SOFTWARE ARCHITECTURE
7. JAVA VIRTUAL MACHINE

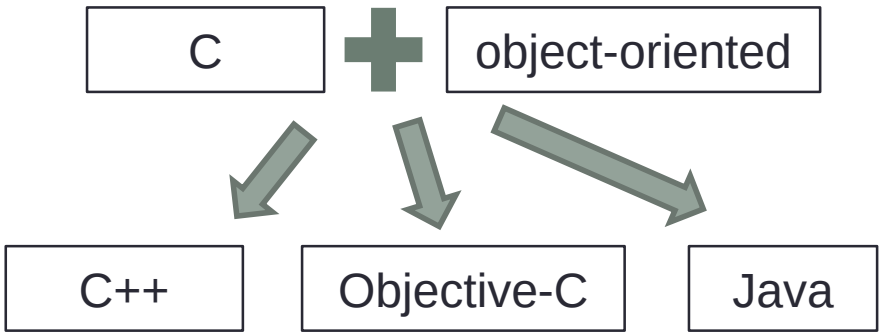
Tatsuya Hagino
hagino@sfc.keio.ac.jp

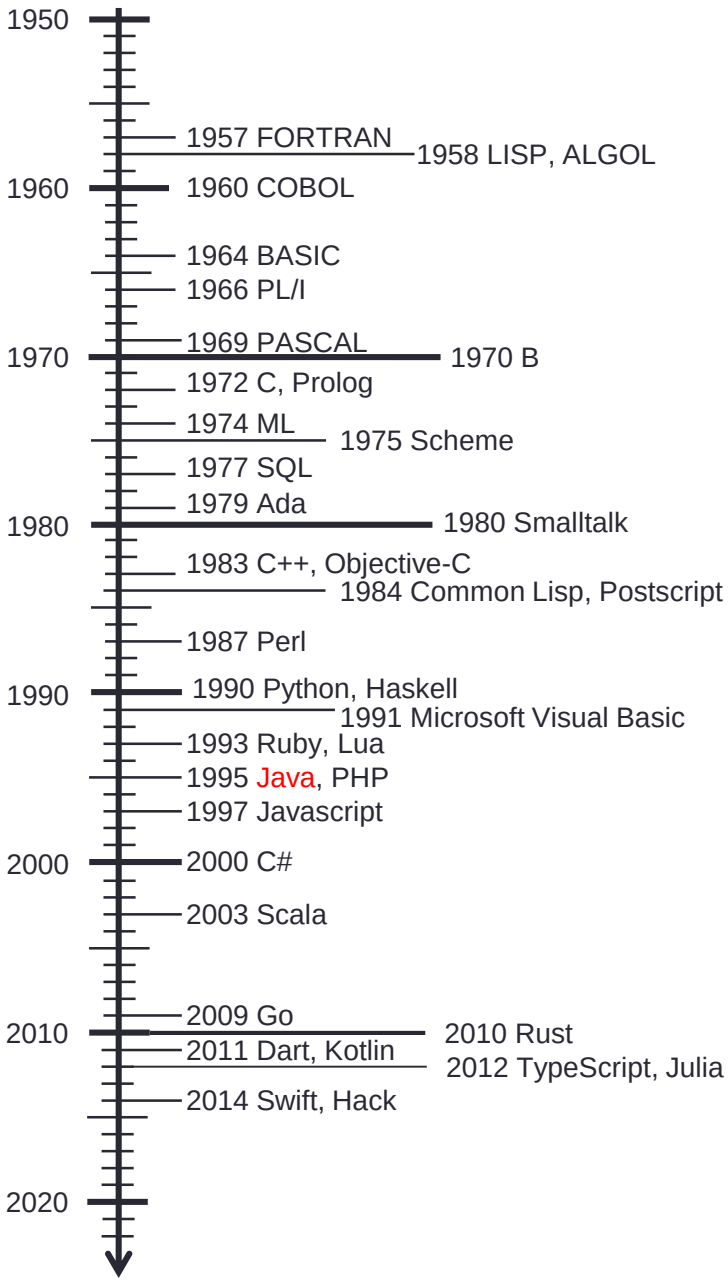
lecture URL

https://vu5.sfc.keio.ac.jp/slide/

Java

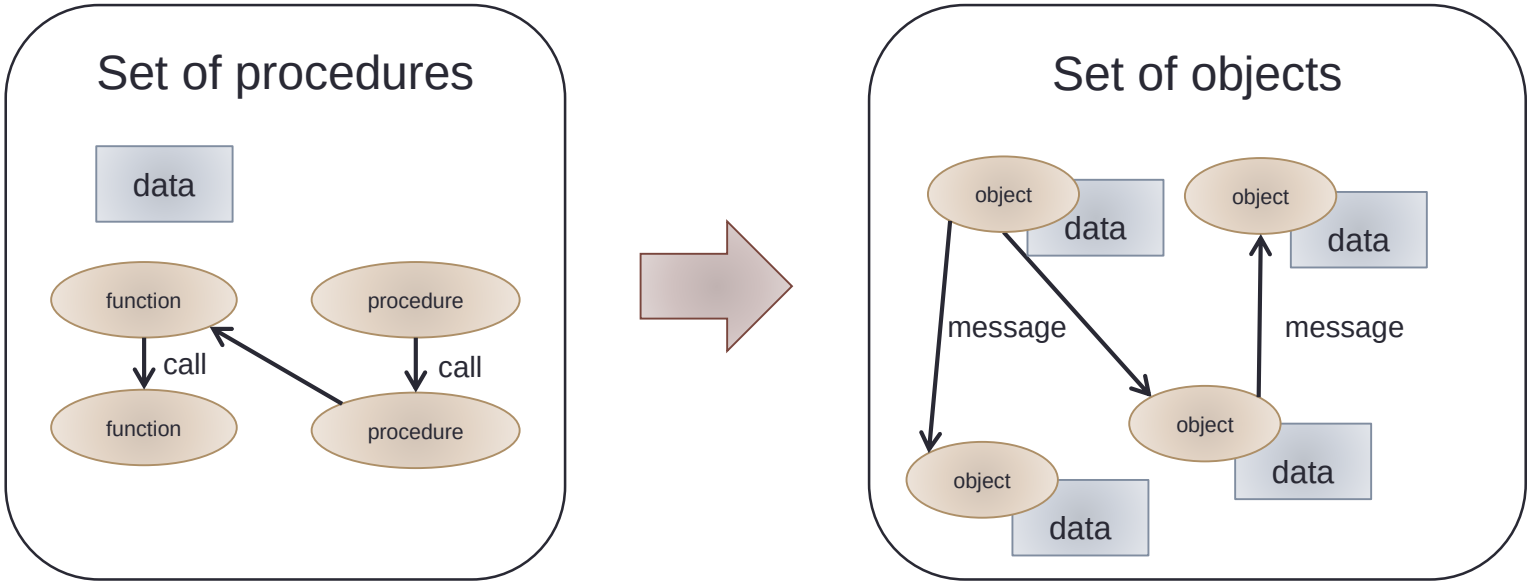
- Programming Language Java
  - Introduced in 1995
  - Developed for embedding into small devices
  - Object-oriented programming language
  - C like syntax
    - no 'goto' statement





Object-oriented Programming Language

- Software = a set of objects
  - Objects send messages to other objects for communication.

