03—Object-oriented Programming in Java

CS1102S: Data Structures and Algorithms

Li Hongyang and Martin Henz

January 21, 2009

Generated on Thursday 21st January, 2010, 11:13

03—Object-oriented Programming in Java

CS1102S: Data Structures and Algorithms

Li Hongyang and Martin Henz

January 21, 2009

Generated on Thursday 21st January, 2010, 11:13

Knowledge Representation View of Objects

- Aggregation/Components
- Classification/Instantiation
- Generalization/Specialization

Aggregation/Components

- Data items can be grouped into bigger ones, while preserving their functionality
- Example:

```
class ScollableTextArea {
    Scrollbar x_scrollbar;
    Scrollbar y_scrollbar;
    TextArea textarea;
    ...
}
```

Classification/Instantiation

- Data structures (objects) are distinguished from their classes
- Objects are created using a new operator on classes

```
class Vehicle {
    public Vehicle(int maxSpeed) {...}
}
...
Vehicle myVehicle = new Vehicle(125);
end
```

Generalization/Specialization

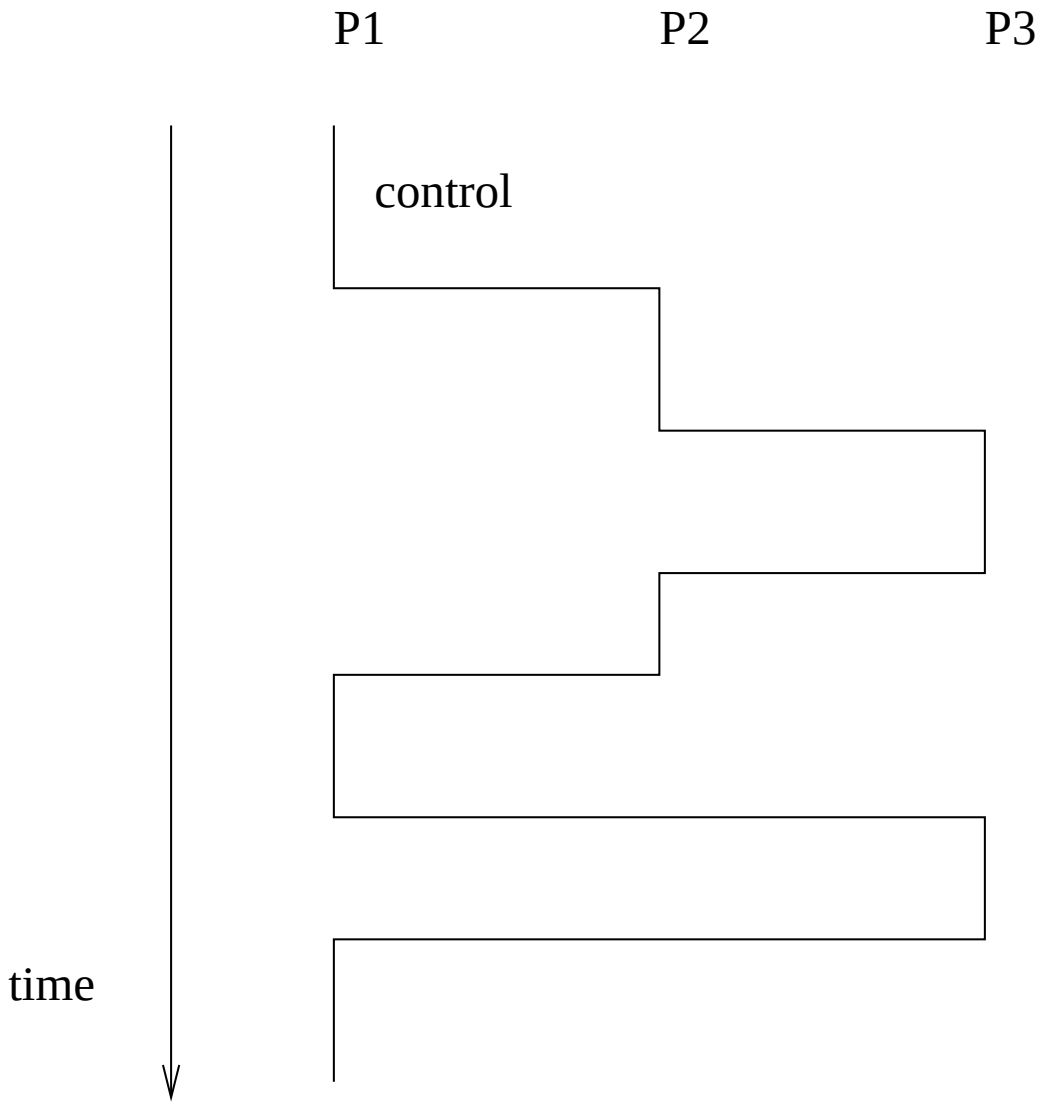
- Add/modify behavior by inheriting a class and (re-)defining methods
- Example:

```
class DecoratedWindow extends Window {
    void draw() {
        super.draw();
        drawDecoration();
    }
    ...
}
```

