

i219 Software Design Methodology
5. Object-oriented
programming language 2

Kazuhiro Ogata (JAIST)

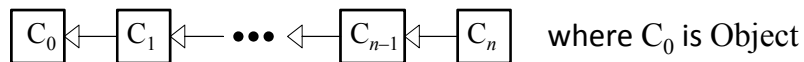
2

Outline of lecture

- Method dispatch
- Inner interface/class
- Enum type

Method dispatch (1)

For a while, let us only consider methods that have no arguments.
Under the condition, let us consider the class diagram:

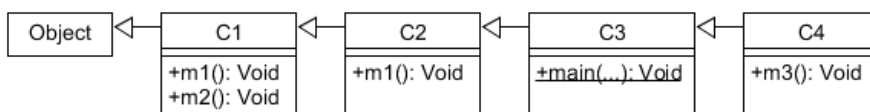


When a message $m()$ is sent to an object a_n of C_n , the search for the method invoked basically starts with C_n ; if there exists i ($0 \leq i \leq n$) such that C_i has the method $m()$, and each C_k ($i-1 \leq k \leq n$) does not have such a method, then the method in C_i is invoked.

Note that executing $o.m()$ where o is a variable of C_i ($i \neq 1$) but refers to an object of C_n , the search starts with C_n .

Note that compiling $o.m()$ fails if o is a variable of C_i such that C_0, \dots, C_i do not have $m()$; type casting may make the compilation successful.

Method dispatch (2)



`(new C4()).m1();` **m1() in C2 is invoked.**

`(new C4()).m2();` **m2() in C1 is invoked.**

`(new C4()).m3();` **m3() in C4 is invoked.**

`C1 o = new C4();`

`o.m1();` **m1() in C2 is invoked.**

`o.m2();` **m2() in C1 is invoked.**

`o.m3();` **A compiler error occurs.**

Method dispatch (3)

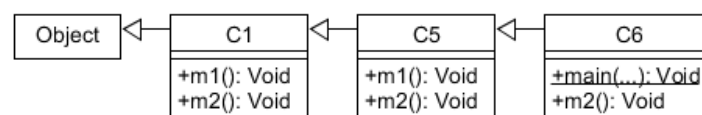
There are two pseudo-variables that refer to the currently active object: `this` and `super`.

Let both `this` and `super` refer to an object of C_n .

If a message `m()` is sent to `this`, the search for the method invoked starts with C_n as the message is sent to an object of C_n .

If a message `m()` is sent to `super`, the search for the method invoked starts with the super class of the class in which `super.m()` appears.

Method dispatch (4)



`(new C6()).m1();` `m1()` in C5 is invoked.

`m1()` in C5 is as follows:

```

public void m1() {
    System.out.println("m1() in C5 was invoked.");
    this.m2();              m2() in C6 is invoked.
    super.m2();            m2() in C1 is invoked.
}
  
```

Method dispatch (5)

A sequence τ_1, \dots, τ_m of types is compatible with a parametr $t_1 p_1, \dots, t_n p_n$ if $m = n$ and for each i ($1 \leq i \leq n$) τ_i is the same as t_i or a subtype of t_i .

E.g., C4, C4 is compatible with C3 x, C4 y.

$t_1 p_1, \dots, t_n p_n \leq t'_1 p'_1, \dots, t'_m p'_m$ if $m' = n$ and for each i ($1 \leq i \leq n$) t_i is the same as t'_i or a subtype of t'_i .

E.g., $x:C3, y:C4 \leq x:C2, y:C3$.

Note that if τ_1, \dots, τ_m is compatible with $t_1 p_1, \dots, t_n p_n$ and $t_1 p_1, \dots, t_n p_n \leq t'_1 p'_1, \dots, t'_m p'_m$ then τ_1, \dots, τ_m is compatible with $t'_1 p'_1, \dots, t'_m p'_m$.

E.g., C4, C4 is also compatible with $x:C2, y:C3$.

Method dispatch (6)

Let \mathbf{P} be a set of parameters with which τ_1, \dots, τ_m is compatible.

$P \in \mathbf{P}$ is called the minimum element of \mathbf{P} if $P \leq P'$ for all $P' \in \mathbf{P}$.

E.g., $\{(C3 x, C4 y), (C2 x, C3 y), (C3 x, C2 y), (C2 x, C2 y), (C1 x, C2 y)\}$ is a set of parameters with which C4, C4 is compatible. C3 x, C4 y is the minimum element of the set.

