Inheritance and Polymorphism

Auxiliar 2 - juampi

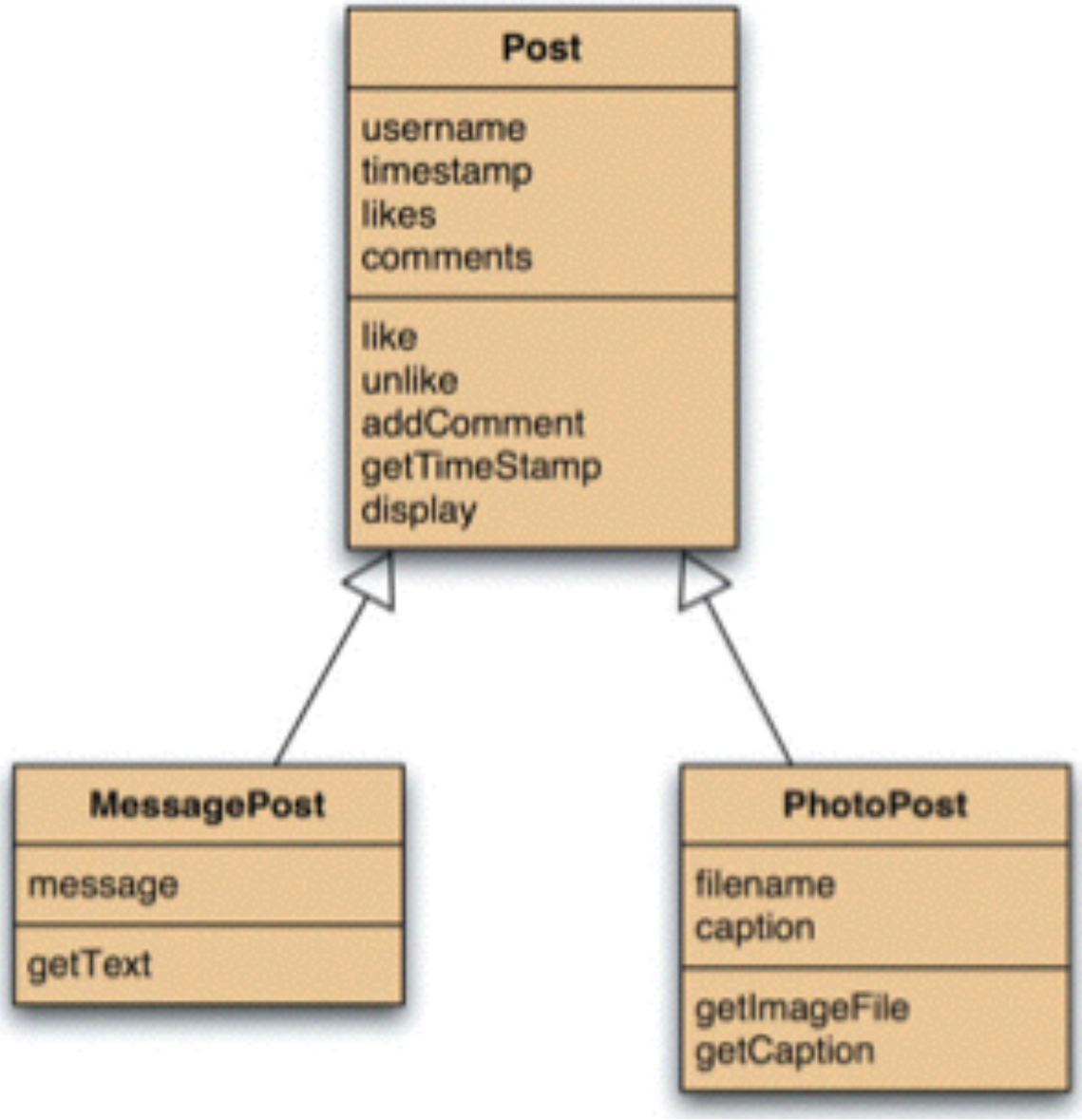
Inheritance and Polymorphism

- Inheritance allow us to define one class as an extension of another.

- A superclass is a class that is extended by another class.

- A subclass is a class that extends (inherits from) another class. It inherits all fields and methods from its superclass.

