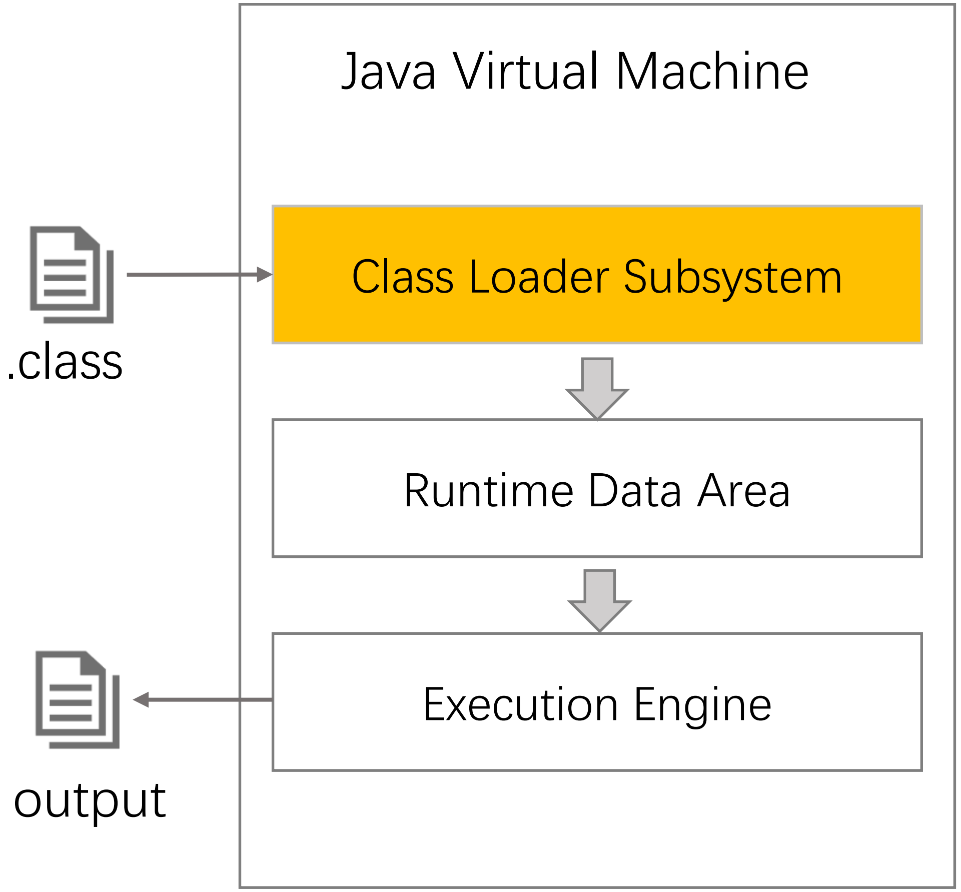
Java is backward compatible: you can run Java 15 compiled jar on JRE 16 but not vice versa.

`java.lang.UnsupportedClassVersionError (version 60.0)` happens because of a higher JDK (JDK 16) during compile time and lower JDK during runtime.



Java SE	Major Version
18	62
17	61
16	60
15	59
14	58
13	57
12	56
11	55
10	54
9	53
8	52
7	51
6.0	50
5.0	49
1.4	48
1.3	47
1.2	46
1.1	45

Java Virtual Machine (JVM)

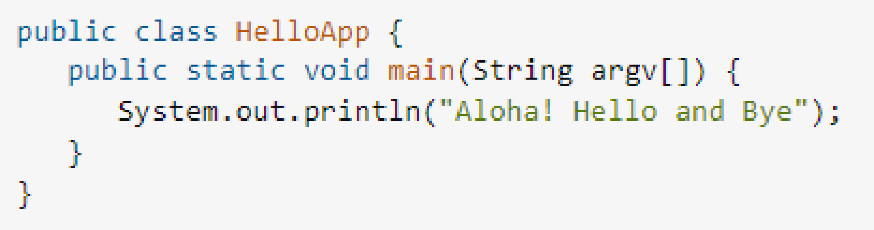


Class Loader

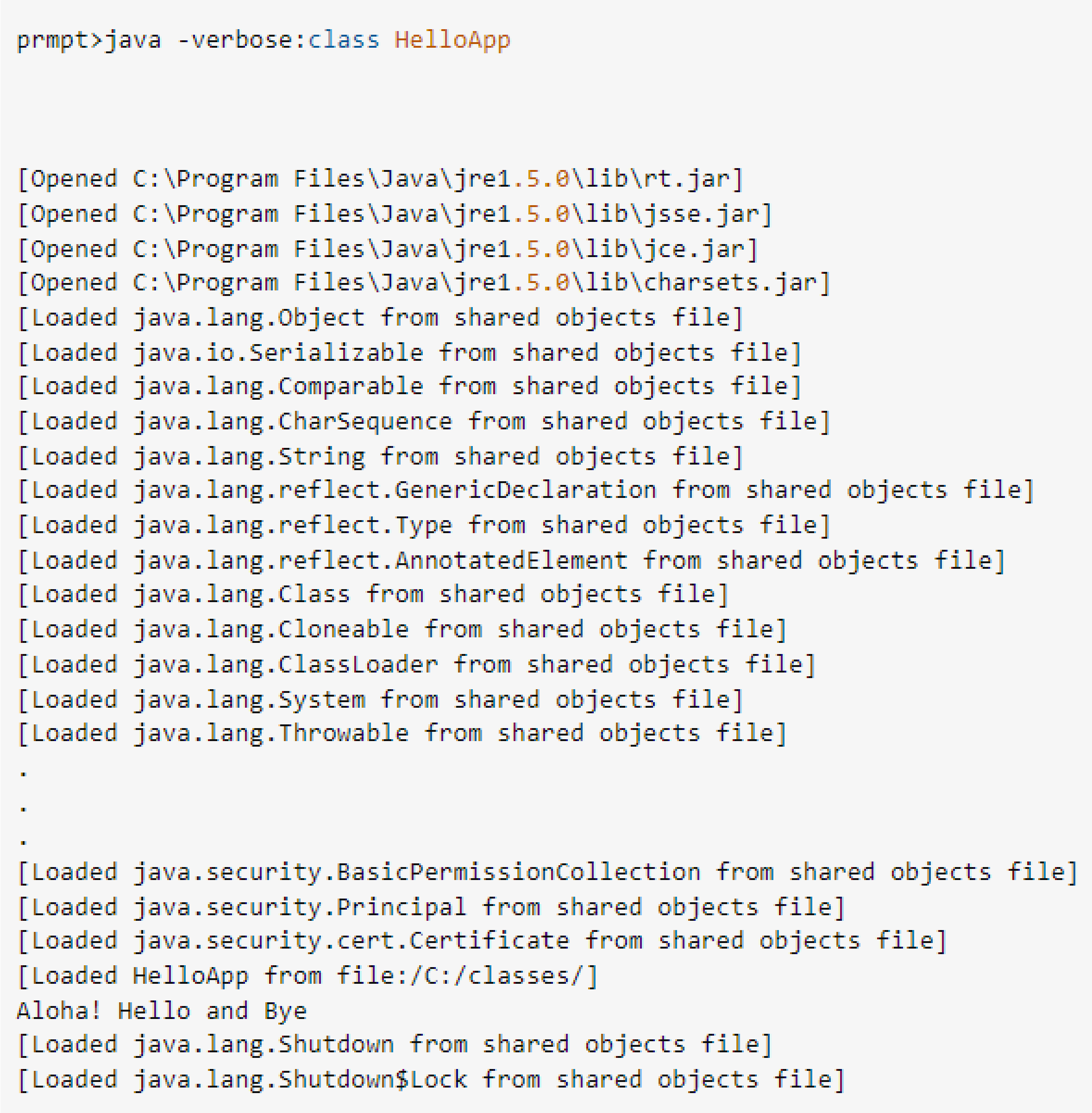
- Locating and loading necessary .class or .jar (**J**ava **AR**chive, aggregations of .class files) files into memory
  - .jar that offers standard Java packages (e.g., java.lang, java.io)
  - .class and .jar (dependency) for your application, which is specified in *classpath*

- Errors occur when class loader fails to locate a required .class

ClassLoader: Loading



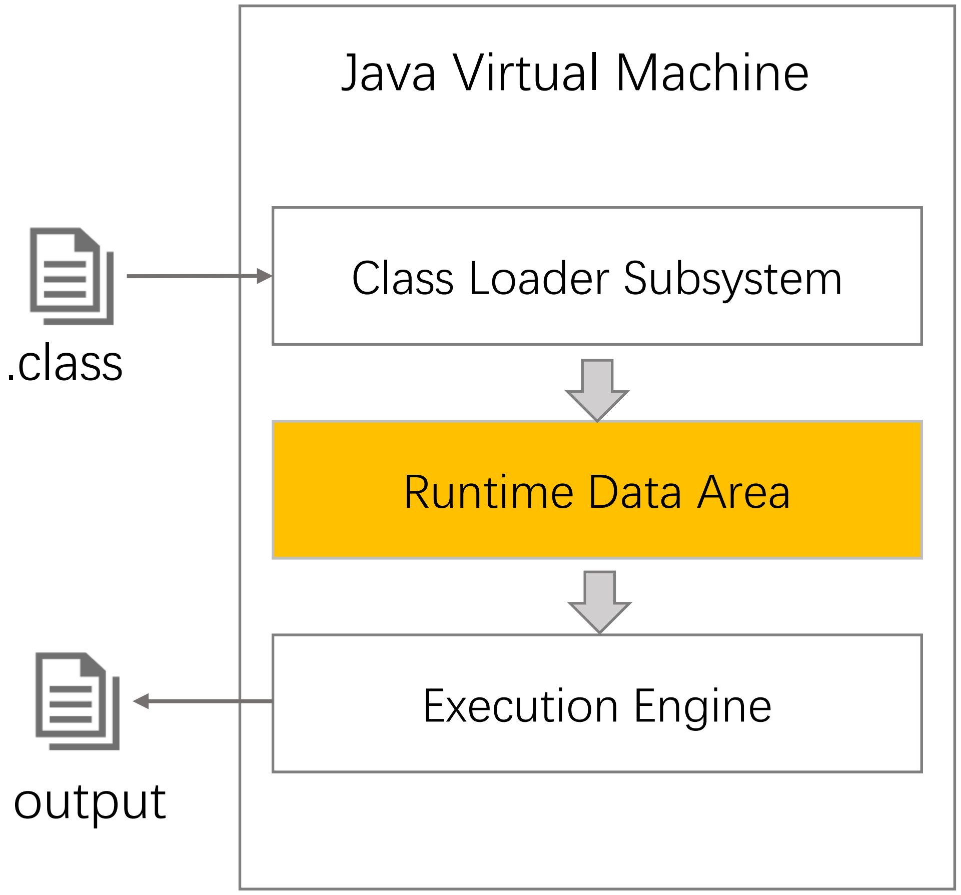
```
public class HelloApp {
   public static void main(String argv[]) {
      System.out.println("Aloha! Hello and Bye");
   }
}
```



```
prmpt>java -verbose:class HelloApp



[Opened C:\Program Files\Java\jre1.5.0\lib\rt.jar]
[Opened C:\Program Files\Java\jre1.5.0\lib\jsse.jar]
[Opened C:\Program Files\Java\jre1.5.0\lib\jce.jar]
[Opened C:\Program Files\Java\jre1.5.0\lib\charsets.jar]
[Loaded java.lang.Object from shared objects file]
[Loaded java.io.Serializable from shared objects file]
[Loaded java.lang.Comparable from shared objects file]
[Loaded java.lang.CharSequence from shared objects file]
[Loaded java.lang.String from shared objects file]
[Loaded java.lang.reflect.GenericDeclaration from shared objects file]
[Loaded java.lang.reflect.Type from shared objects file]
[Loaded java.lang.reflect.AnnotatedElement from shared objects file]
[Loaded java.lang.Class from shared objects file]
[Loaded java.lang.Cloneable from shared objects file]
[Loaded java.lang.ClassLoader from shared objects file]
[Loaded java.lang.System from shared objects file]
[Loaded java.lang.Throwable from shared objects file]
.
.
.
[Loaded java.security.BasicPermissionCollection from shared objects file]
[Loaded java.security.Principal from shared objects file]
[Loaded java.security.cert.Certificate from shared objects file]
[Loaded HelloApp from file:/C:/classes/]
Aloha! Hello and Bye
[Loaded java.lang.Shutdown from shared objects file]
[Loaded java.lang.Shutdown$Lock from shared objects file]
```