E.g., $\{(C2 x, C2 y), (C1 x, C2 y)\}$ is a set of parameters with which C2, C2 is compatible. C2 x, C2 y is the minimum element of the set.

E.g., $\{(C2 x, C3 y), (C3 x, C2 y), (C2 x, C2 y), (C1 x, C2 y)\}$ is a set of parameters with which C3, C3 is compatible. The set does not have the minimum element .

Method dispatch (6)

Let us consider $obj.m(v_1, \dots, v_m)$ that appears in C_k .

For each v_j , let t_j be its type that can be known at compile time and C_{obj} be the class of obj that can be known at compile time.

E.g., if v_j (or obj) is `new C2()`, t_j (or C_{obj}) is C_2 , if v_j (or obj) is x that is a variable declared as C_1 x , then t_j (or C_{obj}) is C_1 even though an object of C_2 is stored in x , if v_j (or obj) is the pseudo-variable `this`, t_j (or C_{obj}) is C_k , and if v_j is `3.14`, t_j is `double`.

If obj is the pseudo-variable `super`, C_{obj} is C_{k-1} (the super class of C_k). Note that `super` cannot be used as an actual parameter of a method, such as v_j .

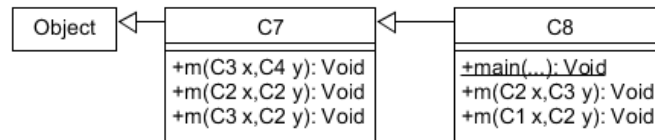
Method dispatch (6)

Let \mathbf{M} be the set of methods in C_0, \dots, C_{obj} such that their names are the same as m , and \mathbf{P} be $\{P \mid m(P) \in \mathbf{M}, t_1, \dots, t_n \text{ is compatible with } P\}$.

If \mathbf{P} has the minimum element P_{\min} , then the method signature $m(P_{\min})$ is bound to the message passing statement $obj.m(v_1, \dots, v_m)$ at compile time. Otherwise, an error occurs.

The method dispatch for $obj.m(v_1, \dots, v_m)$ at runtime is done in the same way as that for $obj.m()$ where m has no parameters. The method signature of the method invoked for the latter case is $m()$, while that for the former is $m(P_{\min})$.

Method dispatch (7)



```
(new C8()).m(new C4(),new C4());
```

m(C3 x,C4 y) in C7 is invoked.

```
C2 x = new C4();
```

```
C2 y = new C4();
```

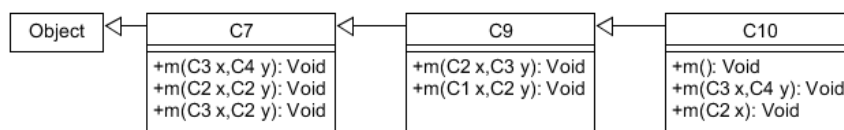
```
(new C8()).m(x,y);
```

m(C2 x,C2 y) in C7 is invoked.

```
(new C8()).m(new C3(),new C3());
```

A compiler error occurs.

Method dispatch (8)



```
(new C10()).m();
```

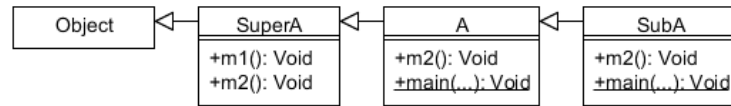
m() in C10 is invoked.

m1() in C10 is as follows:

```

public void m() {
    System.out.println("m) in C10 was invoked.");
    this.m(new C4(),new C4());;    m(C3 x,C4 y) in C10 is invoked.
    super.m(new C4(),new C4());;    m(C3 x,C4 y) in C7 is invoked.
    this.m(new C3());;              m(C2 x) in C10 is invoked.
    // super.m(new C3());; if uncommented, javac complains.
}
  
```

Method dispatch (9)



if this refers to an object of SubA, A & SuperA,
then m2() in SubA, A & SuperA are invoked, respectively

```
public void m1() { System.out.print("m1() in SuperA;"); this.m2(); }
public void m2() { System.out.println("m2() in SuperA;"); }
```

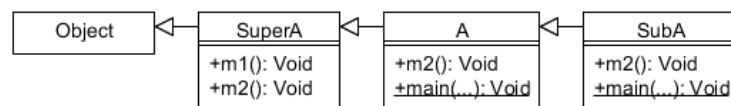
```
public void m2() { System.out.print("m2() in A;"); super.m2(); }
```

m2() in SuperA is invoked

```
public void m2() { System.out.print("m2() in SubA;"); super.m2(); }
```

m2() in A is invoked

Method dispatch (10)



```
public void m1() { System.out.print("m1() in SuperA;"); this.m2(); }
```

```
public void m2() { System.out.println("m2() in SuperA;"); }
```

```
public void m2() { System.out.print("m2() in A;"); super.m2(); }
```