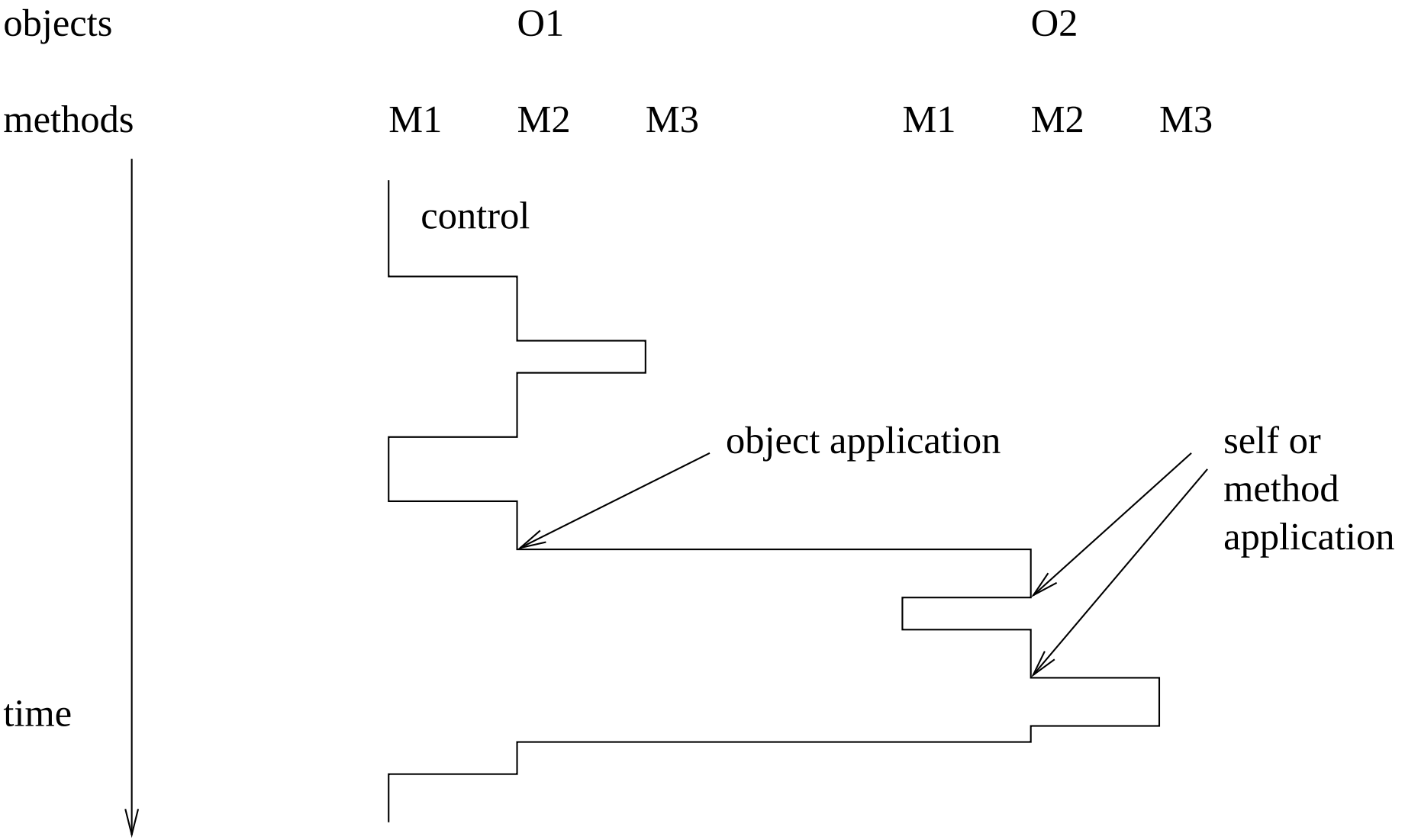
Object-oriented Programming

- Unifies three composition dimensions into a coherent software methodology
  - Aggregation/Components
  - Classification/Instantiation
  - Generalization/Specialization