Inheritance and Polymorphism



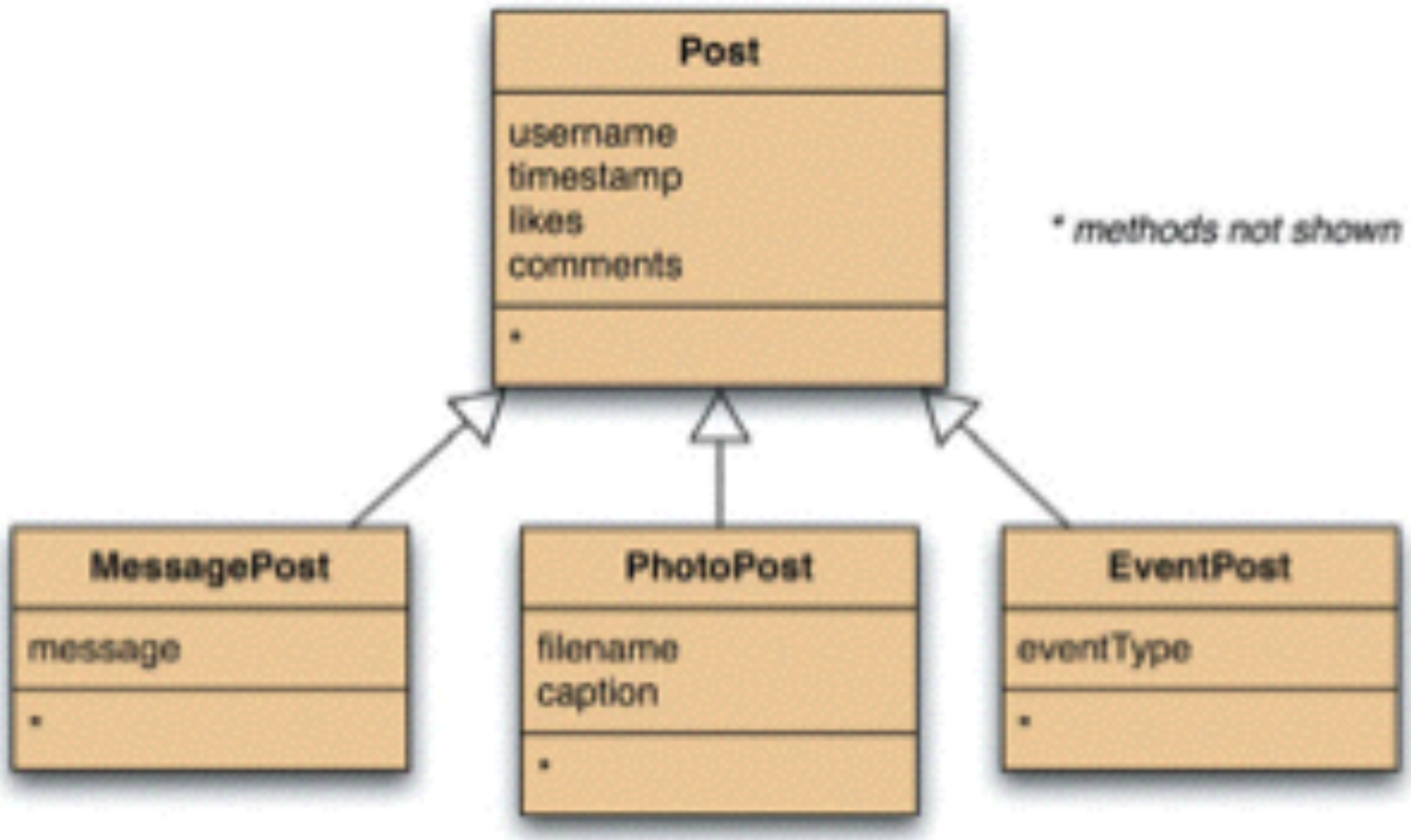
Inheritance in Java

```
public class Post
{
    private String username;  // username of the post's author
    private long timestamp;
    private int likes;
    private ArrayList<String> comments;
    // Constructors and methods omitted.
}
```

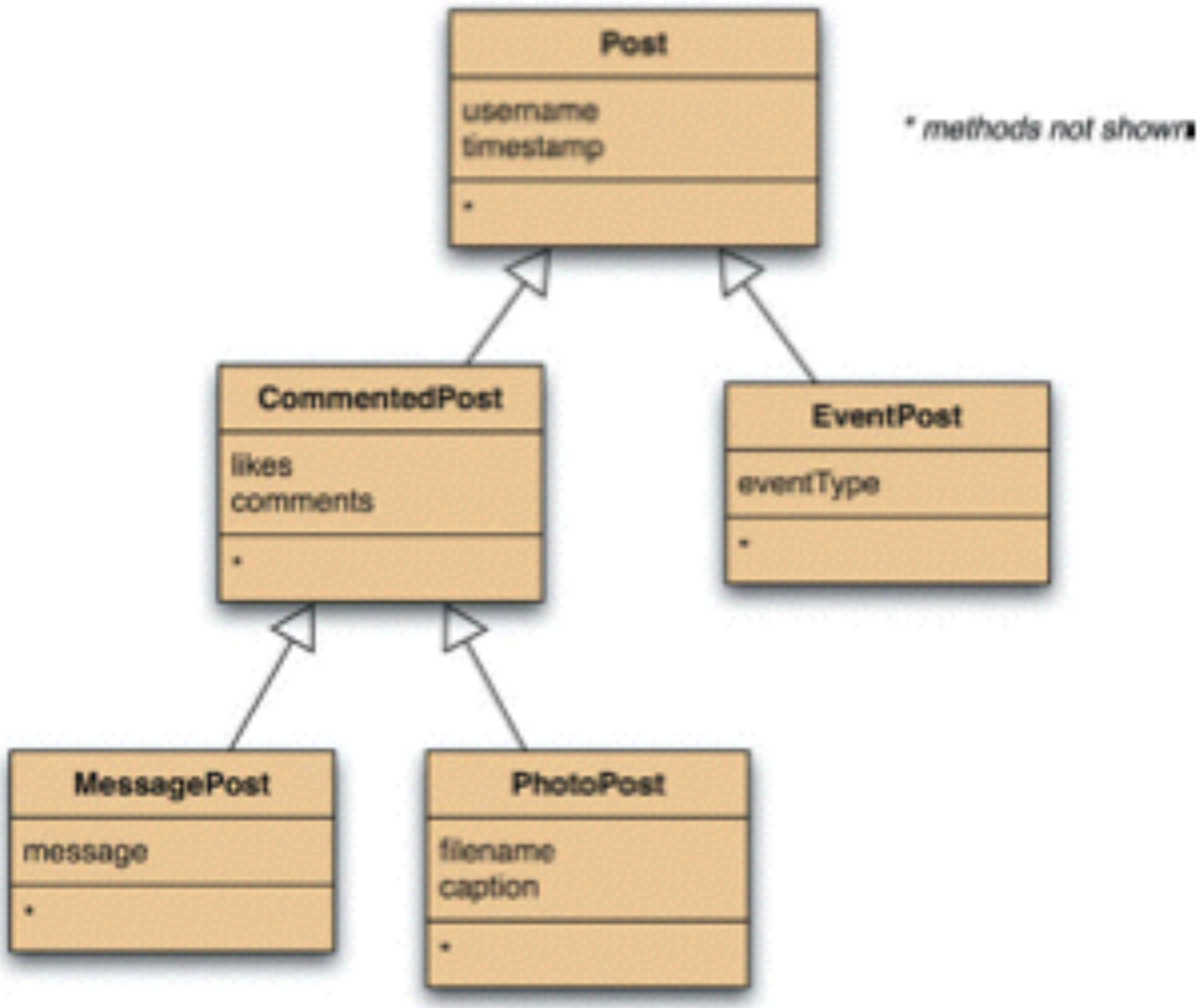
```
public class MessagePost extends Post
{
    private String message;
    // Constructors and methods omitted.
}
```

```
public class PhotoPost extends Post
{
    private String filename;
    private String caption;
    // Constructors and methods omitted.
}
```