Image source: https://stackoverflow.com/questions/2424604/what-is-a-java-classloader

Java Virtual Machine (JVM)



Java Virtual Machine

.class

Class Loader Subsystem

Runtime Data Area

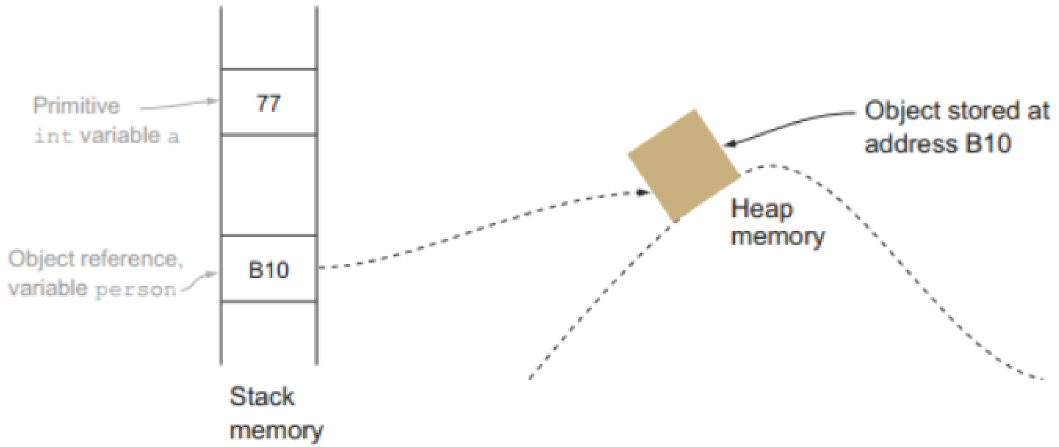
Execution Engine

output

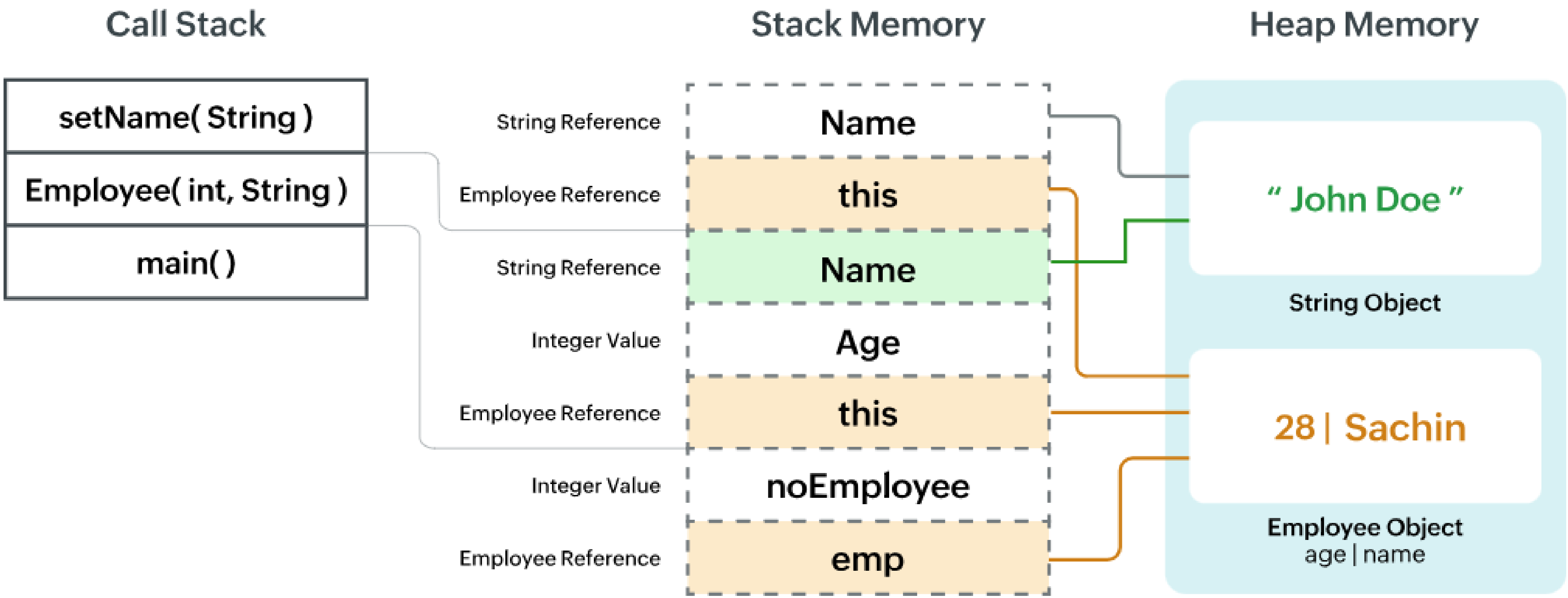
Runtime Data Area

Store all kinds of data and information

- Class-level data in Method Area
- Objects/instances in Heap Area
- Local variables in Stack Area

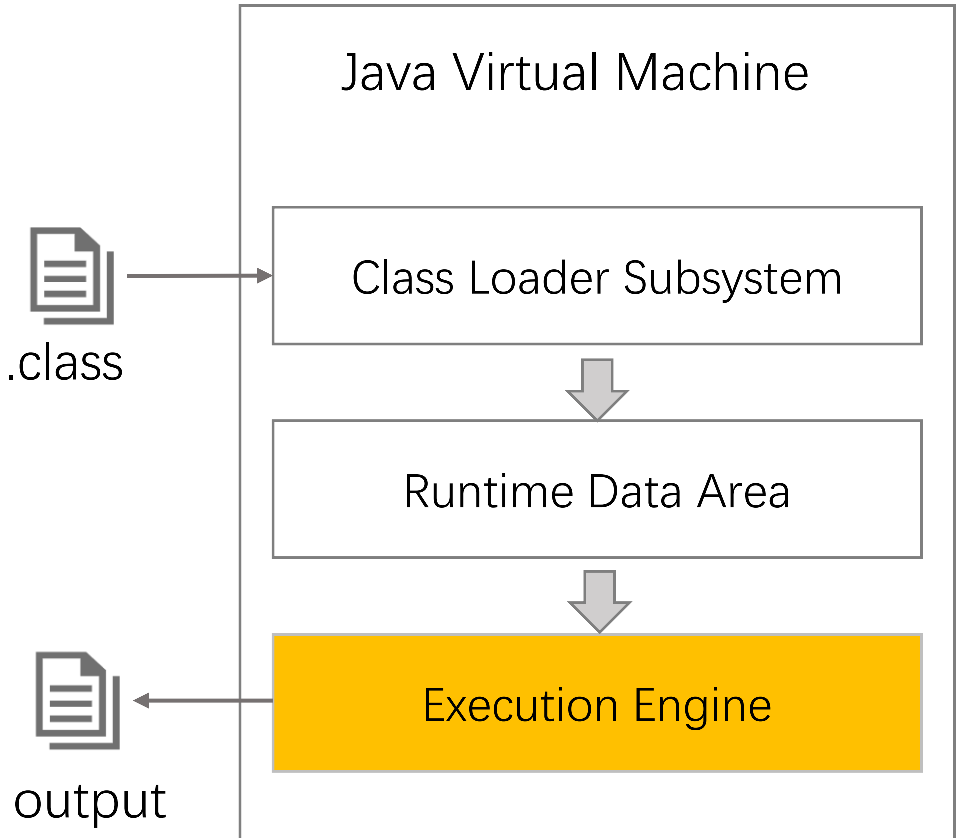


Memory Allocation



https://www.site24x7.com/learn/java/heap-and-stack-memory-management.html

Java Virtual Machine (JVM)