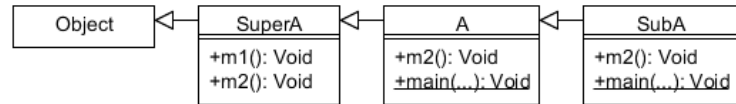
```
public void m2() { System.out.print("m2() in SubA;"); super.m2(); }
```

main(...) in A is as follows:

```
public static void main(String[] args) { (new A()).m1(); }
```

m1() in SuperA;m2() in A;m2() in SuperA;

Method dispatch (11)



```
public void m1() { System.out.print("m1() in SuperA;"); this.m2(); }
```

```
public void m2() { System.out.println("m2() in SuperA;"); }
```

```
public void m2() { System.out.print("m2() in A;"); super.m2(); }
```

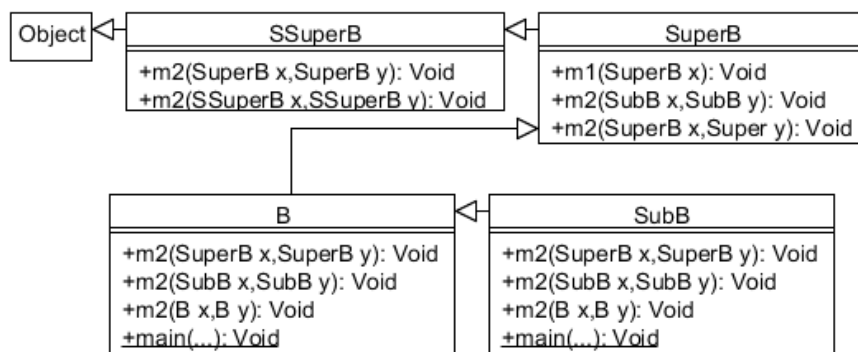
```
public void m2() { System.out.print("m2() in SubA;"); super.m2(); }
```

main(...) in SubA is as follows:

```
public static void main(String[] args) { (new SubA()).m1(); }
```

m1() in SuperA;m2() in SubA;m2() in A;m2() in SuperA;

Method dispatch (12)



Method dispatch (13)

In SSuperB:

```
m2(SuperB x,SuperB y)
{ ...println("m2(SuperB x,SuperB y) in SSuperB;"); }
m2(SSuperB x,SSuperB y)
{ ...println("m2(SSuperB x,SSuperB y) in SSuperB;"); }
```

In SuperB:

```
m1(SuperB x) { ...print("m1(SuperB x) in SuperB;"); AMsgPassing }
m2(SubB x,SubB y) { ...println("m2(SubB x,SubB y) in SuperB;"); }
m2(SuperB x,SuperB y)
{ ...println("m2(SuperB x,SuperB y) in SuperB;"); }
```

Method dispatch (14)

In B:

```
m2(SuperB x,SuperB y) { ...print("m2(SuperB x,SuperB y) in B;");
  super.m2(new B(),new B()); }
m2(SubB x,SubB y) { ...print("m2(SubB x,SubB y) in B;");
  super.m2(new B(),new B()); }
m2(B x,B y) { ...print("m2(B x,B y) in B;"); super.m2(new B(),new B()); }
main(String[] args) { (new B()).m1(new B()); }
```

In SubB:

```
m2(SuperB x,SuperB y) { ...print("m2(SuperB x,SuperB y) in SubB;");
  super.m2(x,y); }
m2(SubB x,SubB y) { ...print("m2(SubB x,SubB y) in SubB;");
  super.m2(x,y); }
m2(B x,B y) { ...print("m2(B x,B y) in SubB;");
  super.m2(x,y); }
main(String[] args) { (new SubB()).m1(new B()); }
```

Method dispatch (15)

1. *AMsgPassing* is (new SubB()).m2(new B(),new B());

```
% java B
m1(SuperB x) in SuperB;m2(B x,B y) in SubB;m2(B x,B y) in B;m2(SuperB
x,SuperB y) in SuperB;
```

```
% java SubB
m1(SuperB x) in SuperB;m2(B x,B y) in SubB;m2(B x,B y) in B;m2(SuperB
x,SuperB y) in SuperB;
```