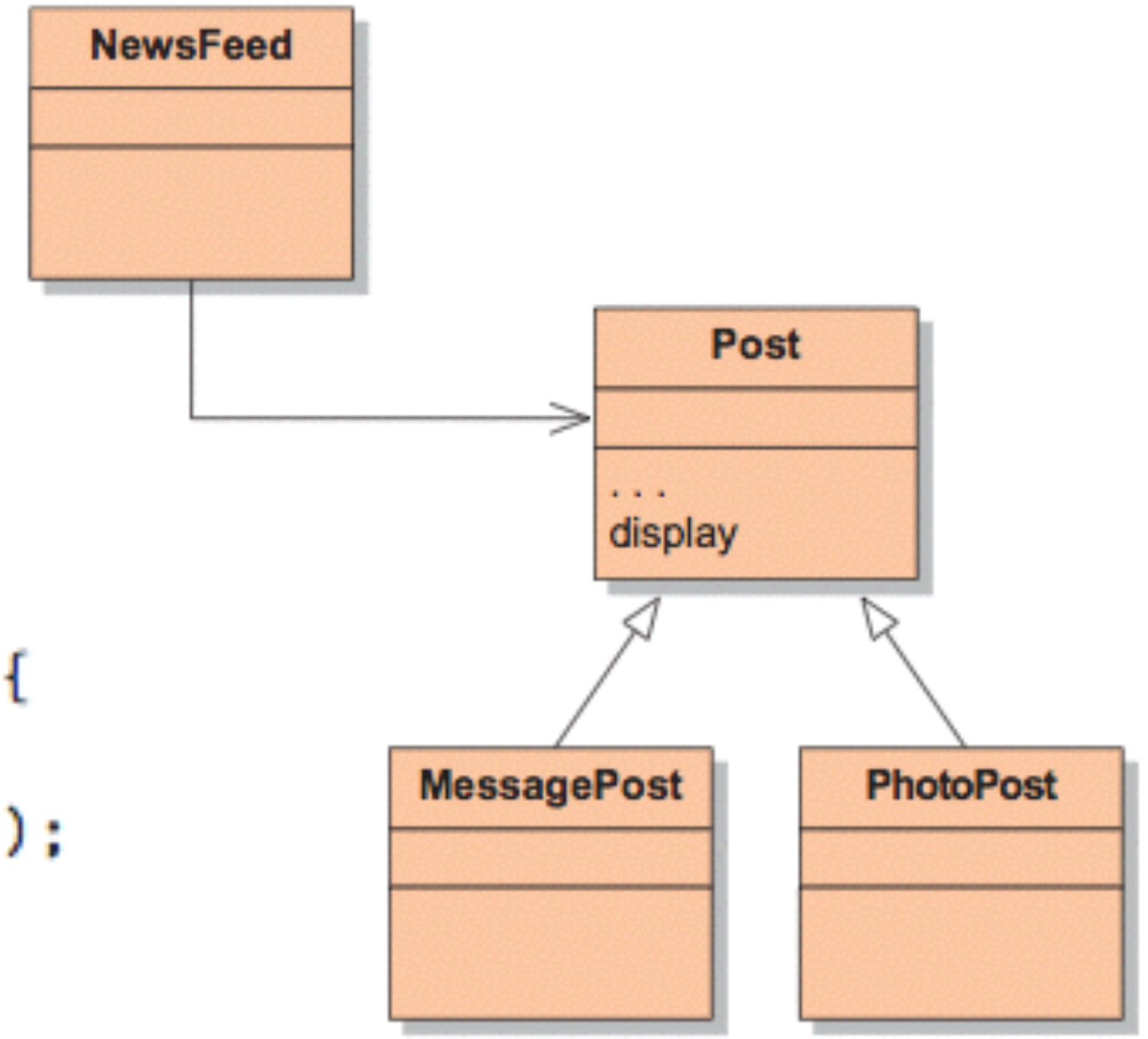
Adding other post types



Adding other post types



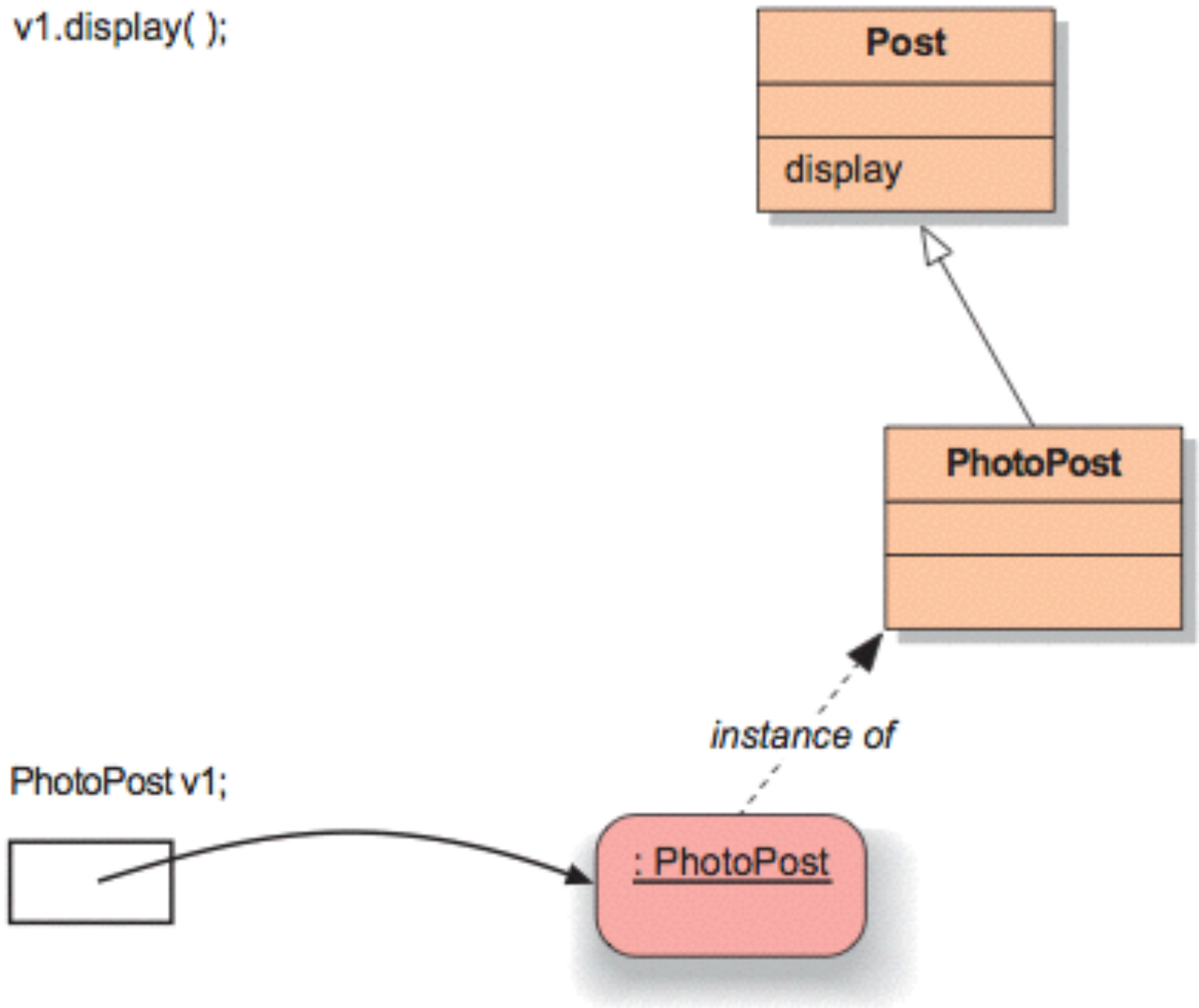