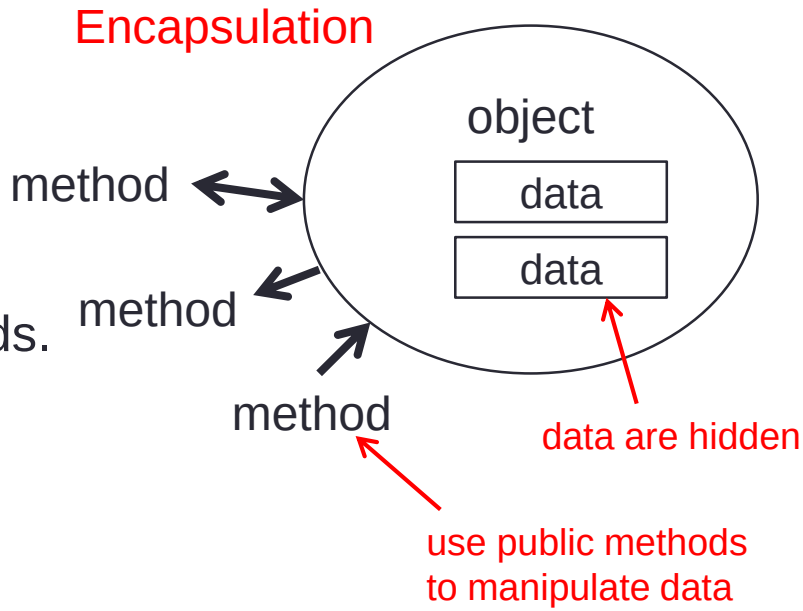


non object-oriented
- Procedures and data are independent
- A procedure is a set of instructions.
- Procedure centered

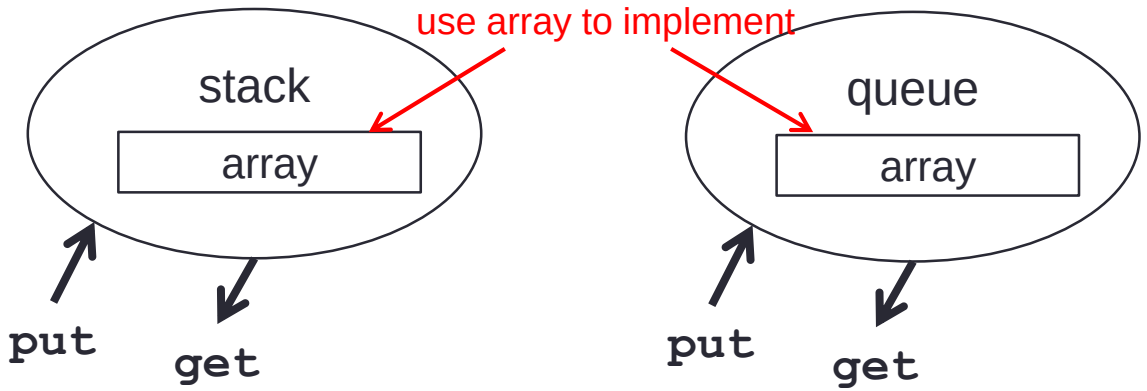
object-oriented
- Each object has data
- Objects receives messages to manipulate data.
- Data centered

Features of object-oriented programming

- Encapsulation
  - Data are encapsulated in a object.
  - Use methods to access data.
  - An object consist of data and methods.
  - Abstract data type
    - vs concrete data type



Abstract data type

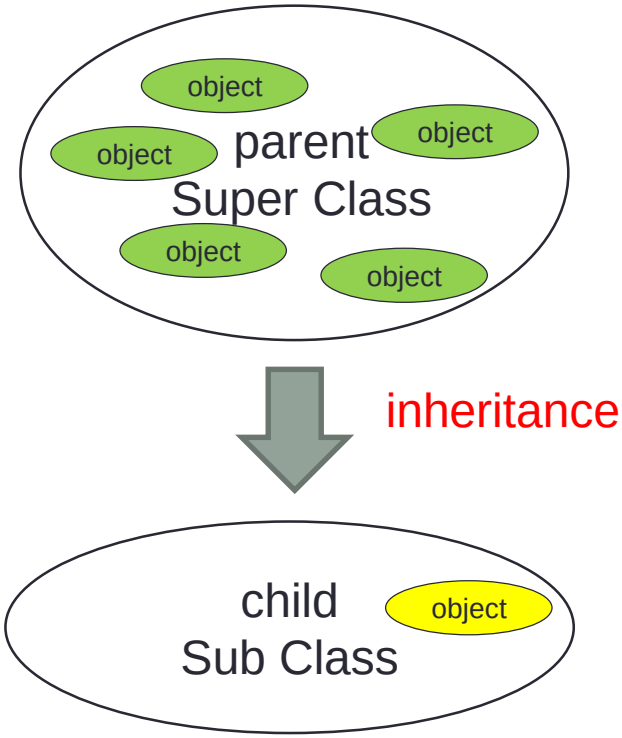


LIFO = Last In First Out

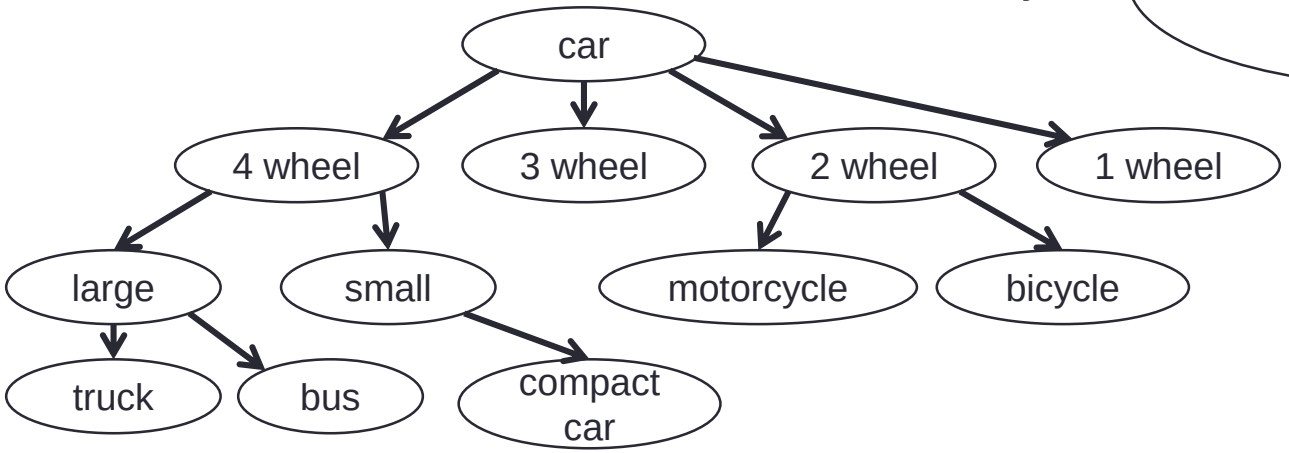
FIFO = First In First Out

Features of object-oriented programming

- Class and Inheritance
  - Each object belongs to a class.
  - The object is an instance of the class.
  - Classes have parent-child relation.
    - Child class inherits parent class.
    - Parent methods are child methods.
    - Parent data are child data.