Execution Engine

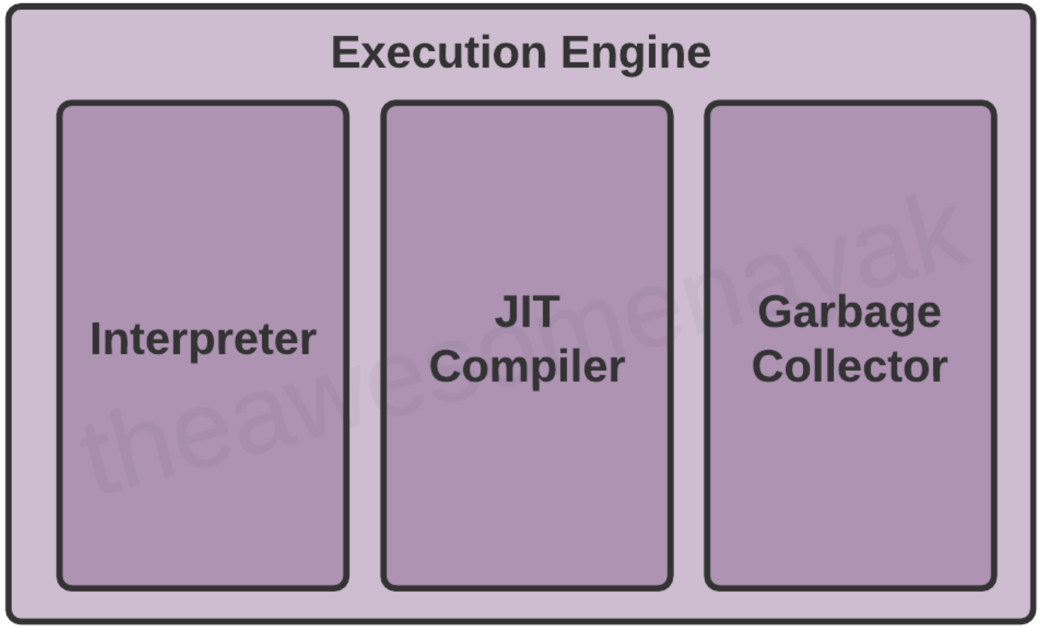
- Translating “run anywhere” .class code to “run on this particular machine” instructions
- Translation is done by Interpreter and JIT Compiler
- Finally, garbage collector identifies objects that are no longer in use and reclaims the memory

Execution Engine: Interpreter & JIT Compiler

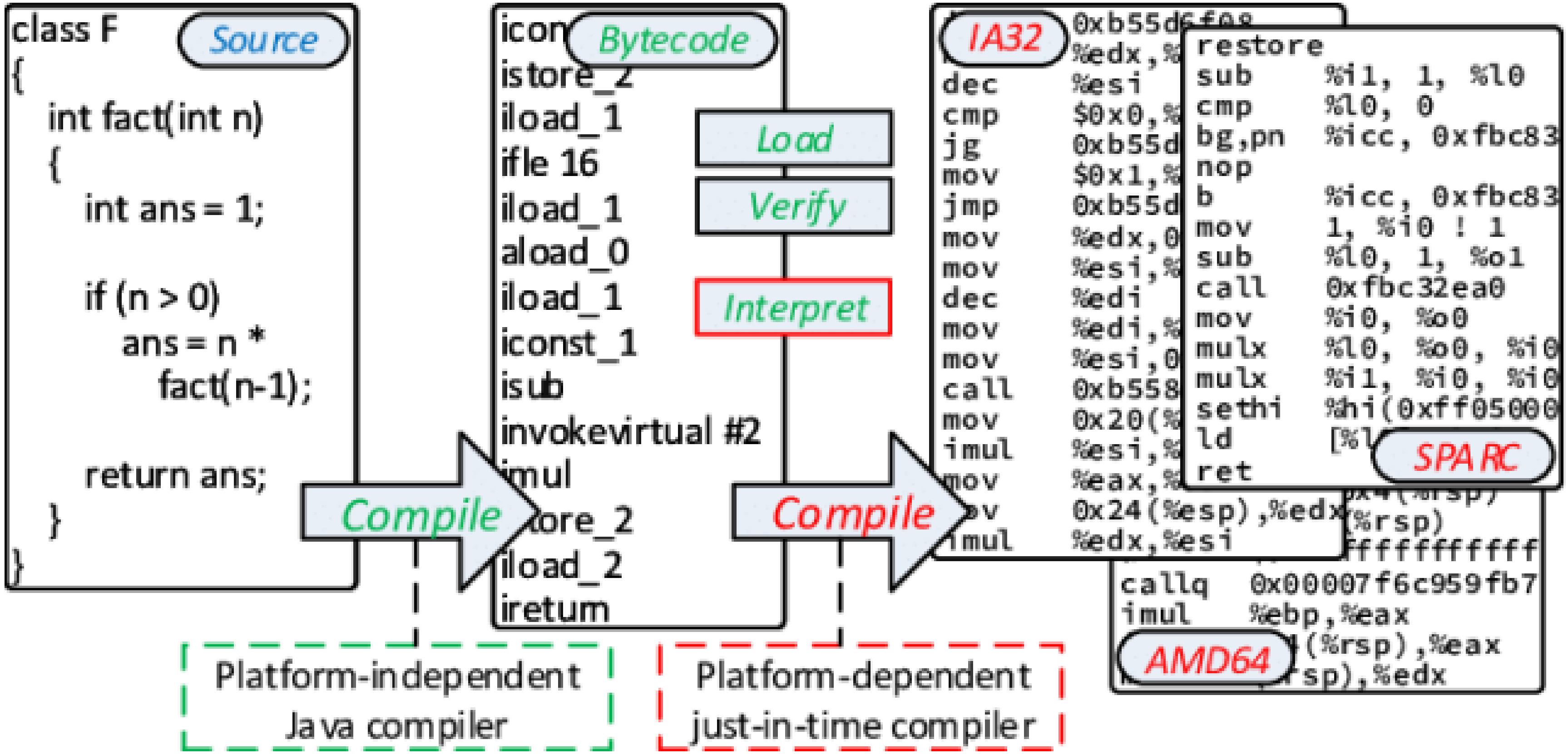
- Mainstream JVMs initially use Interpreter (解释器) to interpret and execute bytecode

- Yet, this is inefficient for Hot Spot code:
    - Methods that are invoked many times
    - Loops that are executed frequently

- For better efficiency, JIT compiler (即时编译器) compiles and optimize Hot Spot code to local machine code



Execution Engine: The Process



```
class F
{
   int fact(int n)
   {
      int ans = 1;

      if (n > 0)
         ans = n *
            fact(n-1);

      return ans;
   }
}
```

```
icon
istore_2
iload_1
ifle 16
iload_1
aload_0
iload_1
iconst_1
isub
invokevirtual #2
mul
tore_2
iload_2
iretum
```

```
            0xb55d6f08
            %edx,%
dec         %esi
cmp         $0x0,%
jg          0xb55d
mov         $0x1,%
jmp         0xb55d
mov         %edx,0
mov         %esi,%
dec         %edi
mov         %edi,%
mov         %esi,0
call        0xb558
mov         0x20(%
imul        %esi,%
mov         %eax,%
  v         0x24(%esp),%edx
imul        %edx,%esi
```

```
restore
sub         %i1, 1, %l0
cmp         %l0, 0
bg,pn       %icc, 0xfbc83
nop
b           %icc, 0xfbc83
mov         1, %i0 ! 1
sub         %l0, 1, %o1
call        0xfbc32ea0
mov         %i0, %o0
mulx        %l0, %o0, %i0
mulx        %i1, %i0, %i0
sethi       %hi(0xff05000
ld          [%l
ret
```

```
            x4(%rsp)
            (%rsp)
            ffffffffff
callq       0x00007f6c959fb7
imul        %ebp,%eax
            4(%rsp),%eax
            sp),%edx
```

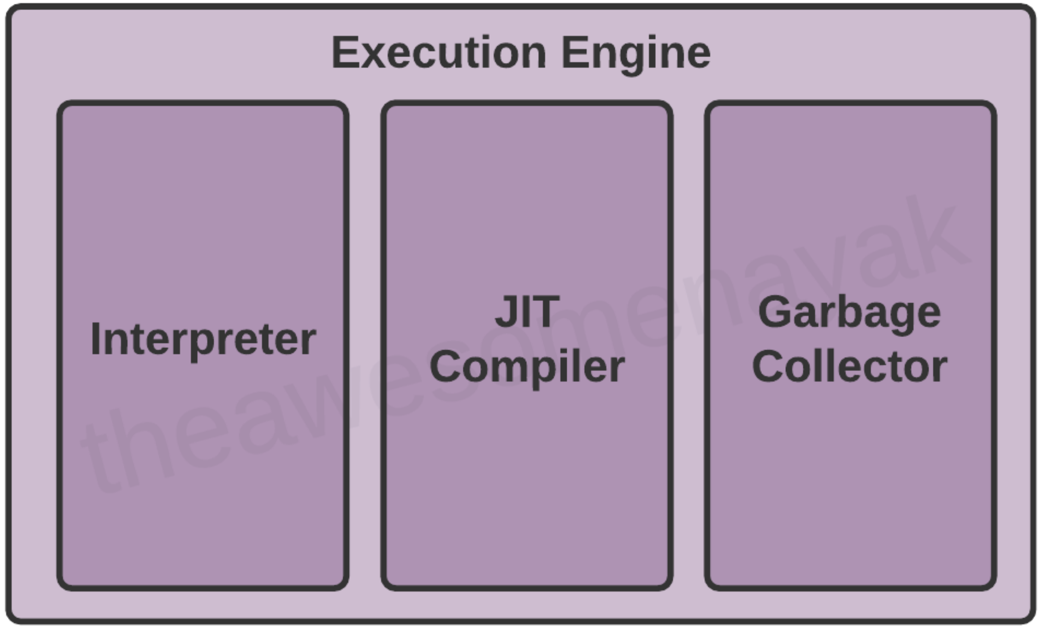
https://www.cesarsotovalero.net/blog/aot-vs-jit-compilation-in-java.html

Execution Engine: Garbage Collection

- **Stack area**: static memory allocation; memory can be computed once the class information is loaded

- **Heap area**: dynamic memory allocation; memory is known only during runtime (how many objects to create or to destroy is unknown until we execute the program)