Demo: version 2 (con herencia)



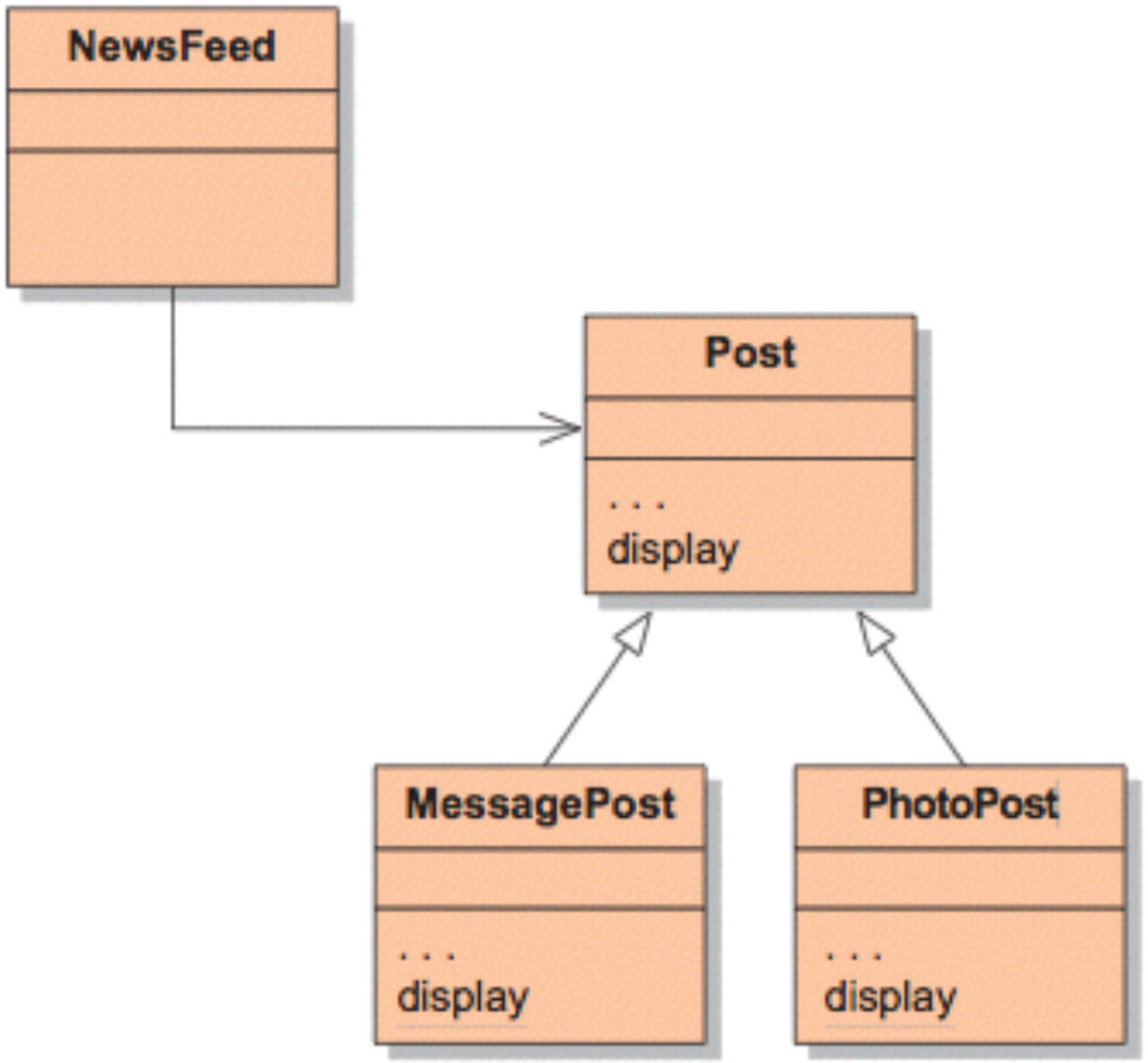

```
for(Post post : posts) {
    post.display();
    System.out.println();
}
```