- SIMULA was developed in mid 1960s in Norway
- Concepts of SIMULA used in Smalltalk (1980s)
- C++ (1980s), Java (1990s)

Control Flow in Procedural Languages (time)

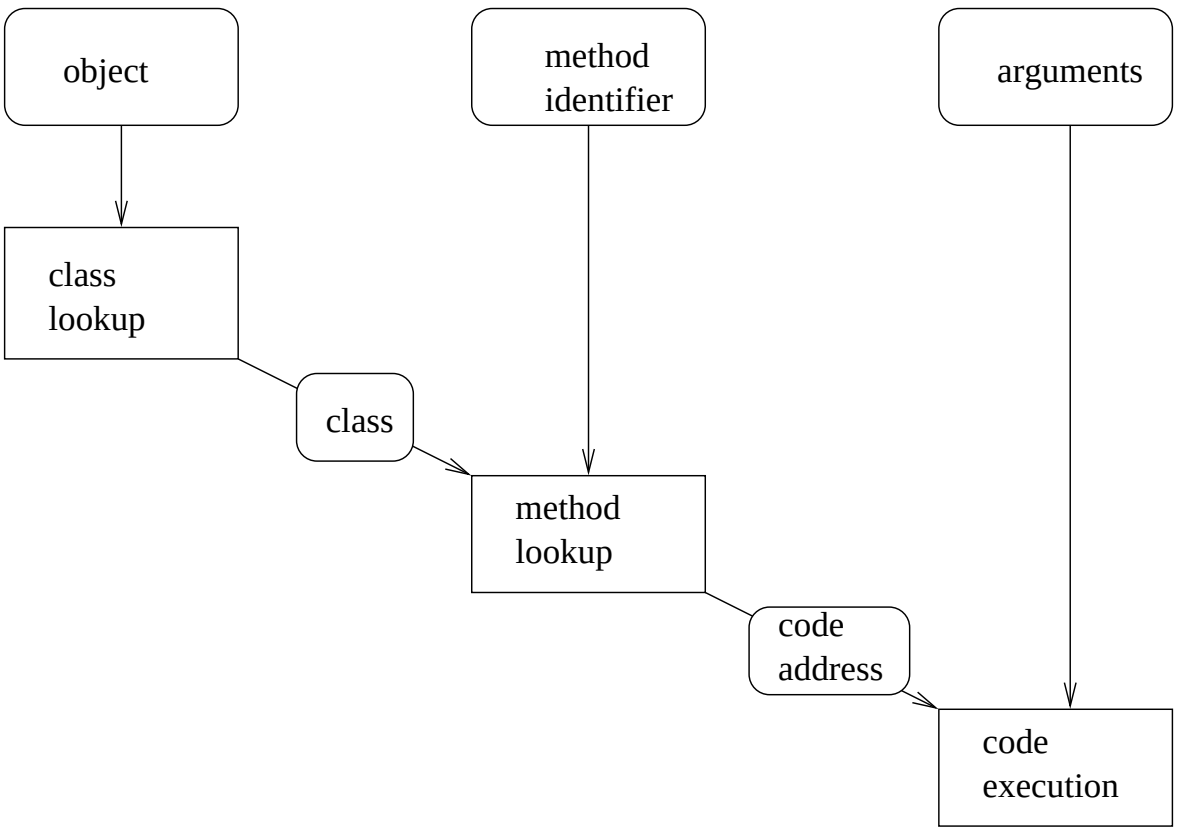


Control Flow in Object-oriented Languages (time)