- Class hierarchy
  - Parent-child relation creates class hierarchy.



Java Program Example

```
public class Human {
  private String name;
  public Human(String name) {
    this.name = name;
  }
  public String getName() {
    return name;
  }
}

public class Student extends Human {
  private int id;
  public Student(int id, String name) {
    this.id = id;
    Human(name);
  }
  public int getID() {
    return id;
  }

  public static void main(String[] args) {
      Student s = new Student(12345, "Hagino");
      System.out.println(s.getName());
    }

}
```

class name

local data

constructor for objects

refer itself

method

parent class

invoke a method

create a new object

Human

- Class for human
- Holds a name as its data

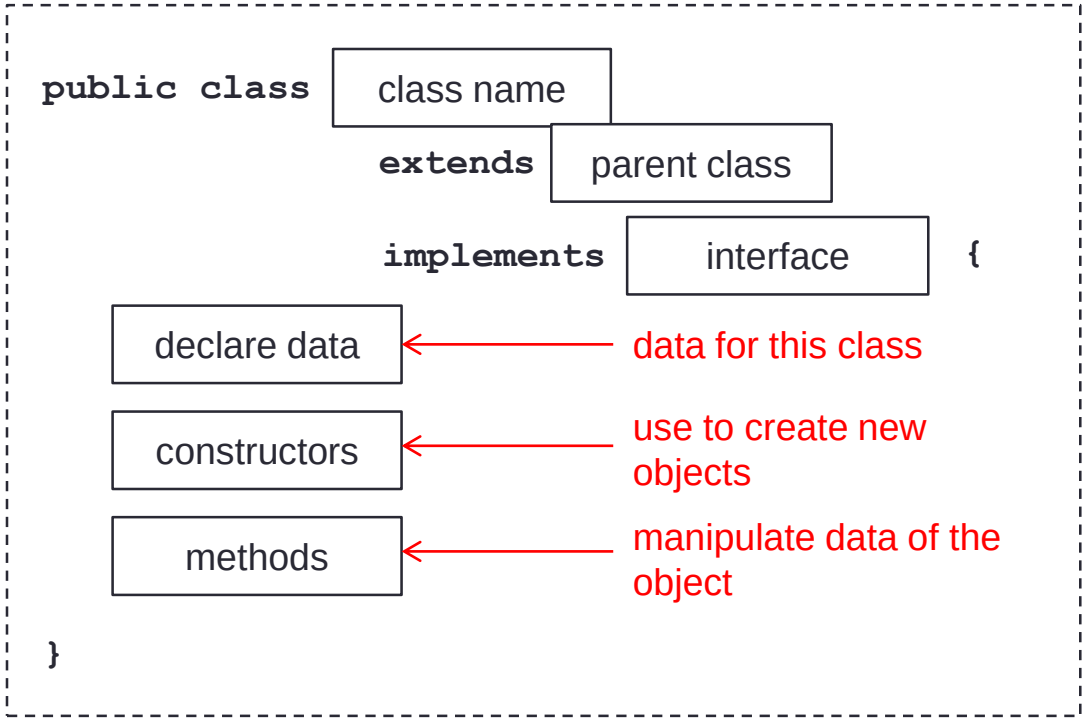
Student

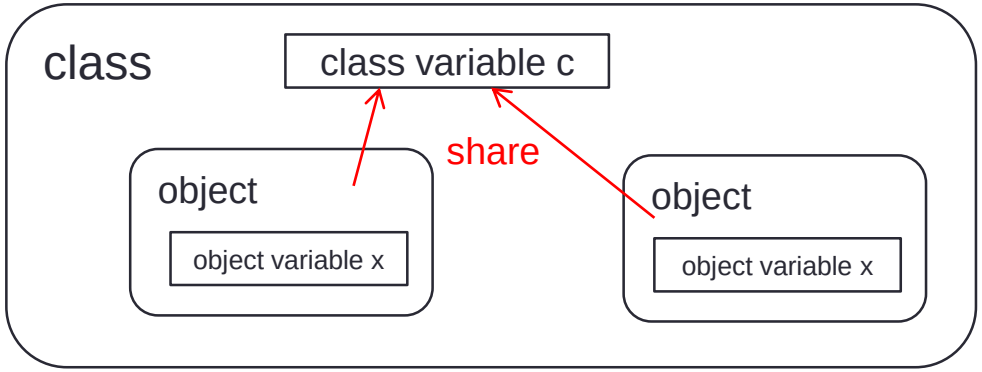
- Class for students
- Subclass of Human
- Holds a student id number as its data.

- Create a Student object.
- Use getName to get the name and print it out.

Java Class

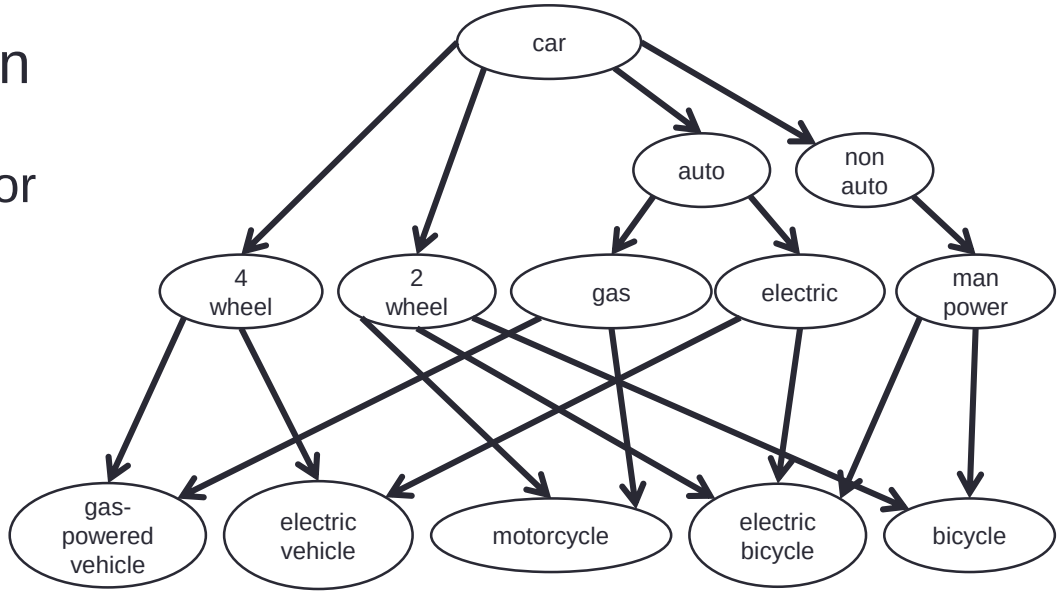
- Class declaration
  - specify parent class
  - specify implementing interface
  - declare data
  - declare constructors
  - declare methods
- Class variables
  - shared data among objects in the class
  - declare data with **`static`**
- Class methods
  - cannot refer objects
  - methods for the class