v1.display();

v1.display( );

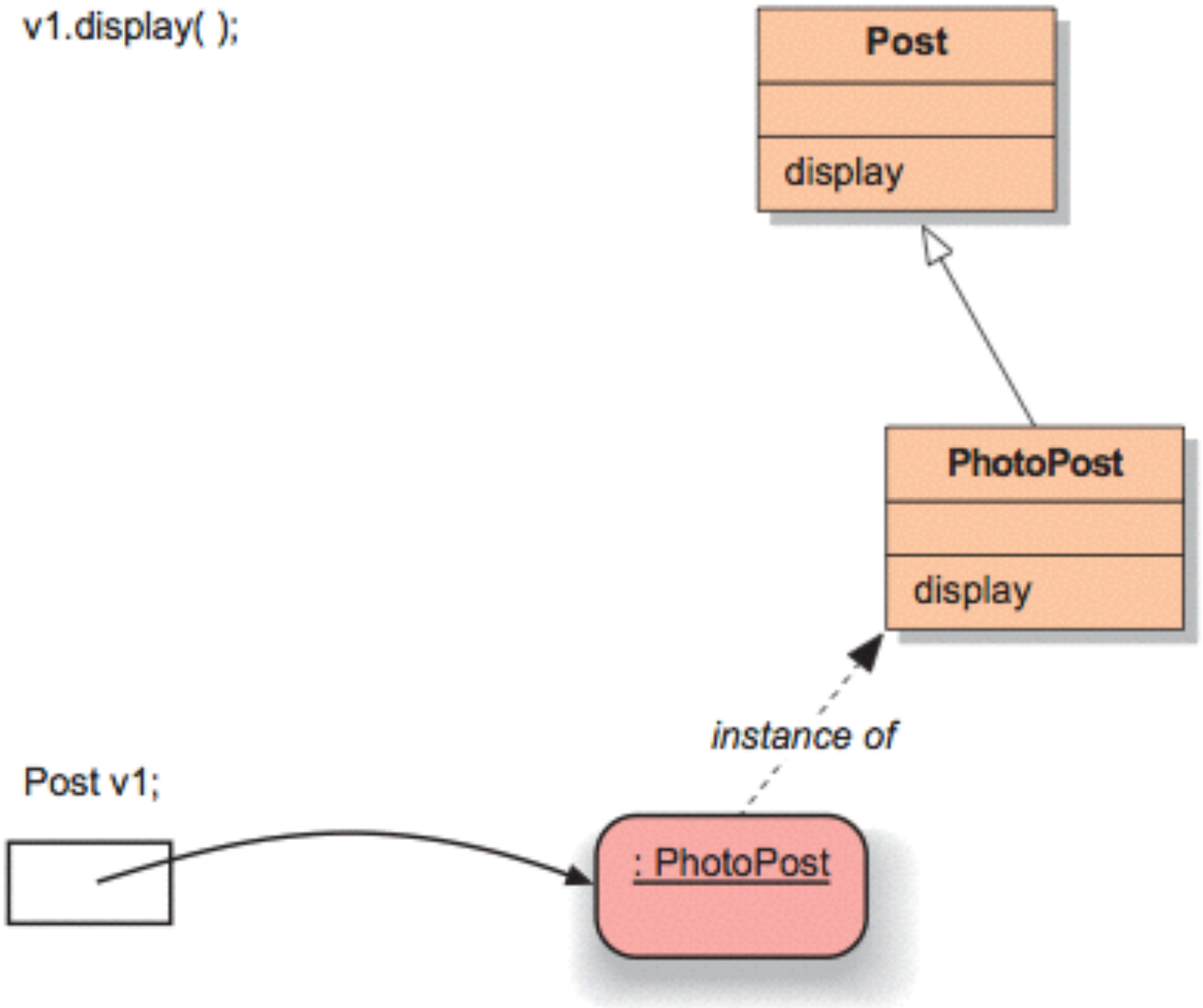


Demo: version 3 (overriding)



v1.display();

v1.display( );



Method Definition (Point printString method)

- modifier
- return type
- method name
- argument types

```
public String printString(){
    return x + ","+y;
}
```