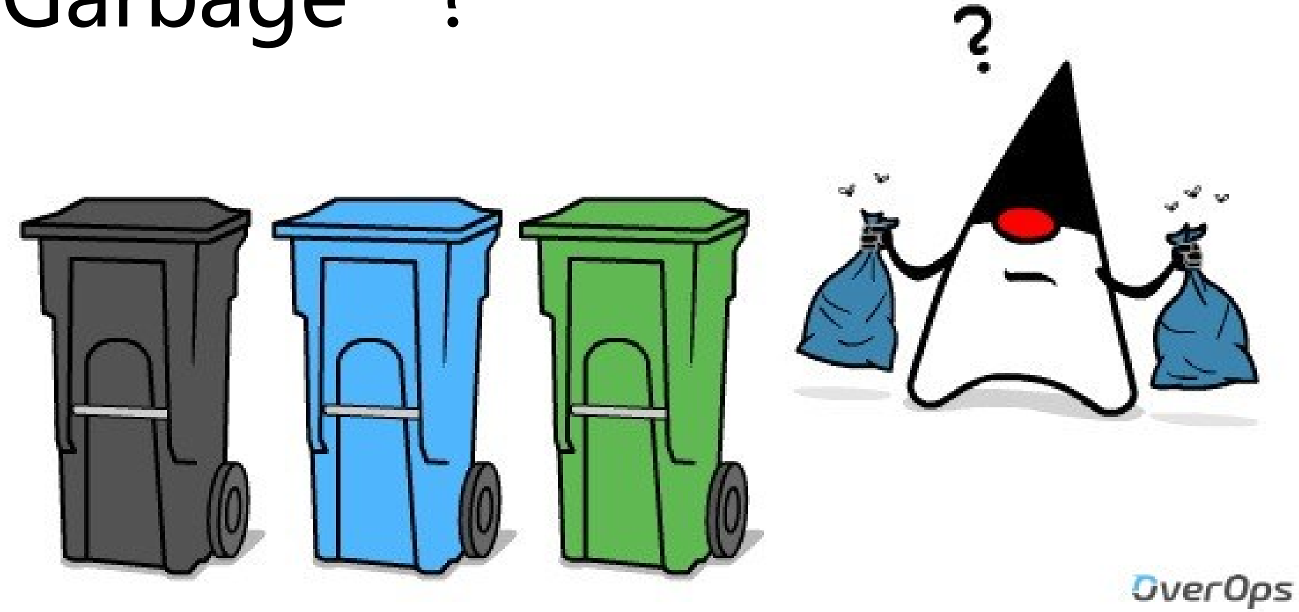
- Garbage collection focuses on the heap area



Which Objects are “Garbage”？

- 引用计数算法
  (Reference Counting)

- 可达性分析算法
  (Reachability Analysis)



Reference Counting

Reference counting GC algorithms associate a reference count with each object.

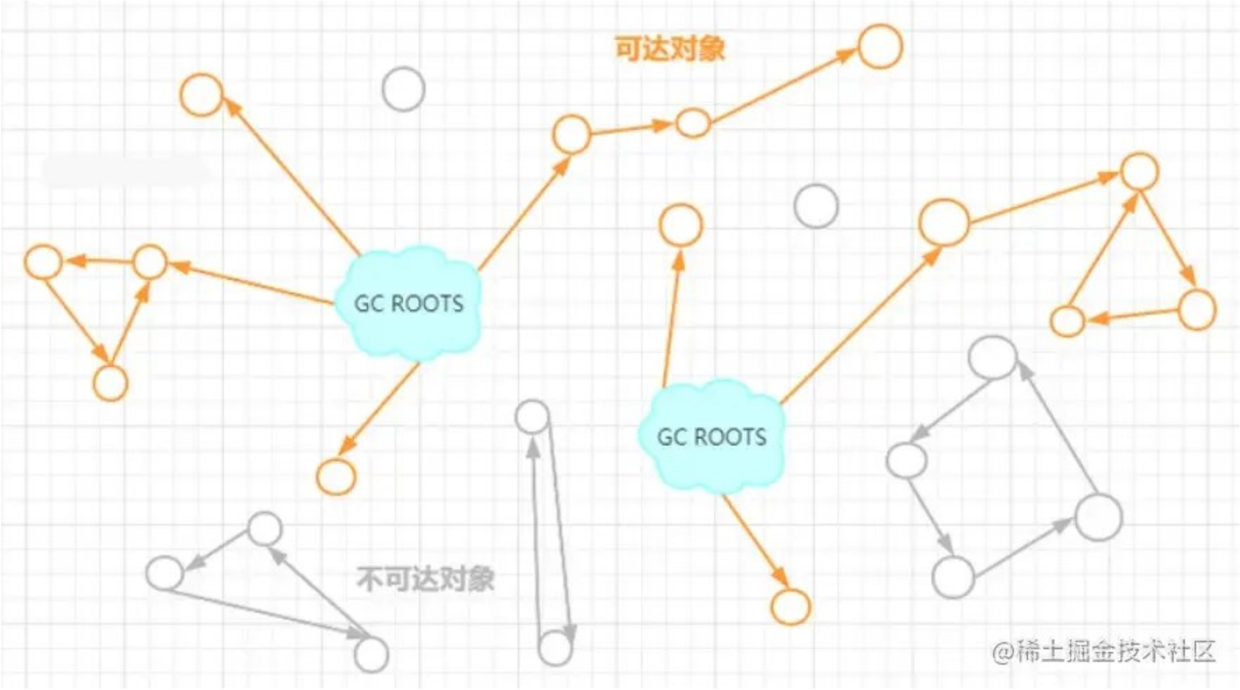
These algorithms consider an object to be alive as long as the number of references to that object is greater than zero.

What to do for Cyclic References ?

Reachability Analysis

- Starting from GC Roots, find reachable objects iteratively

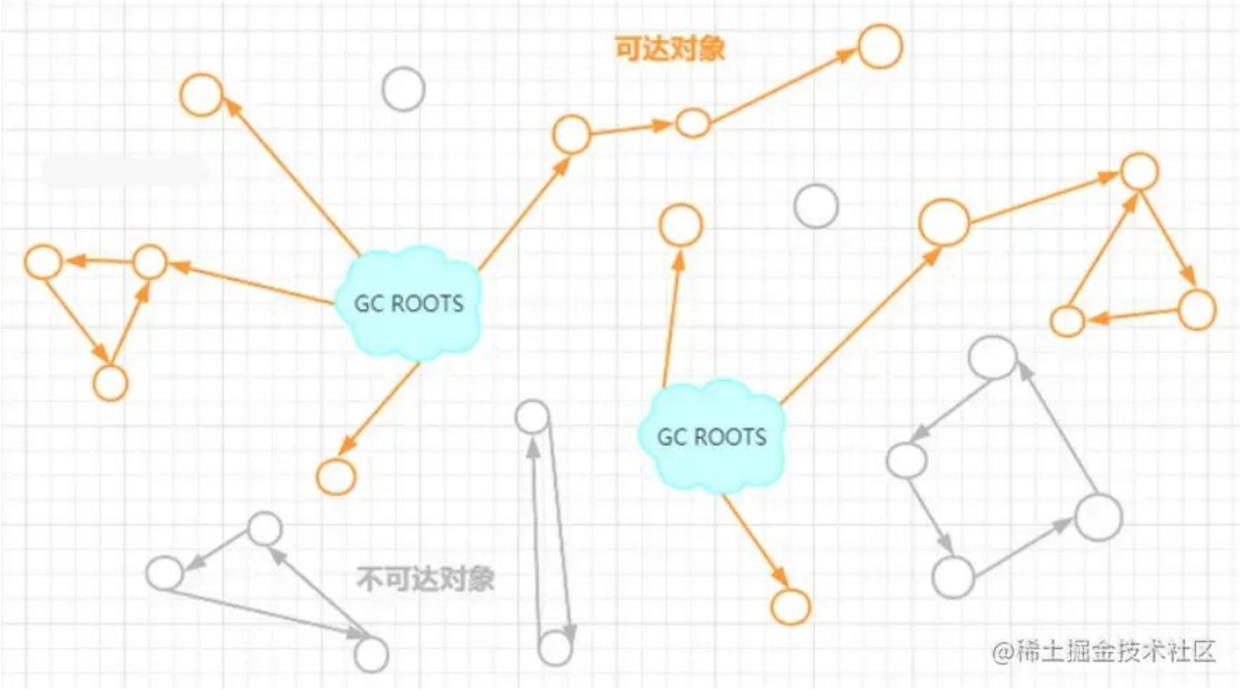
- Other objects, even with >0 reference count, will be garbage collected



https://juejin.cn/post/7123853933801373733

Reachability Analysis

- GC roots are objects that are themselves referenced by the JVM and thus keep every other object from being garbage-collected:
  - Local variables in the main method
  - Static variables of the main class
  - ……



https://juejin.cn/post/7123853933801373733

Lecture 1

- Course introduction
- Computer system & programs
- Java review and JVM
- Software design principles & OOP concepts

OO Concepts

- Encapsulation (封装)
- Abstraction (抽象)
- Inheritance (继承)
- Polymorphism (多态)

Software Design Principles

- High Cohesion, Low Coupling (高内聚、低耦合)
- Information Hiding (信息隐藏)
- Reusability (可复用性)

High Cohesion, Low Coupling

- Modules (模块): A complex software system can be divided into simpler pieces called *modules*

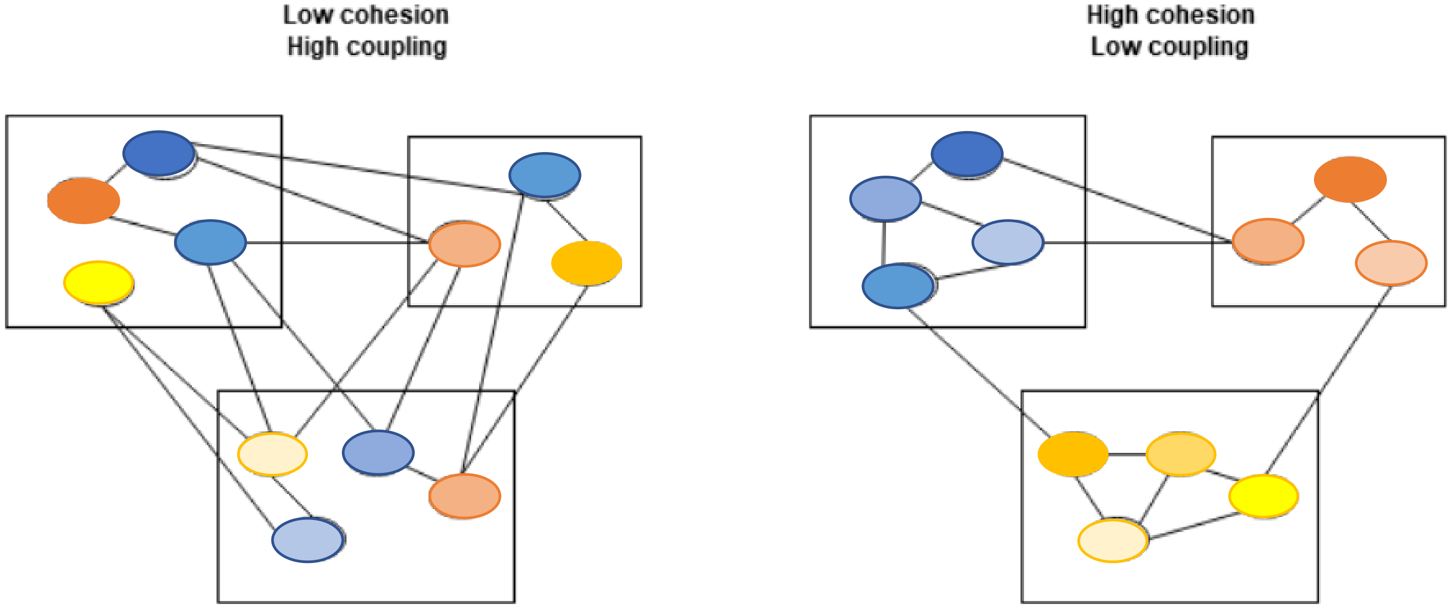
- Cohesion (内聚): How elements of a module are functionally related to each other