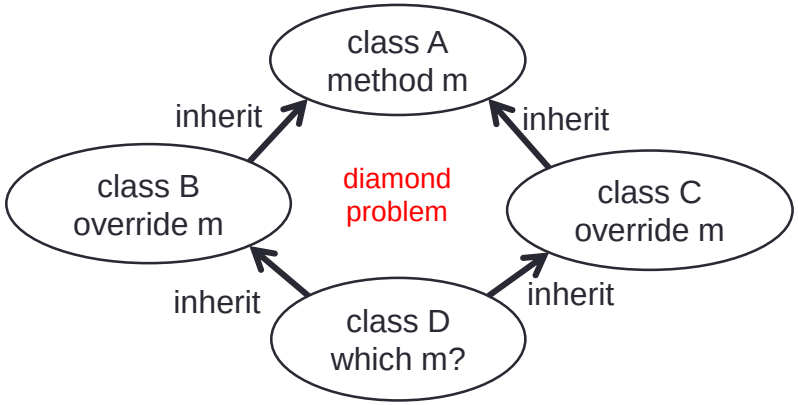


extends vs implements

- Can a class has more than one parent?
  - inherit properties from two or more parent classes.



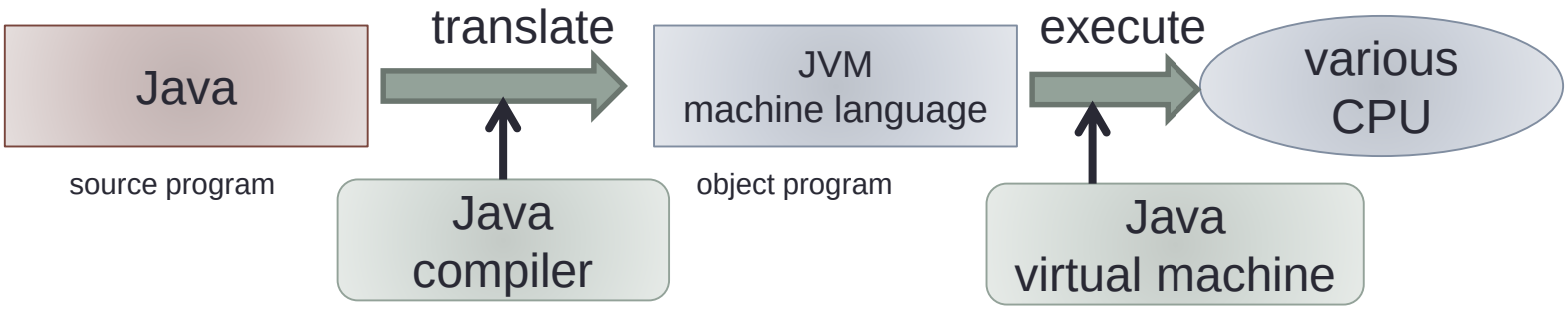
- Multiple inheritance
  - diamond problem
  - Which methods to inherit



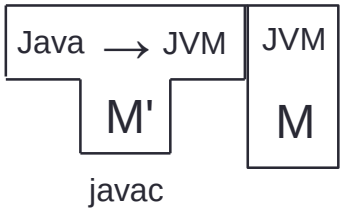
- Solution in Java
  - Only one parent (use extends)
  - Use interface to have multiple features.
    - interface = class without implementation

Java Virtual Machine

- Speed vs Portability
  - Compiler + Interpreter
  - Compile Java programs to JVM (Java Virtual Machine) machine programs
  - JVM can be implemented on various architectures.

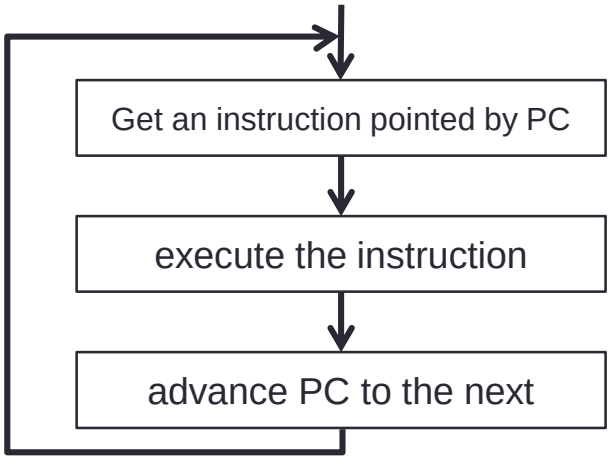


- Java Virtual Machine
  - Virtual (i.e. not physically exists) ideally thought CPU
  - Designed for Java compilation
  - Implements Java objects
  - Use GC to manage memory



Implementation of Virtual Machine

- Virtual Machine
  - Virtual CPU
  - Simpler than real CPU
  - Model ideal CPU
  - Usually byte machine
- Implement virtual machine
  - Emulate the virtual CPU



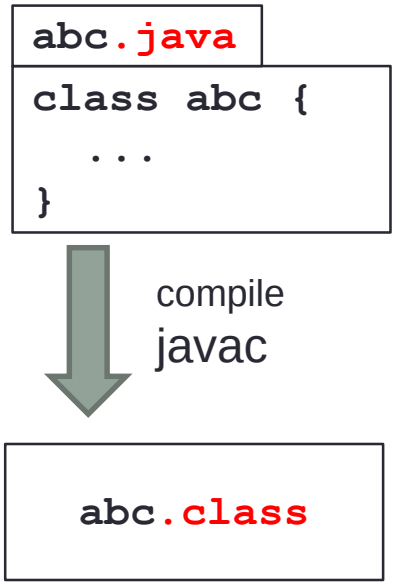

```
for (;;) {
  op = *pc++;
  switch (op) {
  case ILOAD:
    index = *pc++;
    *--sp = fp[index];
    break;
  case IADD:
    v1 = *sp++;
    v2 = *sp;
    *sp = v1 + v2;
    break;
  ....
  }
}
```

PC: Program Counter

Structure of Java Virtual Machine

- JVM codes are stored in a class file.
- Actual codes are stored in methods.

```
ClassFile {
  u4 magic;
  u2 minor_version;
  u2 major_version;
  u2 constant_pool_count;
  cp_info constant_pool[constant_pool_count-1];
  u2 access_flags;
  u2 this_class;
  u2 super_class;
  u2 interfaces_count;
  u2 interfaces[interfaces_count];
  u2 fields_count;
  field_info fields[fields_count];
  u2 methods_count;
  method_info methods[methods_count];
  u2 attributes_count;
  attribute_info attributes[attributes_count];
}
```

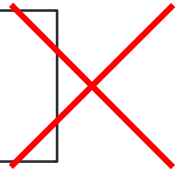


JVM Data Type

- Primitive data type
  - integer
  - floating point
  - boolean
  - returnAddress
- Reference data type
  - Objects