2. *AMsgPassing* is this.m2(new B(),new B());

```
% java B
m1(SuperB x) in SuperB;m2(SuperB x,SuperB y) in B;m2(SuperB x,SuperB y) in
SuperB;
```

```
% java SubB
m1(SuperB x) in SuperB;m2(SuperB x,SuperB y) in SubB;m2(SuperB x,SuperB y)
in B;m2(SuperB x,SuperB y) in SuperB;
```

Method dispatch (16)

3. *AMsgPassing* is this.m2(new SubB(),new SubB());

```
% java B
m1(SuperB x) in SuperB;m2(SubB x,SubB y) in B;m2(SuperB x,SuperB y) in
SuperB;
```

```
% java SubB
m1(SuperB x) in SuperB;m2(SubB x,SubB y) in SubB;m2(SubB x,SubB y) in
B;m2(SuperB x,SuperB y) in SuperB;
```

4. *AMsgPassing* is this.m2(this,this);

```
% java B
m1(SuperB x) in SuperB;m2(SuperB x,SuperB y) in B;m2(SuperB x,SuperB y) in
SuperB;
```

```
% java SubB
m1(SuperB x) in SuperB;m2(SuperB x,SuperB y) in SubB;m2(SuperB x,SuperB y)
in B;m2(SuperB x,SuperB y) in SuperB;
```

Method dispatch (17)

5. *AMsgPassing* is `SuperB o = new SubB(); o.m2(new B(),new B());`;

```
% java B
m1(SuperB x) in SuperB;m2(SuperB x,SuperB y) in SubB;m2(SuperB x,SuperB y)
in B;m2(SuperB x,SuperB y) in SuperB;
```

```
% java SubB
```

```
m1(SuperB x) in SuperB;m2(SuperB x,SuperB y) in SubB;m2(SuperB x,SuperB y)
in B;m2(SuperB x,SuperB y) in SuperB;
```

6. *AMsgPassing* is `SuperB o1 = new SubB(); o1.m2(new SSuperB(),new B());`;

```
% java B
m1(SuperB x) in SuperB;m2(SSuperB x,SSuperB y) in SSuperB;
```

```
% java SubB
```

```
m1(SuperB x) in SuperB;m2(SSuperB x,SSuperB y) in SSuperB;
```

Method dispatch (18)

7. *AMsgPassing* is `this.m2(new SSuperB(),new B());`;

```
% java B
m1(SuperB x) in SuperB;m2(SSuperB x,SSuperB y) in SSuperB;
```

```
% java SubB
```

```
m1(SuperB x) in SuperB;m2(SSuperB x,SSuperB y) in SSuperB;
```

8. *AMsgPassing* is `this.m2(new Object(),new B());`;

javac complains.

Inner class/interface

Classes & interfaces can be declared inside a class; called inner classes & interfaces.

```
public class NestedClass {
    private interface MyInterface { String m(); }
    private class MyClass implements MyInterface
    { public MyClass() {}
      public String m() { return "MyClass"; } }
    private MyClass o = new MyClass();
    public String m() { return "I have " + o.m() + "."; } }
```

←----- an inner interface

←----- an inner class

Compiling NestedClass (NestedClass.java), in addition to NestedClass.class, NestedClass\$MyInterface.class & NestedClass\$MyClass.class are made for the inner interface & class.

Enum type (1)

An enum type is a special class of Java.

```
public enum TokenName { PLUS, MUL, NUM, UNDEF, }
```

↑
instances of TokenName; (called enum constants)
no other instances of TokenName

An enum constant *EC* in an enum type *ET* is referred as *ET.EC*.

```
public String toString() throws IllegalStateException {
    if (this.name == TokenName.PLUS) { return "plus"; }
    else if (this.name == TokenName.MUL) { return "mul"; }
    else if (this.name == TokenName.NUM) { return "num=" + num; }
    else if (this.name == TokenName.UNDEF) { return "undef:" + undef; }
    else { throw new IllegalStateException("in Token1!"); } }
```

↑
checks if all cases are covered at runtime

Enum type (2)

When an enum type *ET* is used in switch, however, an enum constant *EC* should be referred as *EC* instead of *ET.EC*.

```
public String toString() throws IllegalStateException {
    switch(this.name) {
        case PLUS: return "plus";
        case MUL: return "mul";
        case NUM: return "num=" + num;
        case UNDEF: return "undef:" + undef;
        default: throw new IllegalStateException("in Token2!"); } }
```

Using an enum type in switch, an inner class is made by javac. This is why `Token2$1.class` is generated when compiling `Token2.java`.

Summary

- Method dispatch
- Inner interface/class
- Enum type