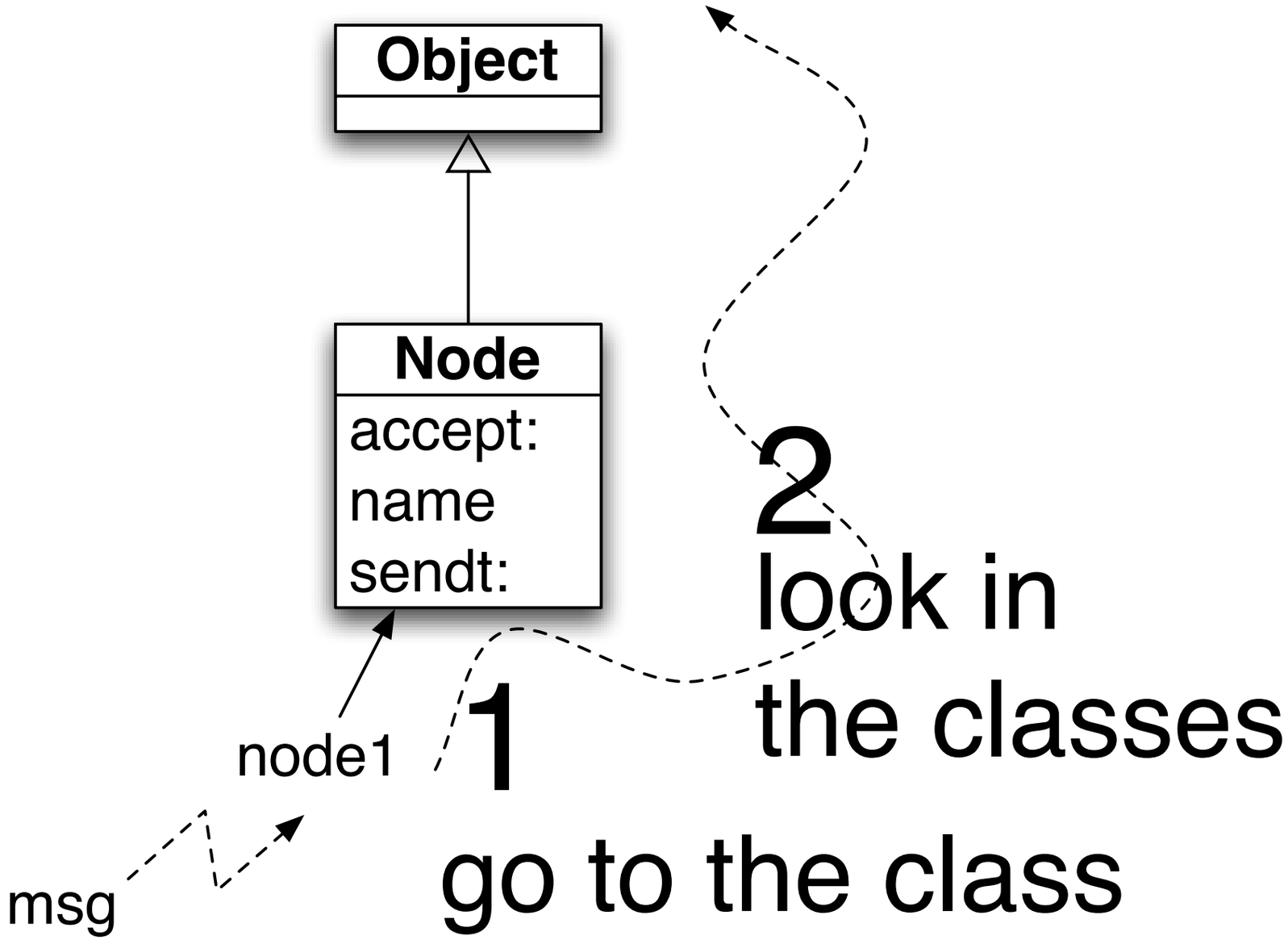
Message Sending - coloredPoint.printString()

- Looking up the method that should be executed and executing it.
- When a message is sent, the method corresponding to the message is looked up through the inheritance chain