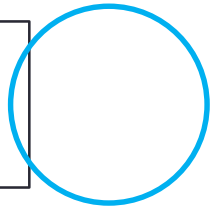
primitive data integer

```
int x = 1;
x.add(3);
```



object integer

```
Integer x = new Integer(1);
x.add(3);
```

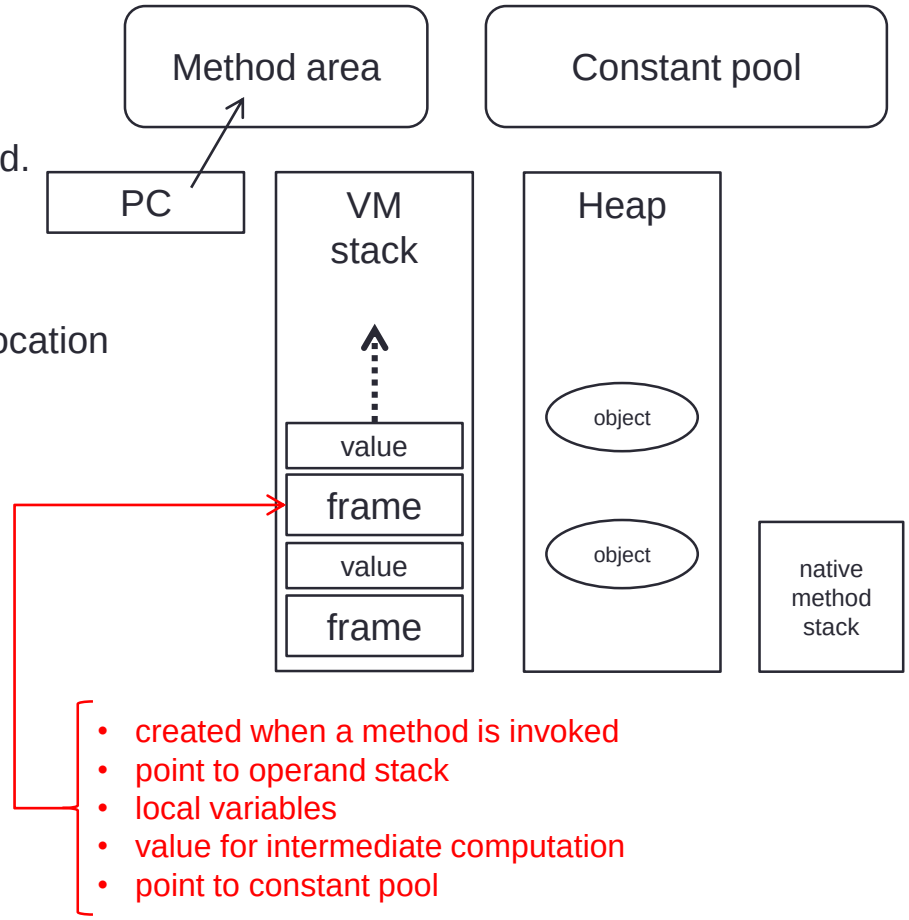


No add is actually implemented.

- Java is not pure object-oriented.
  - efficiency vs pure
  - Primitive data types are not objects.
  - Objects for primitive data

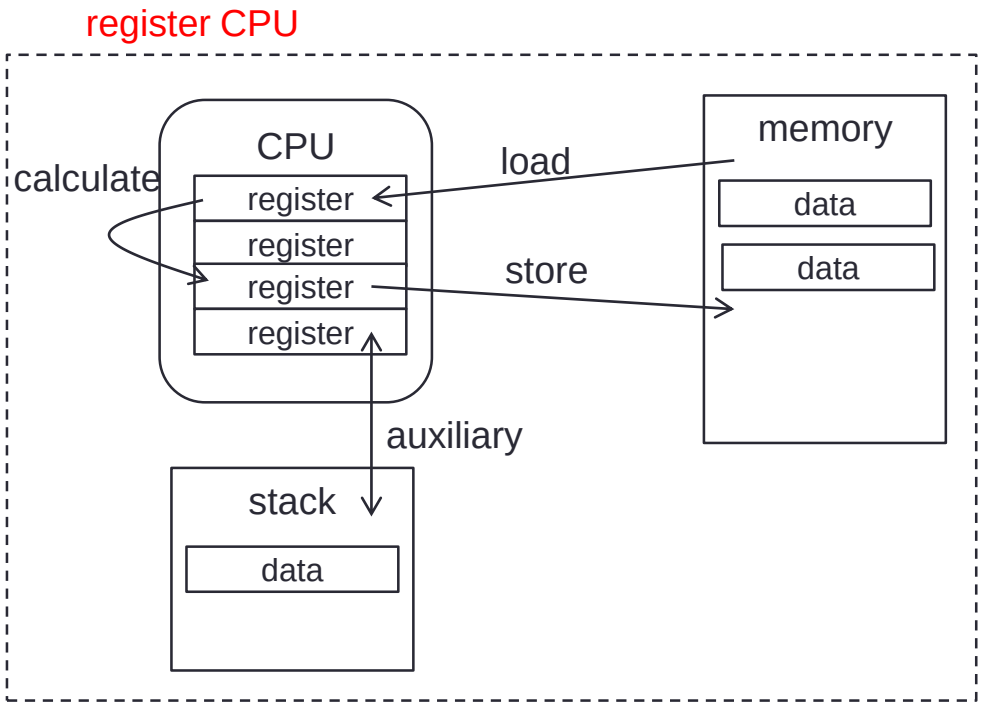
Structure of JVM

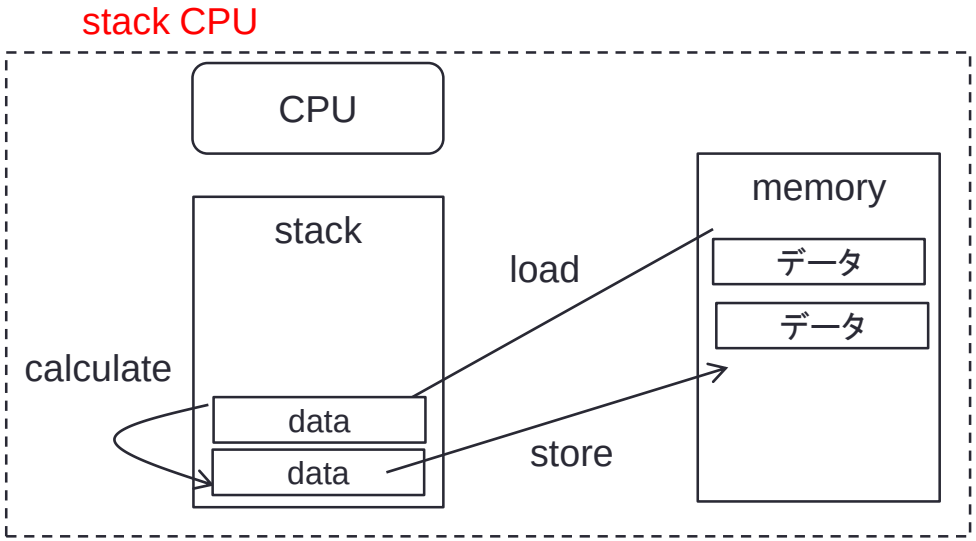
- PC register
  - Points to code of the currently executing method.
  - returnAddress data type
- Virtual machine stack
  - Holds frames which are created by method invocation
  - Pushed on to stack
  - Discarded when the method finishes.
- Heap
  - Shared by all the threads.
  - Holds objects and arrays.
- Method area
  - Shared by all the threads.
  - Holds codes for methods.
- Constant pool
  - Classes and interfaces have constant pool.
  - Constants and information of methods and fields are stored.
- Native method stack
  - Used when executing native codes.