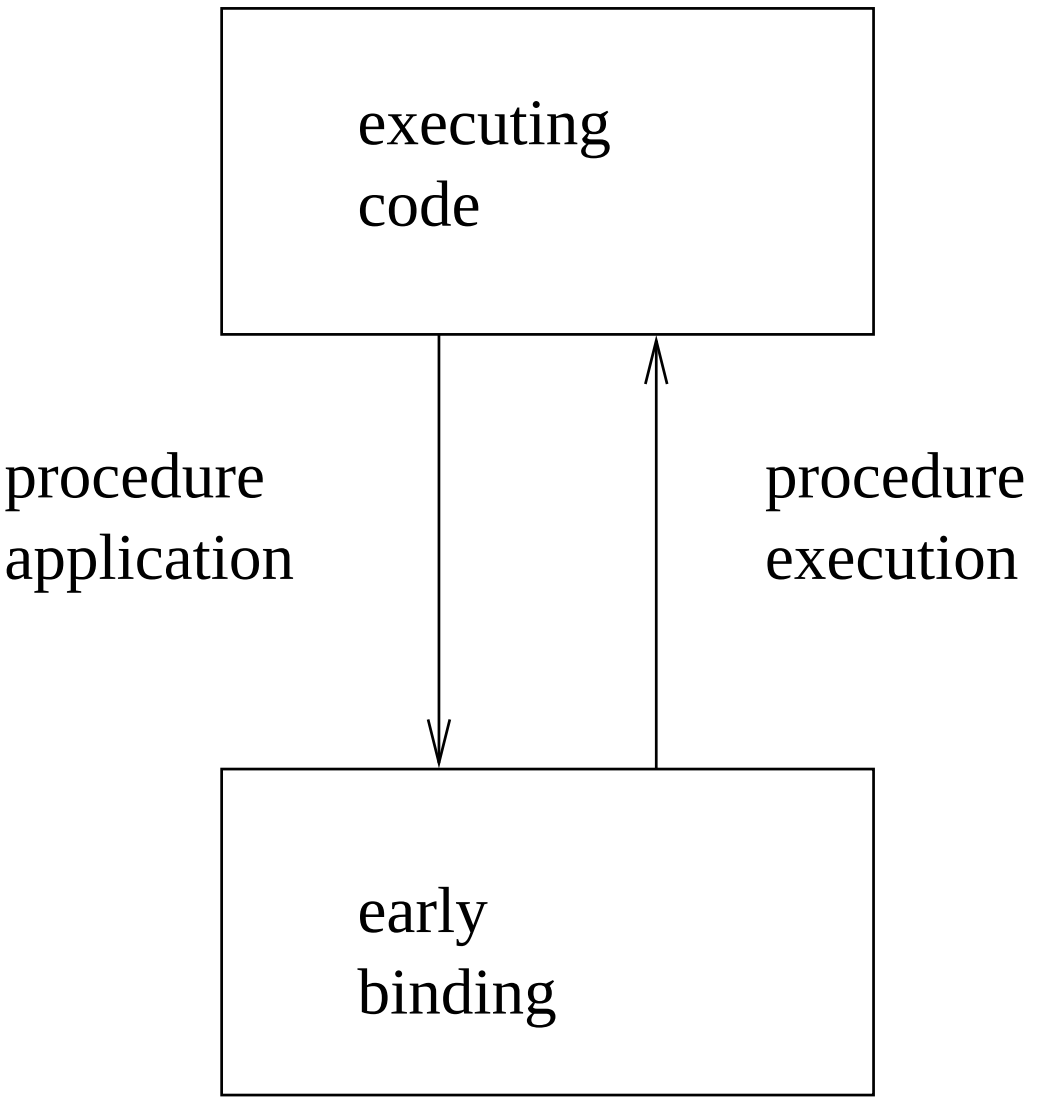


Execution of Object Application

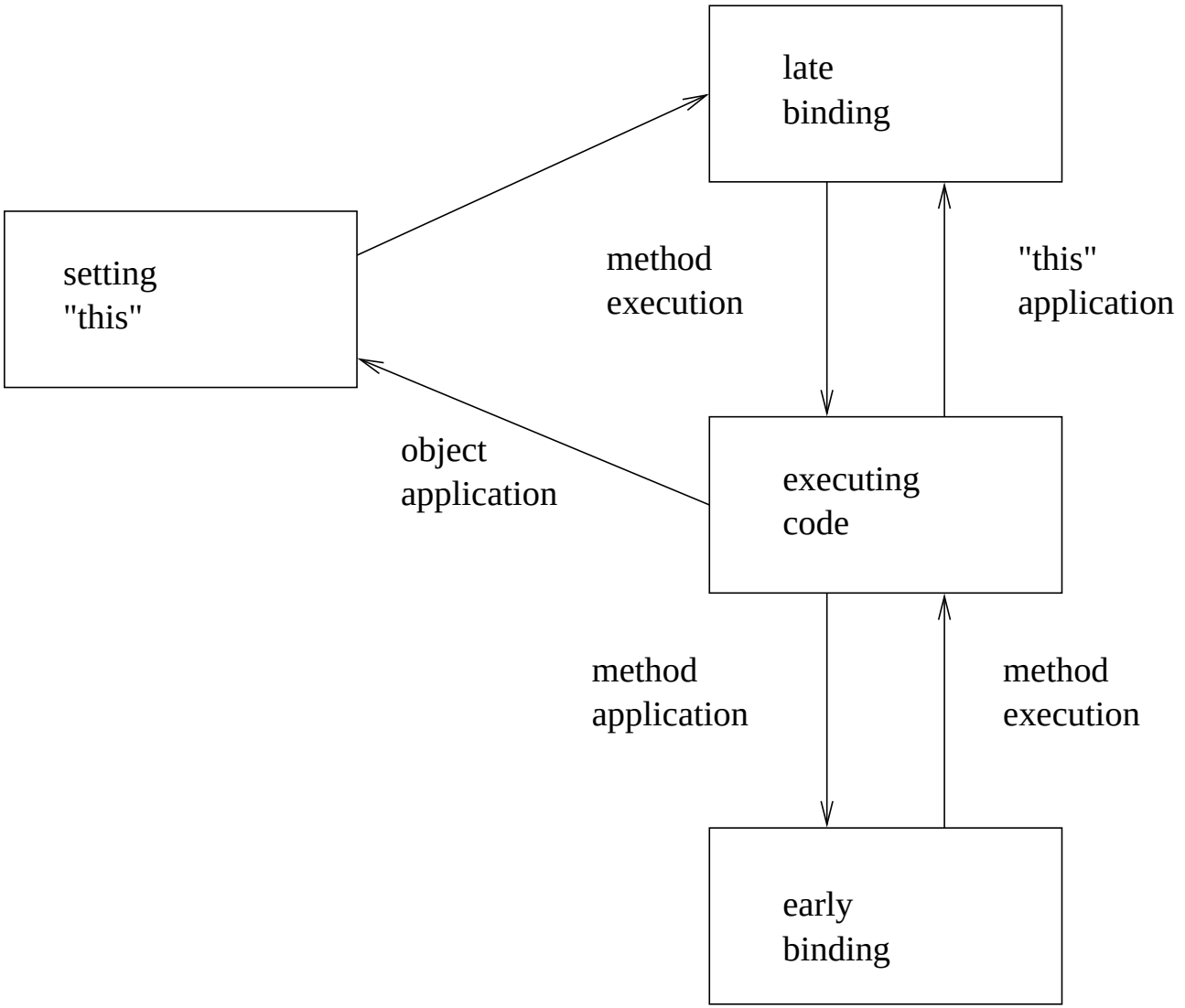
```
window . move ( 100 , 50 )
```



Early Binding in Procedural Languages



Late Binding in Object-oriented Languages



Overriding of Methods

- In Java, a method can be "overridden", if the parameter types are exactly the same.
- Example:

```
class A {
    public m(C c) {
        System.out.println("A");
} }
class B {
    public m(C c) {
        System.out.println("B");
} }
...
B myObject = new B();
myObject.m(new C());
```

Method Dispatching

Method dispatching uses:

- the class of the actual object
- the descriptor of the method, consisting of:
  - the name of the method
  - the *types* of the arguments

Example

```
class A {
    public int m(Person P) {
        System.out.println("1");
} }
class B extends A {
    public int m(Student P) {
        System.out.println("2");
} }
B myB = new B();
Person p = new Student();
myB.m(p); // prints 1
```

Java “Specialty”: Co-variance of Array Types

(see page 15)

```
Person [] arr = new Employee[ 5 ];
arr[ 0 ] = new Student( ... );
```

- This program compiles, but
- produces a runtime error ArrayStoreException