- Coupling (耦合): How different modules depend on each other

High Cohesion, Low Coupling

- High cohesion: modules are self-contained and have a single, well-defined purpose; all of its elements are directly related to the functionality that is meant to be provided by the module
- Low coupling: modules should be as independent as possible from other modules, so that changes to one module will have minimal impact on other modules

Difficult to read, understand, reuse, test, and maintain



Easy to understand, extend, and modify

Source: Software Architecture with C++ by Adrian Ostrowski, Piotr Gaczkowski

OO Concepts

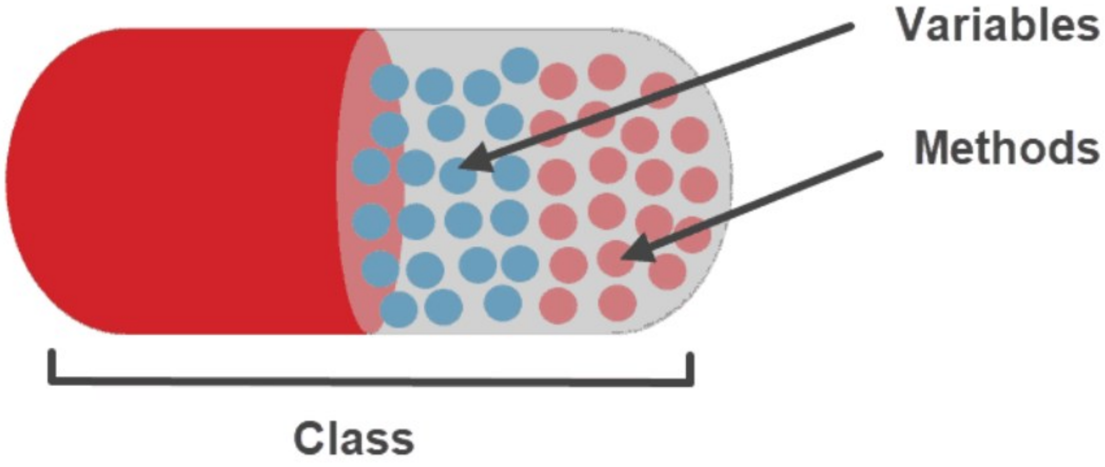
- Encapsulation (封装)
- Abstraction (抽象)
- Inheritance (继承)
- Polymorphism (多态)

Software Design Principles

- High Cohesion, Low Coupling (高内聚、低耦合)
- Information Hiding (信息隐藏)
- Reusability (可复用性)

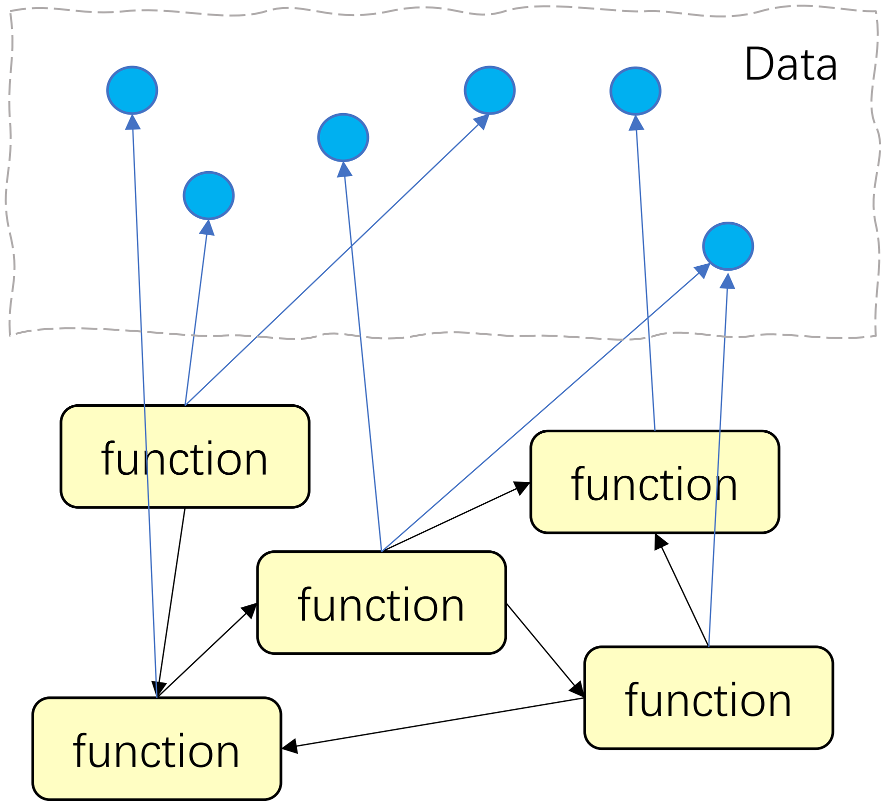
Encapsulation

- Bundling the data and functions which operate on that data into a single unit, e.g., a class in Java.
- Program should interact with object data *only* through the object's methods.



Encapsulation is achieved by the **Access Control** mechanism in Java

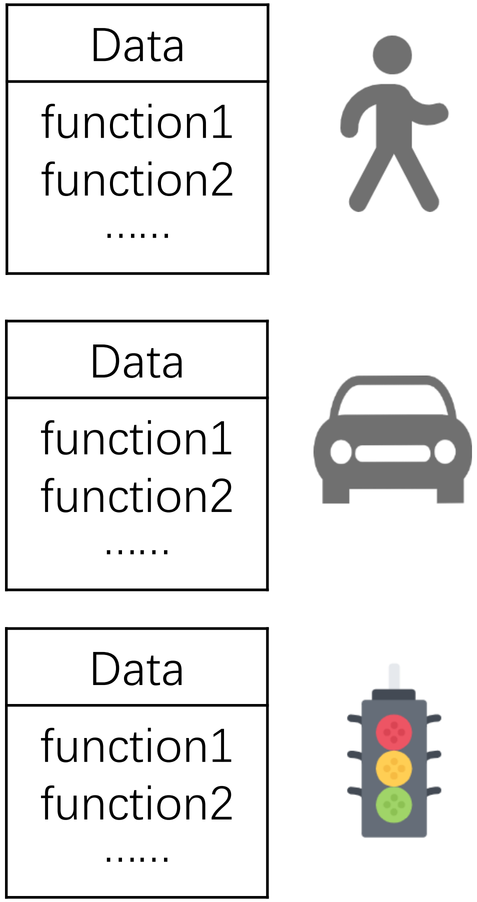
Procedural Design



High coupling. Reduced information hiding.
Hard to make changes and to scale.

Traffic Control System

Object-oriented Design



Data
function1
function2
......

Data
function1
function2
......

Data
function1
function2
......

High cohesion. Good information hiding.
Easier to maintain and extend.

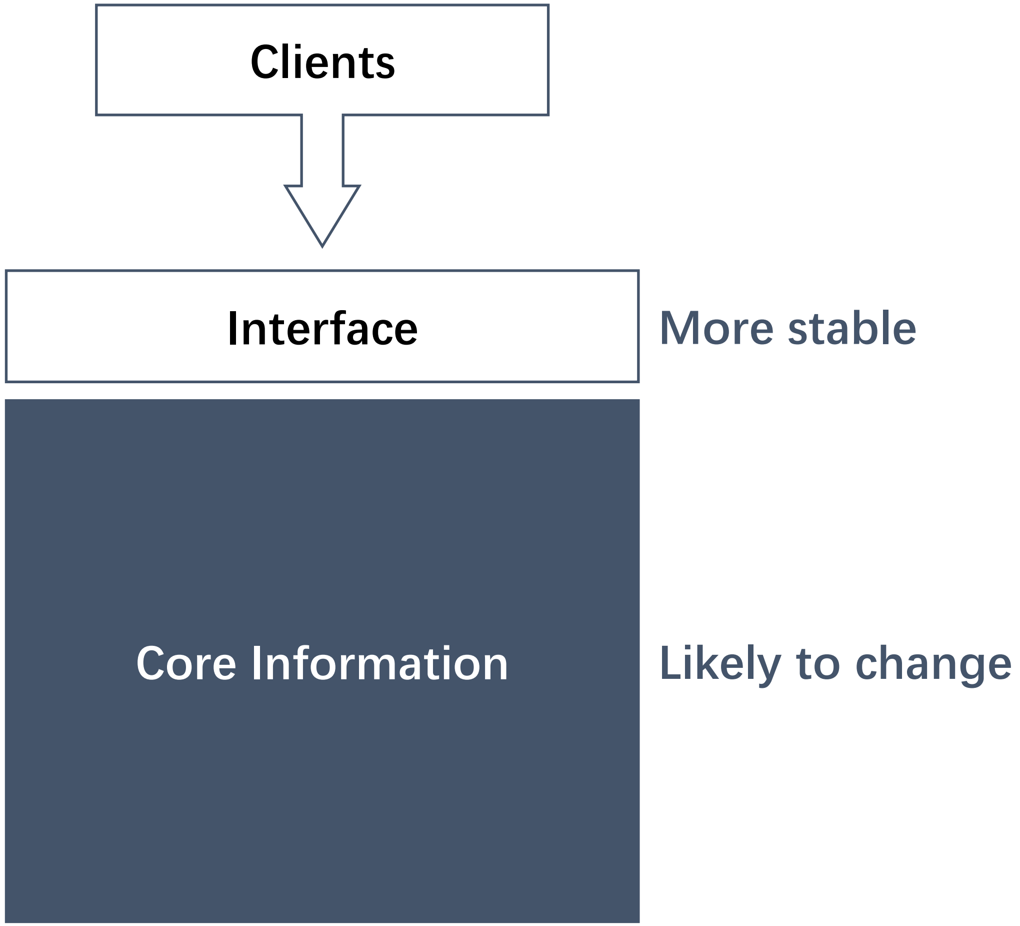
Image source: https://www.hseblog.com/cross-road-safely/