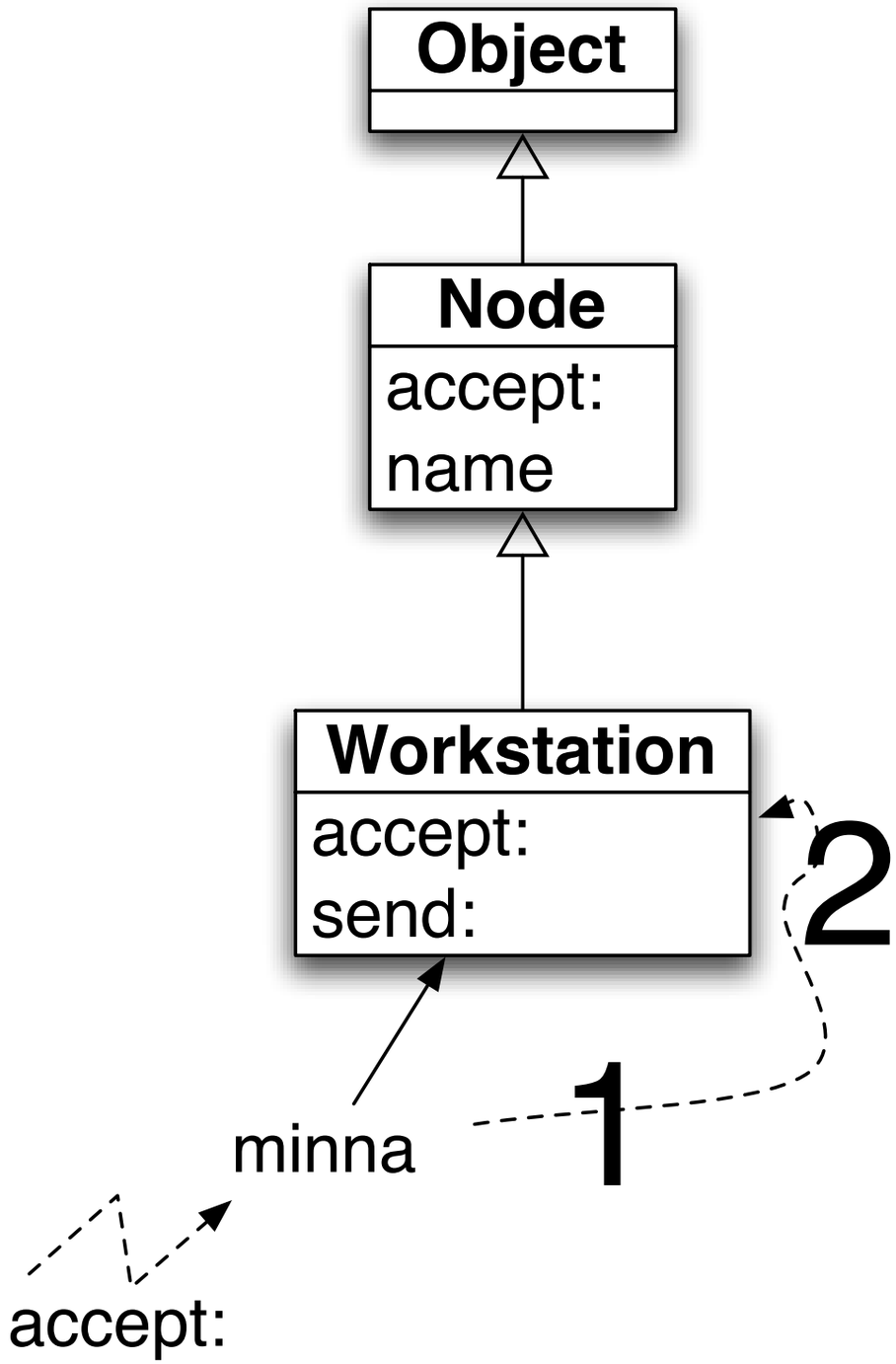
Method Lookup

- The lookup starts in the class of the receiver.
- If the method is defined in that class, it is returned.
- Otherwise the search continues in the superclasses of the receiver's class. If no method is found and there is no superclass to explore (class Object), this is an ERROR.