CPU

- Register CPU
  - Calculation on registers
  - Accumulator is the most important register for calculation.
  - Data are loaded to registers, calculated, and stored to memory.
  - Stack may be used for complicated calculation.
- Stack CPU
  - CPU has no register.
  - Calculation on stack

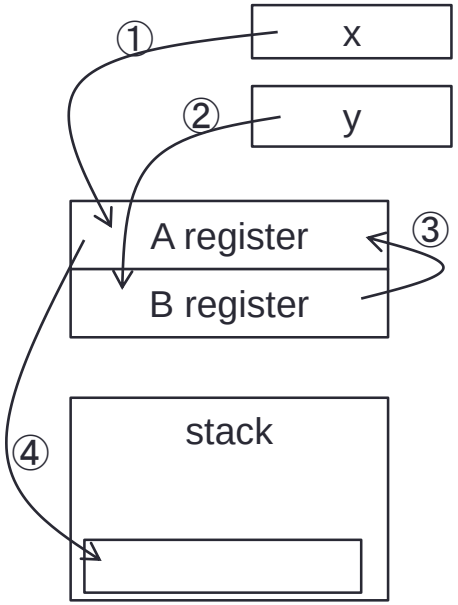




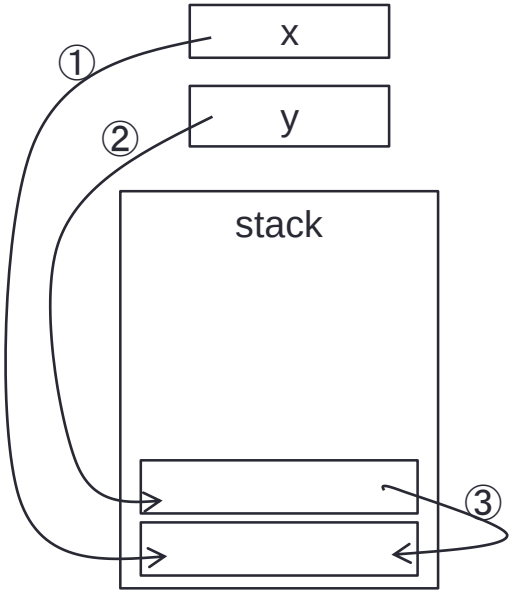
Register vs Stack

```
z = (x + y) * (u + w);
```

- register CPU
  - ① load x, A
  - ② load y, B
  - ③ add B, A
  - ④ push A
  - ⑤ load u, A
  - ⑥ load w, B
  - ⑦ add B, A
  - ⑧ pop B
  - ⑨ mult A, B
  - ⑩ store B, z



- stack CPU
  - ① load x
  - ② load y
  - ③ add
  - ④ load u
  - ⑤ load w
  - ⑥ add
  - ⑦ mult
  - ⑧ store z

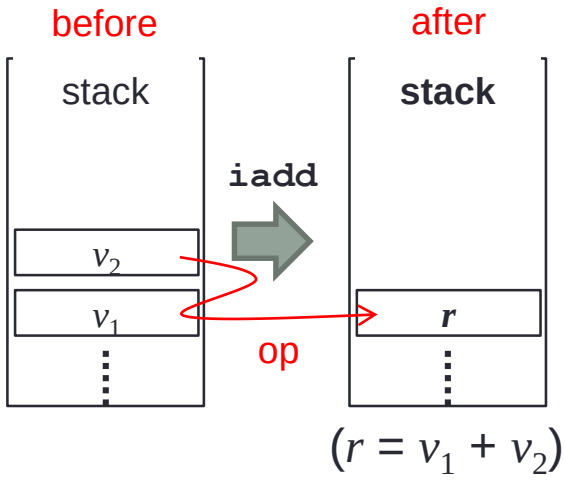


JVM Instruction (1)

- JVM instruction
  - Op code is one byte.
  - May have several operands.
  - There are 202 op codes defined (one is no used).
  - Three op codes are reserved.

- Integer addition: **`iadd`**
  - op code: **`iadd = 0x60`**
  - operand: none
  - stack operation:

    `iadd` : . . . , v_1, v_2 ⇒ . . . , r $(r = v_1 + v_2)$



JVM Instruction (2)

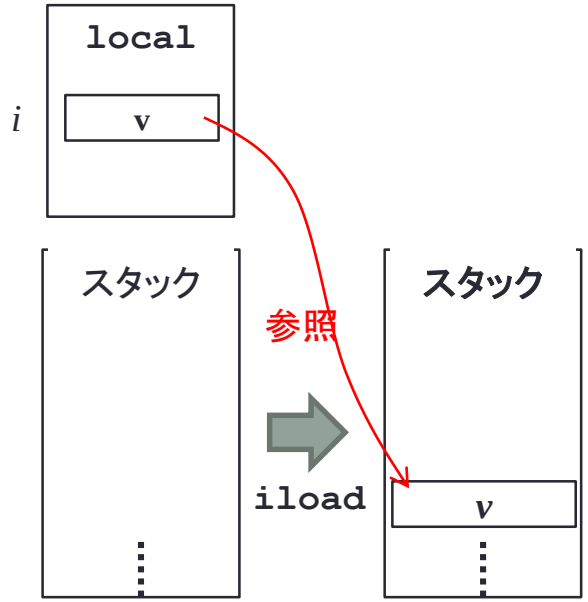
- Load value from a local variable
  - **`iload`** = **`0x15`**
  - operand: 1 byte, index

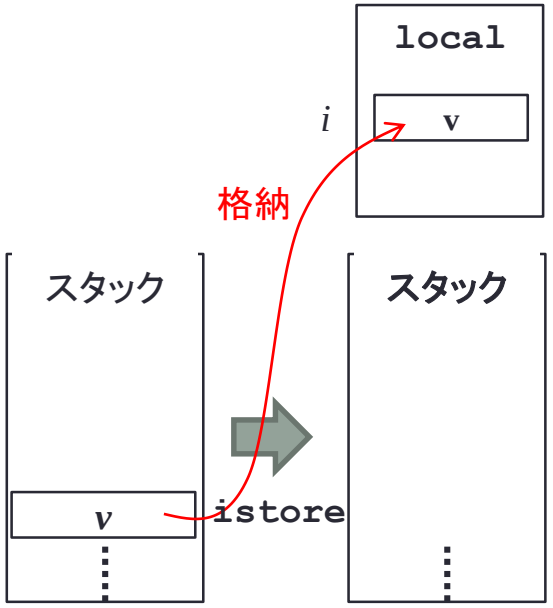
`iload`, i : . . . ⇒ . . . , **`local`**[i]

  - **`local`**: array of local variables
  - For small indexes, 0, 1, 2, and 3, there are special one byte op code: **`iload_0`**, **`iload_1`**, **`iload_2`** and **`iload_3`**