Information Hiding

- Key idea: Hiding certain information, such as design decisions, data, and implementation details, from client programs

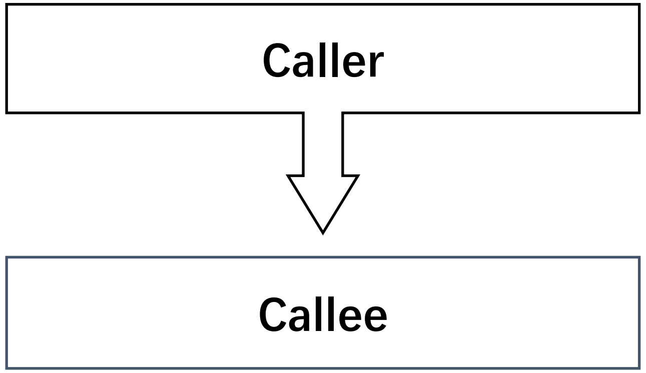
- Advantages: Client programs won’t have to change even if the core design or implementation is changed

Increasing coupling -> breaking information hiding



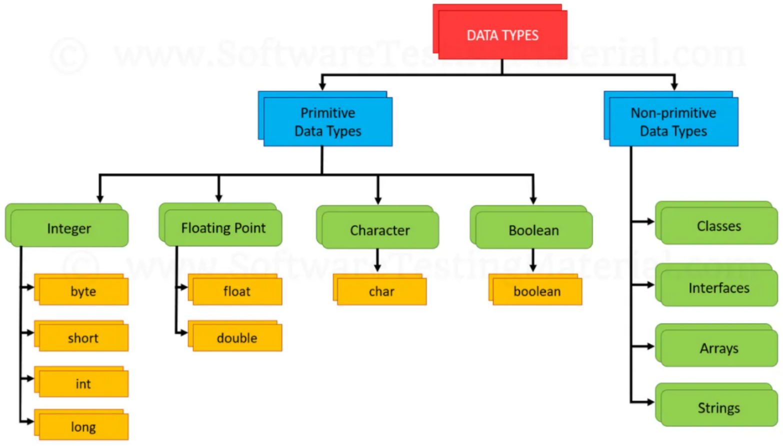
Information Hiding

Example 1. Function Call



The caller function doesn’t have to know how the callee function works internally; it only has to know callee’s arguments and return type

Example 2. Data Representation



You don’t need to know how a data type is implemented in order to use it;

Image source: https://www.softwaretestingmaterial.com/data-types-in-java/

OO Concepts

- Encapsulation (封装)
- Abstraction (抽象)
- Inheritance (继承)
- Polymorphism (多态)

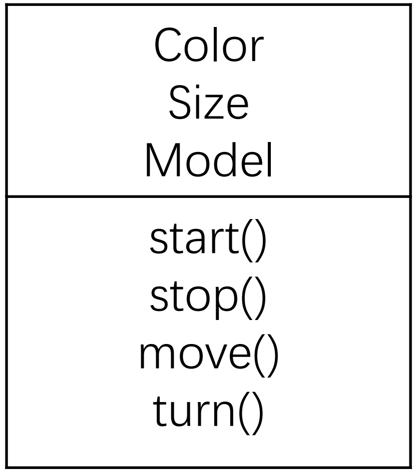
Software Design Principles

- High Cohesion, Low Coupling (高内聚、低耦合)
- Information Hiding (信息隐藏)
- Reusability (可复用性)

Abstraction

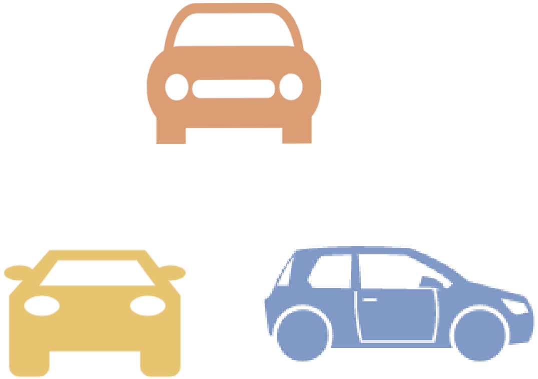
- Abstraction simplifying complex systems by exposing only the necessary details.
- Abstraction solves problem at design level
- Achieved in Java by interface and abstract class

Car Class



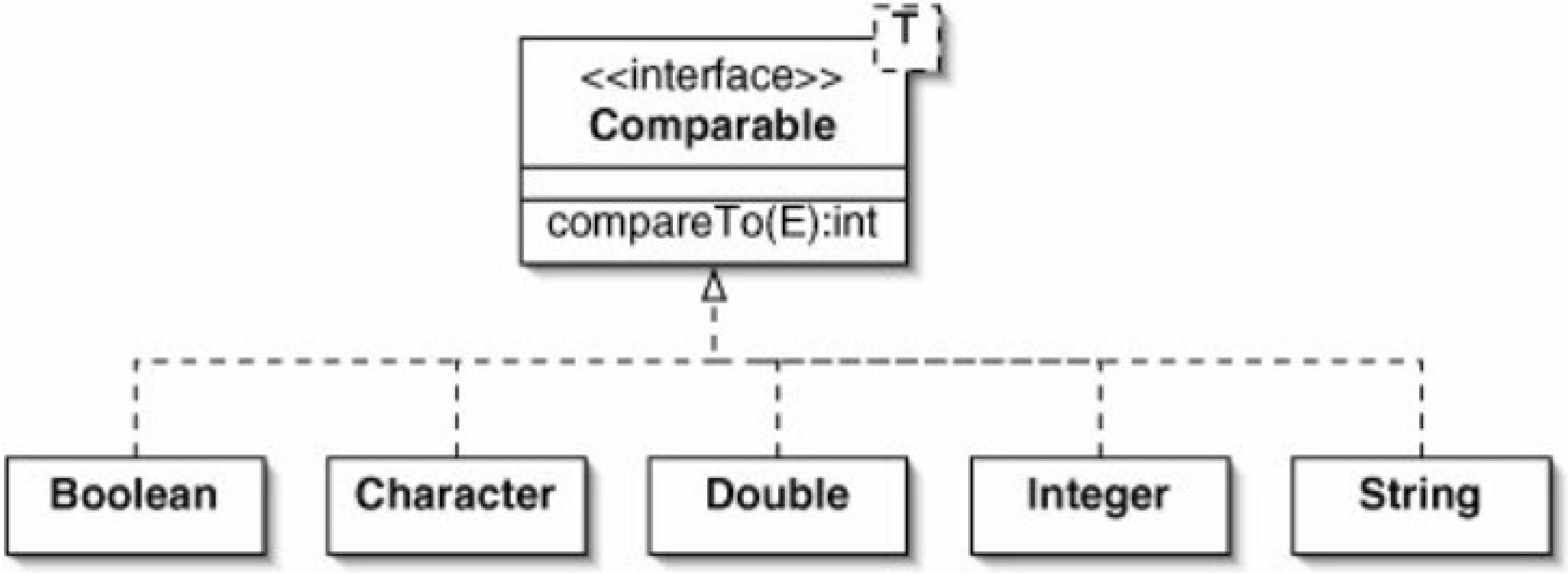
Color
Size
Model

start()
stop()
move()
turn()



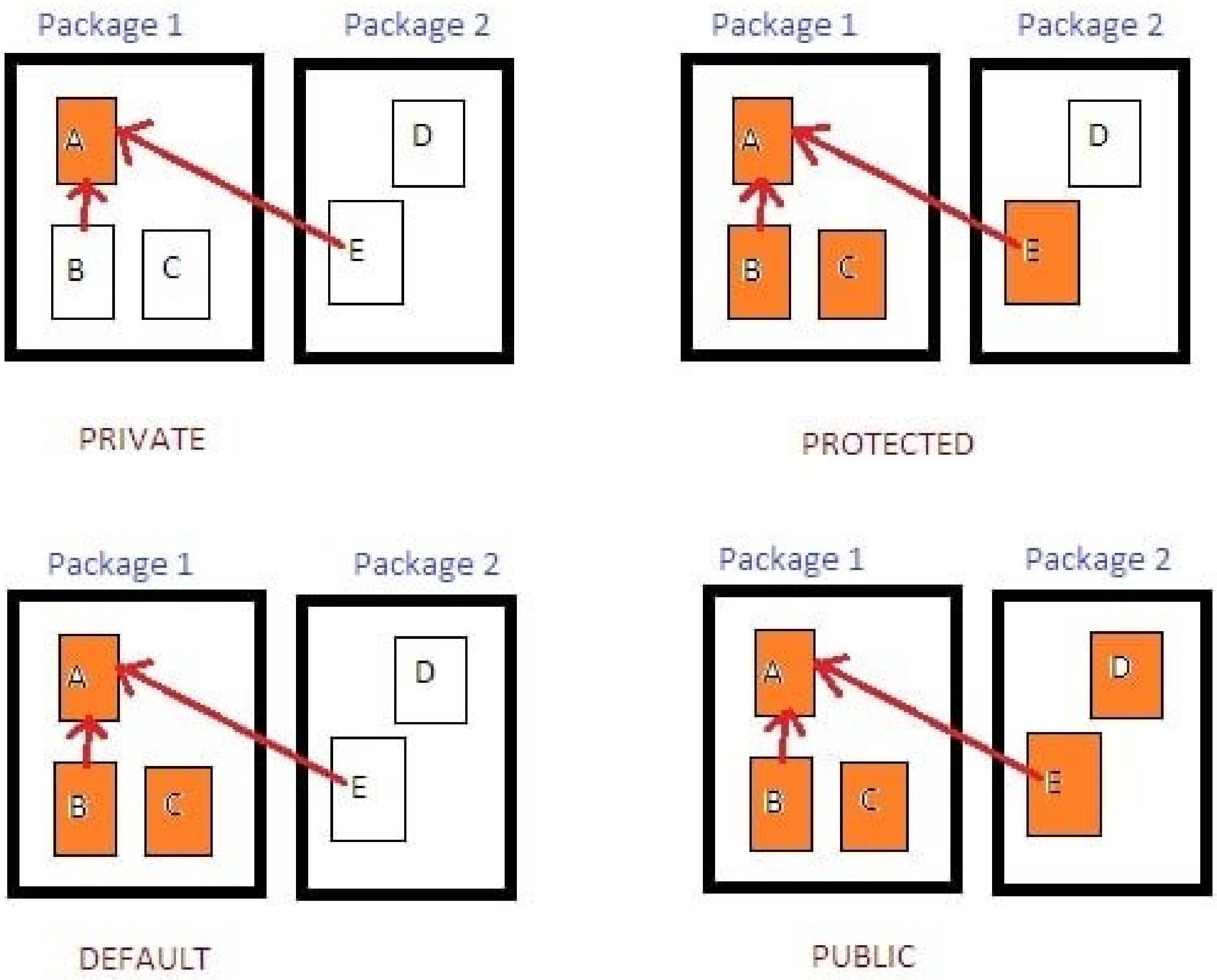
Abstraction

- Example: All the numeric data types can be abstracted to be comparable



Access Control

Use access modifiers to determine whether other classes can use a particular field or invoke a particular method



Reusability

- Reusable code refers to the use of the same source code collection in multiple applications or systems.
- At its best, code reuse is accomplished by sharing common classes or collections of functions and procedures. At its worst, code reuse is accomplished by copying and then modifying existing code.
- In OOP, code reusability is to design objects in a way that can later on be used on other systems.