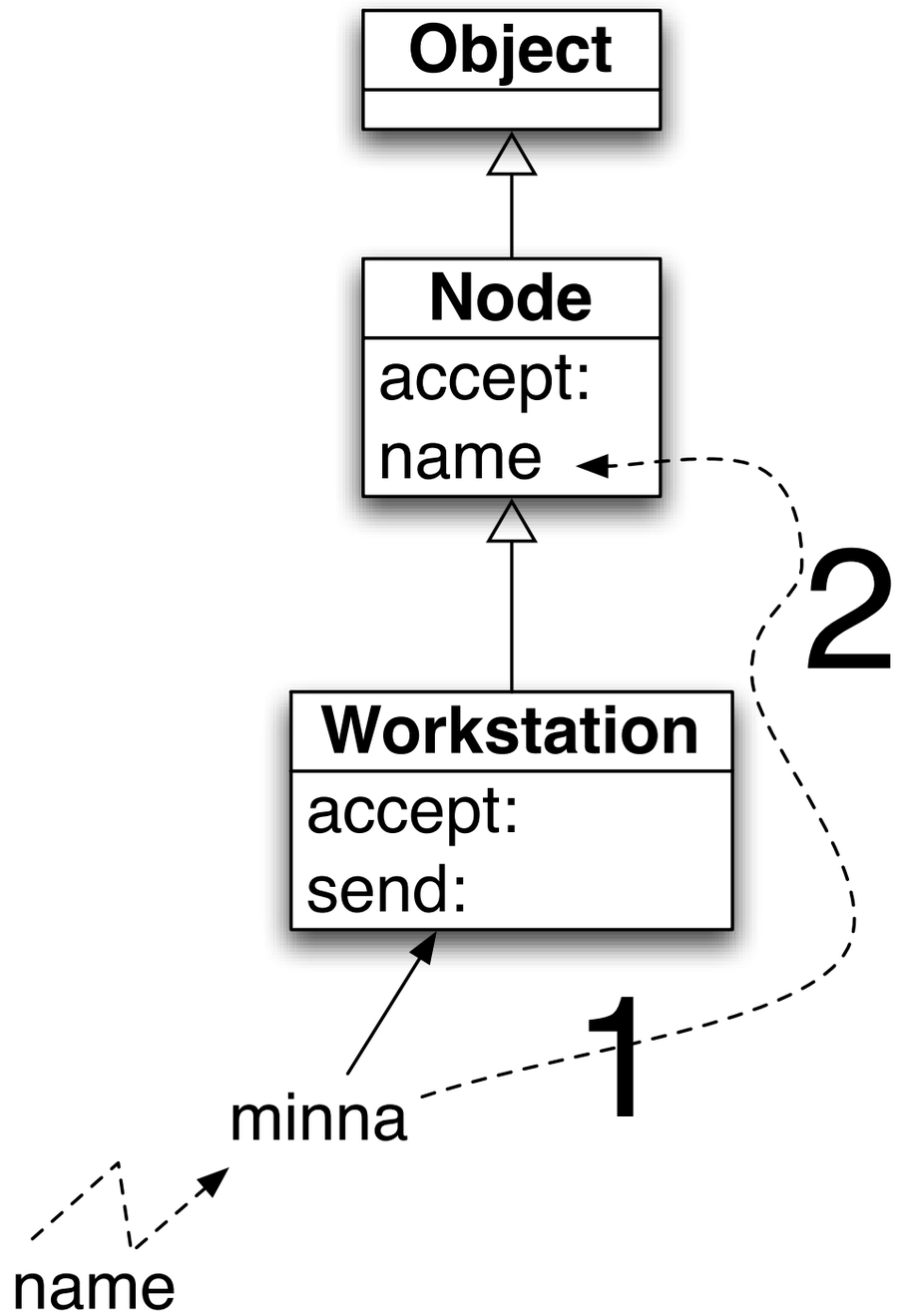
Lookup: class + inheritance



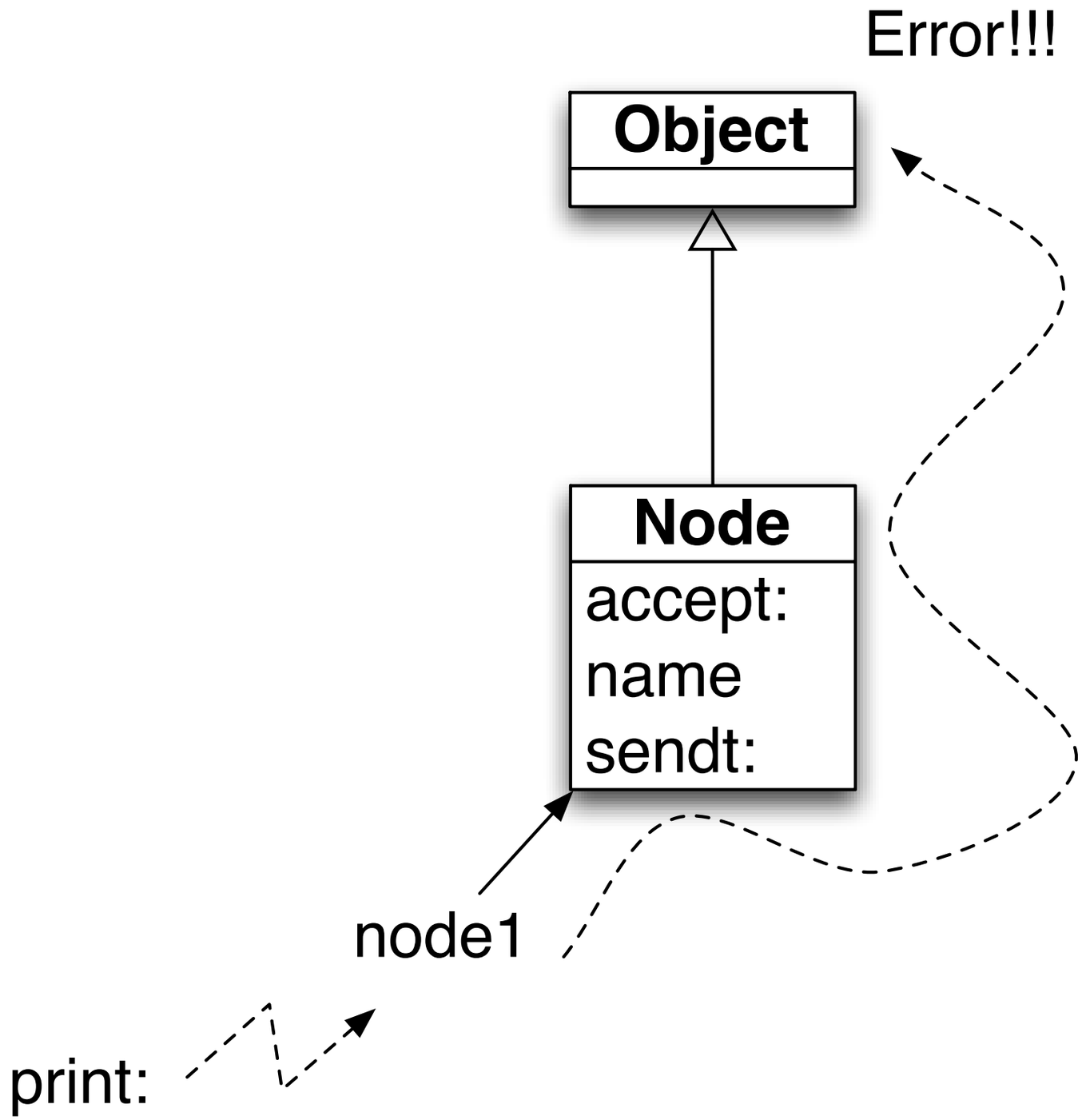
Lookup: class + inheritance



Lookup: class + inheritance



Lookup: class + inheritance



Overriding method printString

Point

```
public String printString(){
    return x + ","+y;
}
```

ColoredPoint

```
public String printString(){
    return x + ","+y+","+color;
}
```

How invoke to overridden methods?

Point

```
public String printString(){
    return x + ","+y;
}
```

ColoredPoint

```
public String printString(){
    return super.printString()+","+color;
}
```

The semantic of the super

- Like this, super is a pseudo-variable that refers to the receiver of the message

- It is used to invoke overridden methods.

- When using this, the lookup of the method begins in the class of the receiver.

- When using super; the lookup of the method begins in **the superclass of the class of the method containing** the super expression.

Exercises

Exercise 9.11 Assume that you see the following lines of code:

```
Device dev = new Printer();
dev.getName();
```

`Printer` is a subclass of `Device`. Which of these classes must have a definition of method `getName` for this code to compile?

Exercise 9.12 In the same situation as in the previous exercise, if both classes have an implementation of `getName`, which one will be executed?

Exercise 9.13 Assume that you write a class `Student` that does not have a declared superclass. You do not write a `toString` method. Consider the following lines of code:

```
Student st = new Student();
String s = st.toString();
```

Will these lines compile? If so, what exactly will happen when you try to execute?

We are ready