- Store value to a local variable
  - **`istore`** = **`0x36`**

`istore`, i : . . . , v ⇒ . . .
(**`local`**[i] ← v)



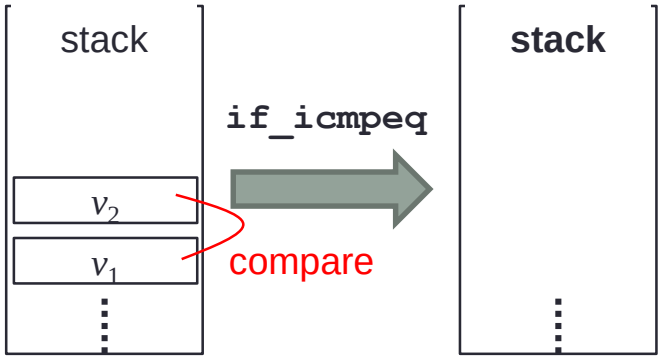


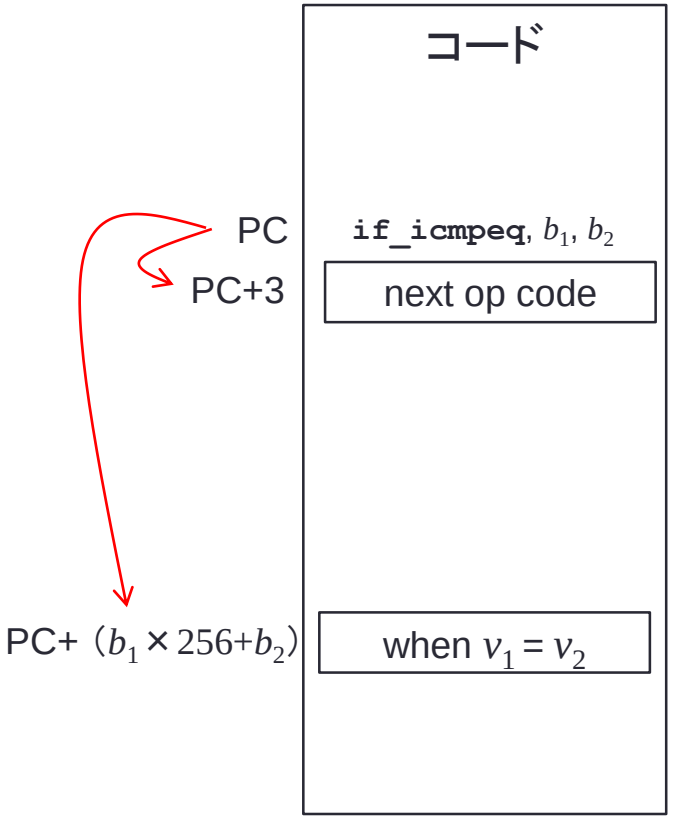
JVM Instruction (3)

- Conditional
  - **`if_icmpeq` = `0x9f`**
  - operand: 2 bytes, offset to the code to jump

$$\texttt{if\_icmpeq}, b_1, b_2 : \ldots, v_1, v_2 \Rightarrow \ldots$$

  - Pop v_1 and v_2 from the stack, compare them and if they are equal, add $(b_1 \times 256 + b_2)$ to PC
  - It they are not equal, just add 3 to PC to execute the next op code.





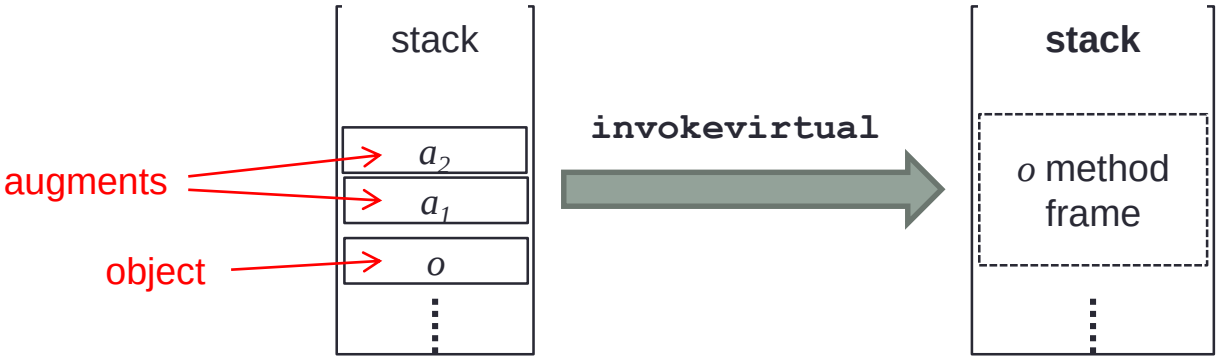
JVM Instruction (4)

- Method invovation
  - **`invokevirtual` = `0xb6`**

`invokevirtual`, i_1, i_2 : . . . , o, [a_1, [a_2 . . .]] ⇒ . . .

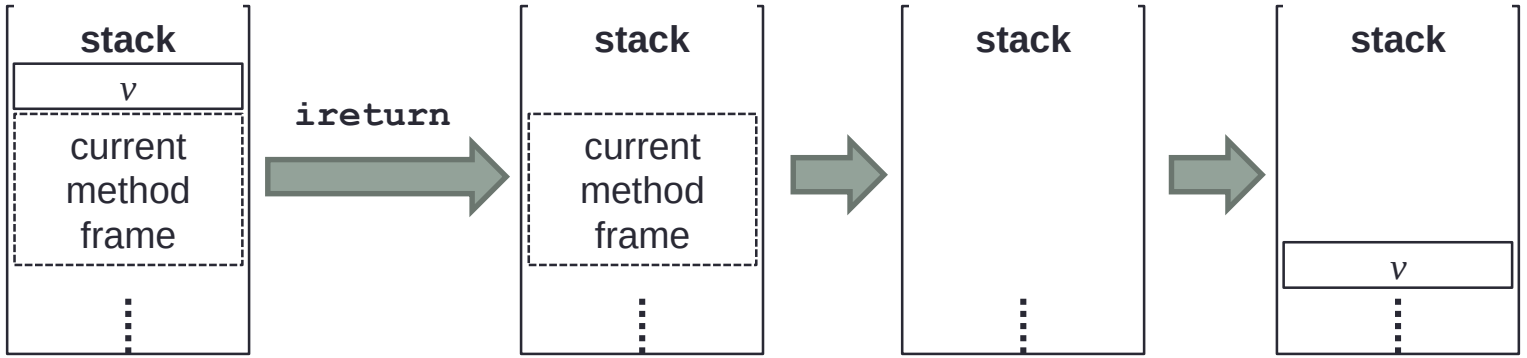
o : object
a_1, a_2, . . .: arguments to the method
$i_1 \times 256 + i_2$: index of constant pool
class name, method name, descriptor (number of arguments and their type)

  - Find a method from the class name, method name and descriptor.
  - Create a new frame with object o and arguments a_1, a_2, . . .
  - Set PC to the first op code of the method.
  - If **`synchronized`** is set, obtain the monitor for o.