OO Concepts

- Encapsulation (封装)
- Abstraction (抽象)
- Inheritance (继承)
- Polymorphism (多态)

Software Design Principles

- High Cohesion, Low Coupling (高内聚、低耦合)
- Information Hiding (信息隐藏)
- Reusability (可复用性)

Inheritance

- Motivation: objects are similar and share common logics
- Inheritance allows a new class (subclass, child class, derived class) to be created by deriving variables and methods from an existing class (superclass, parent class, base class)
- Reduce code redundancy & support good code reuse

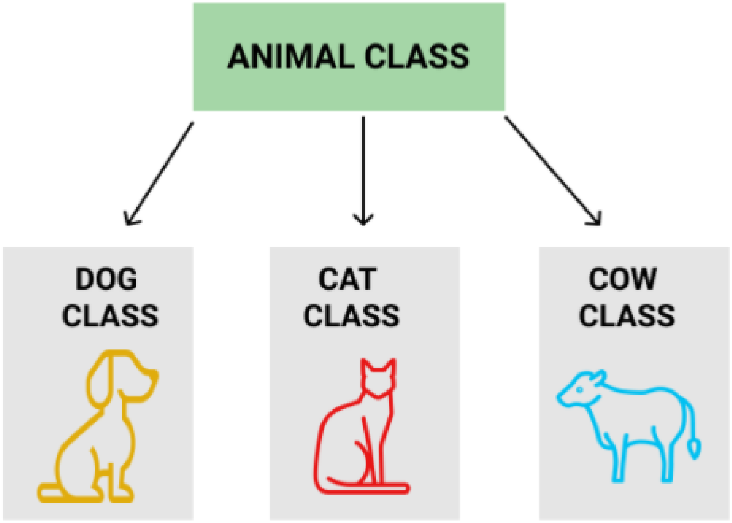
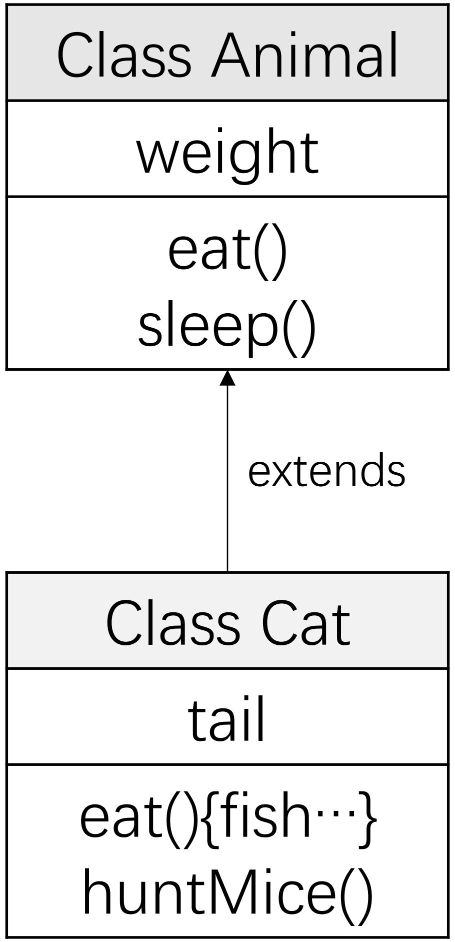


Image source: OOP Inheritance. San Joaquin Delta College. https://eng.libretexts.org/@go/page/34639

Inheritance

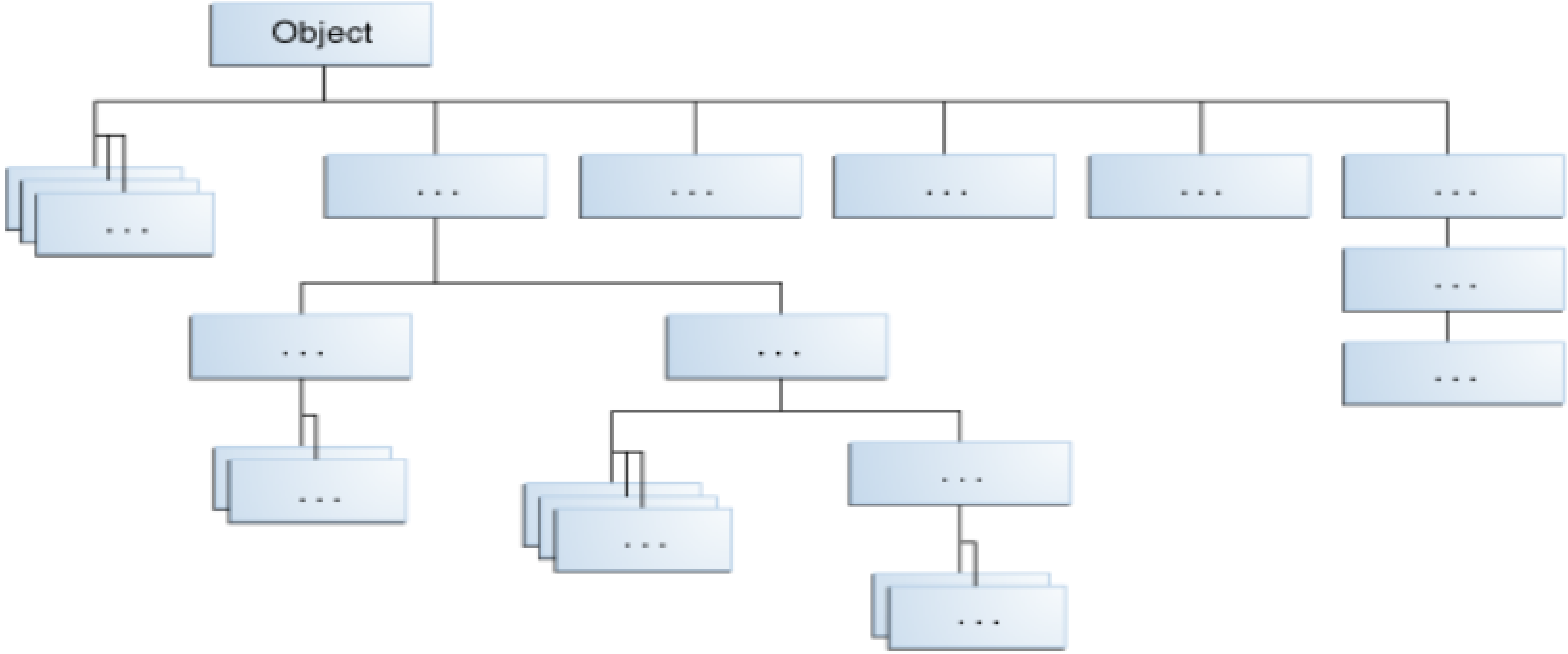
- Subclass could use inherited field directly (`weight`)
- Subclass could declare new fields (`tail`)



- Subclass could use inherited method directly (`sleep()`)
- Subclass could override methods in superclass (`eat()`)
- Subclass could declare new methods (`huntMice()`)

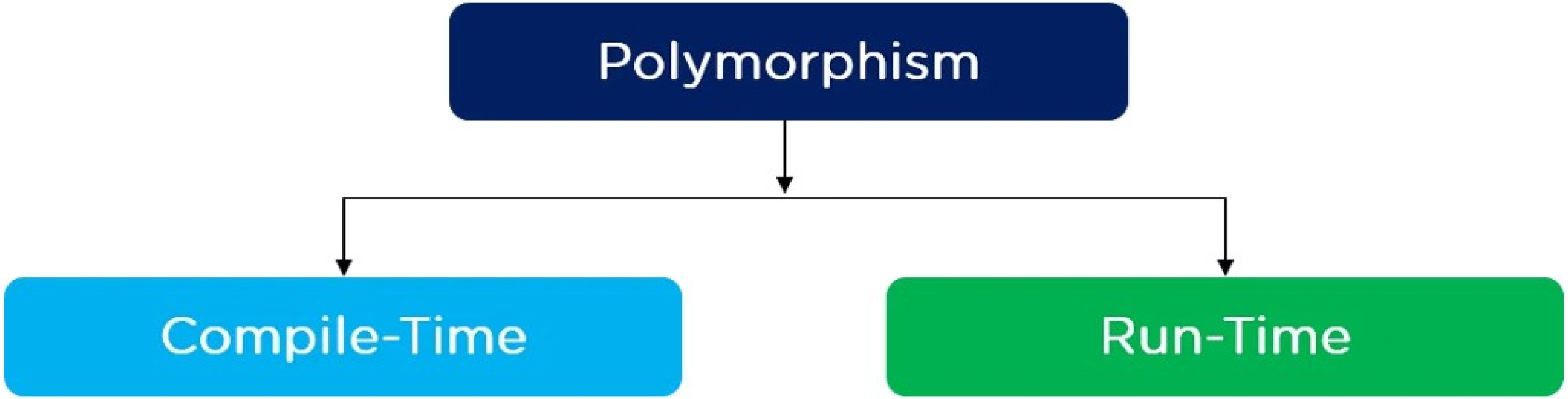
The Java Class Hierarchy

- The `Object` class is the parent class of all the classes

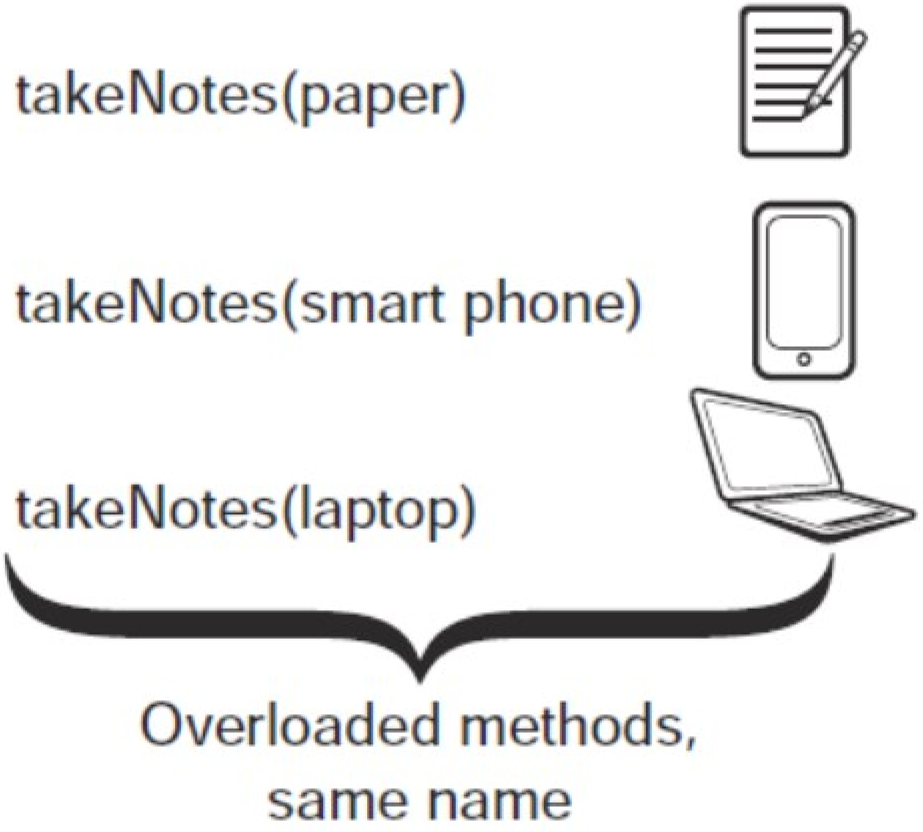


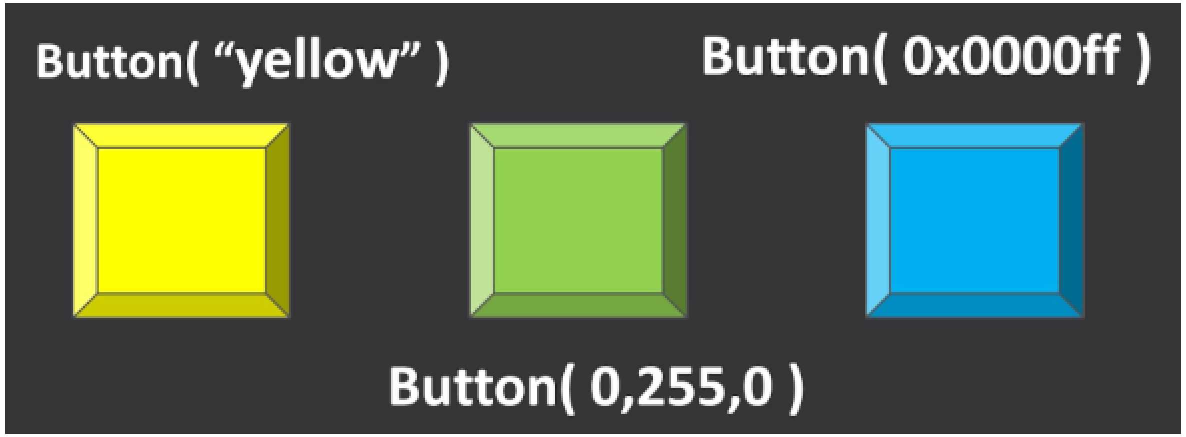
What is Polymorphism?

In general, "polymorphism" refers to the ability of a single entity or concept to take on multiple forms or have multiple meanings.



Compile-time Polymorphism





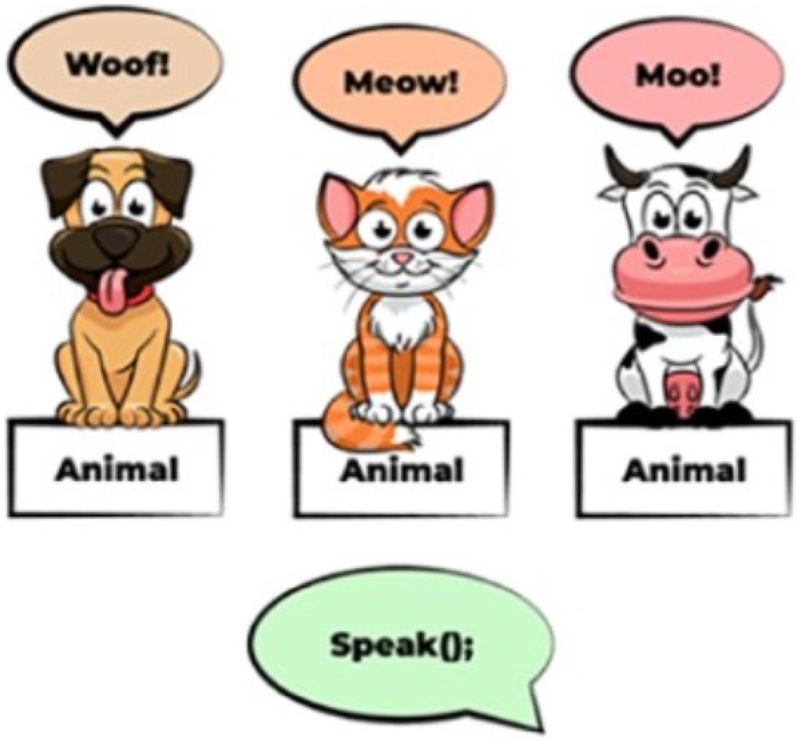
Images:
https://gyansetu-java.gitbook.io/core-java/method-overlaoding
https://www.examtray.com/java/last-minute-java-constructor-overloading-explained-examples-tutorial

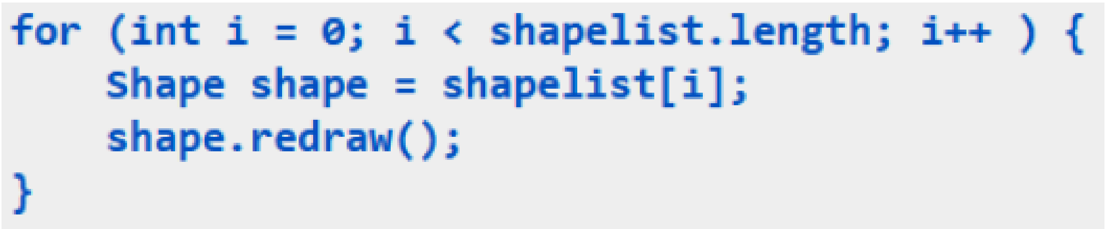
Static Binding

- Binding: mapping the name of the method to the final implementation.
- Method overloading are resolved using static binding, in which mapping is resolved at compile time

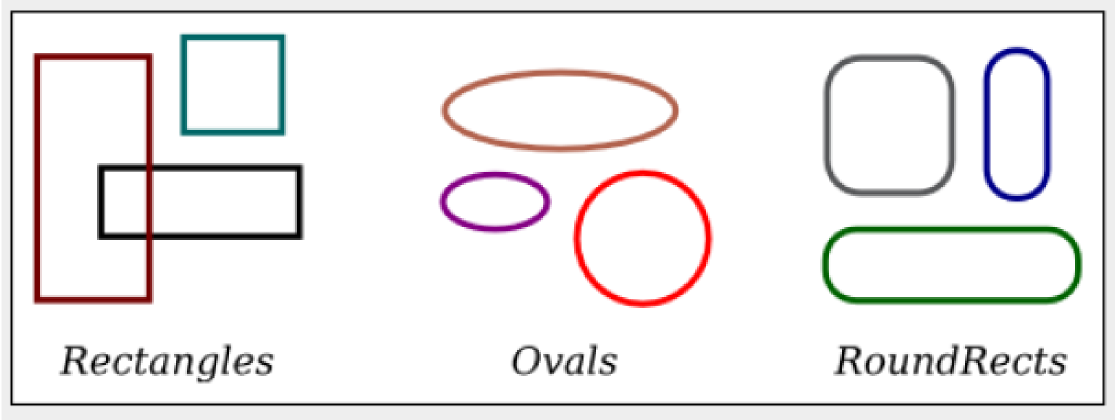
```
class Calculator{
    public int sum(int a, int b){
        return a+b;
    }

    public int sum(int a, int b, int c){
        return a+b+c;
    }
}
```

Runtime Polymorphism





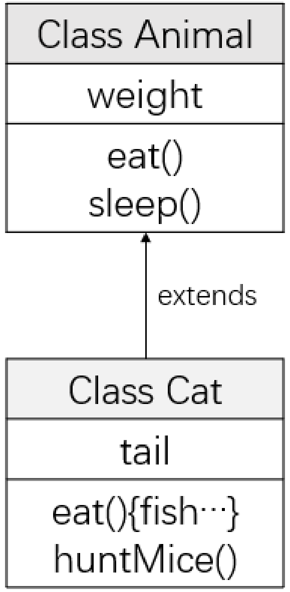
```
for (int i = 0; i < shapelist.length; i++ ) {
    Shape shape = shapelist[i];
    shape.redraw();
}
```



Images: https://codegym.cc/groups/posts/polymorphism-in-java

Dynamic Binding

- Binding: mapping the name of the method to the final implementation.
- Method overriding are resolved using dynamic binding, in which the mapping is resolved at execution time



```
Animal x = new Cat();
x.eat();
```

✓ Compilation ok, since Animal type has eat() method

✓ At execution time, x refers to a Cat object, so invoking Cat's eat() method

Next Lecture

- Generics
- ADT
- Collections