JVM Instruction (5)

- Return from method
  - **`ireturn`** = **`0xac`** etc.
  - **`ireturn`** : . . . , v ⇒
  - Returns an integer value.
  - Pop v from the current frame.
  - Discard the current frame.
  - Push v to the caller frame.
  - Put back the previous PC.

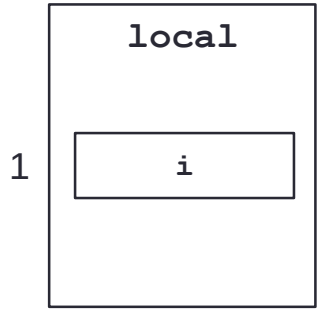


Compilation Example (1)

- Use ``**`javap -c`**'' to see the result of compilation.
- Compilation of a simple loop

```
void spin() {
  int i;
  for (i = 0; i < 100; i++) {
    ; // Loop body is empty
  }
}
```

```
Method void spin()
 0 iconst_0
 1 istore_1
 2 goto 8
 5 iinc 1,1
 8 iload_1
 9 bipush 100
11 if_icmplt 5
14 return
```



Compilation Example (2)

- Method with arguments

```
int addTwo(int i, int j) {
  return i + j;
}
```

```
Method int addTwo(int,int)
 0 iload_1
 1 iload_2
 2 iadd
 3 ireturn
```

- Method invocation

```
int add12and13() {
  return addTwo(12, 13);
}
```

```
Method int add12and13()
0 aload_0
1 bipush 12
3 bipush 13
5 invokevirtual #4 // Example.addTow(II)I
8 ireturn
```

- **`aload_0`**: push itself (**`this`**) to the stack
- Use **`invokestatic`** for class methods.

Compilation Example (3)

- Create an instance

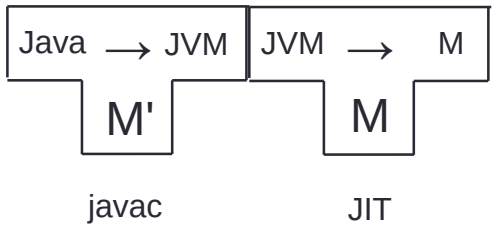
```
Object create() {
   return new Object();
}
```

```
Method java.lang.Object create()
 0 new #1 // Class java.lang.Object
 3 dup
 4 invokespecial #4 // Method java.lang.Object.<init>()V
 5 areturn
```

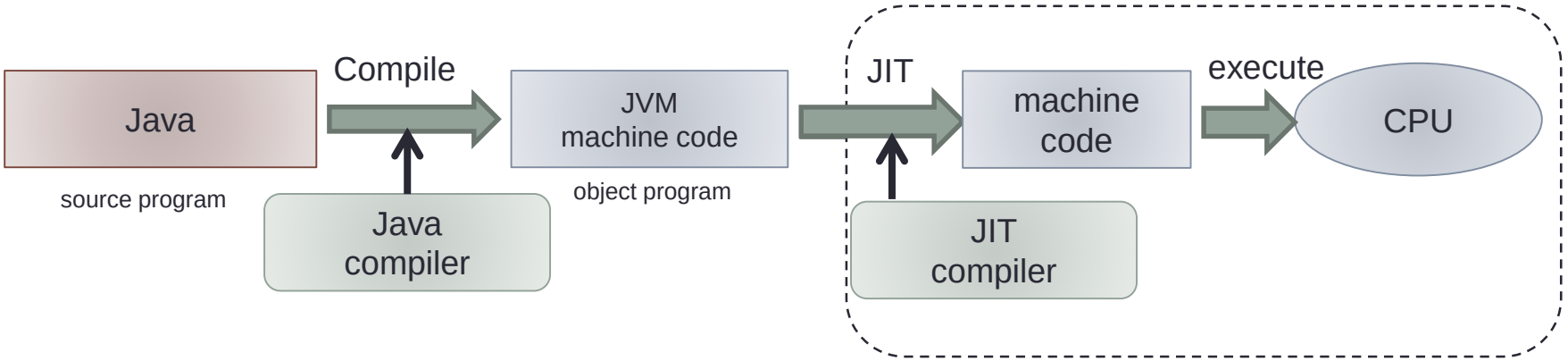
- **`new`** to create a new object.
- Invoke the parent constructor.
- **`areturn`** returns the created object.

JIT Compiler

- Merit and demerit of virtual machines
  - High portability
  - Slower than native machine codes



- JIT compiler
  - Just In Time
  - Translate into native machine codes when executed.
  - The first translation may take time, but the translated codes run at native speed.

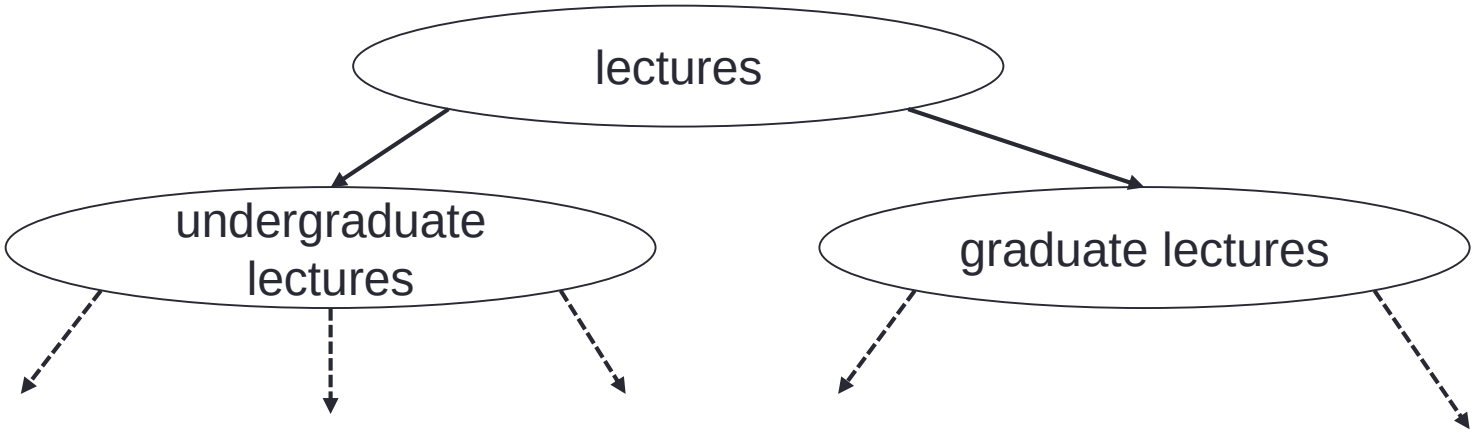


Summary

- Java
- Object-oriented Programming Languages
- Virtual machine
- Reference
  - The Java Virtual Machine Specification, Java SE 8 Edition
    - Tim Lindholm, Frank Yellin, Gilad Bracha, Alex Buckley
    - Addison-Wesley
  - https://docs.oracle.com/javase/specs/

Homework (7)

- Write class hierarchy of lectures in SFC.



- Deadline:
  - This Saturday