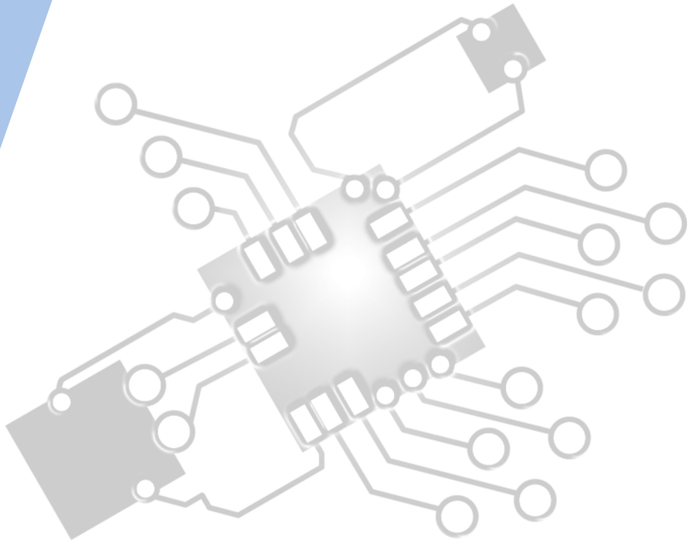
Features of OOP

IB Computer Science



Content developed by
Dartford Grammar School
Computer Science Department

HL Topics 1-7, D1-4

1: System design

2: Computer Organisation

3: Networks

4: Computational thinking

5: Abstract data structures

6: Resource management

7: Control

D: OOP

HL & SL D.2 Overview

D.2 Features of OOP

D.2.1 Define the term encapsulation

D.2.2 Define the term inheritance

D.2.3 Define the term polymorphism

D.2.4 Explain the advantages of encapsulation

D.2.5 Explain the advantages of inheritance

D.2.6 Explain the advantages of polymorphism

D.2.7 Describe the advantages of libraries of objects

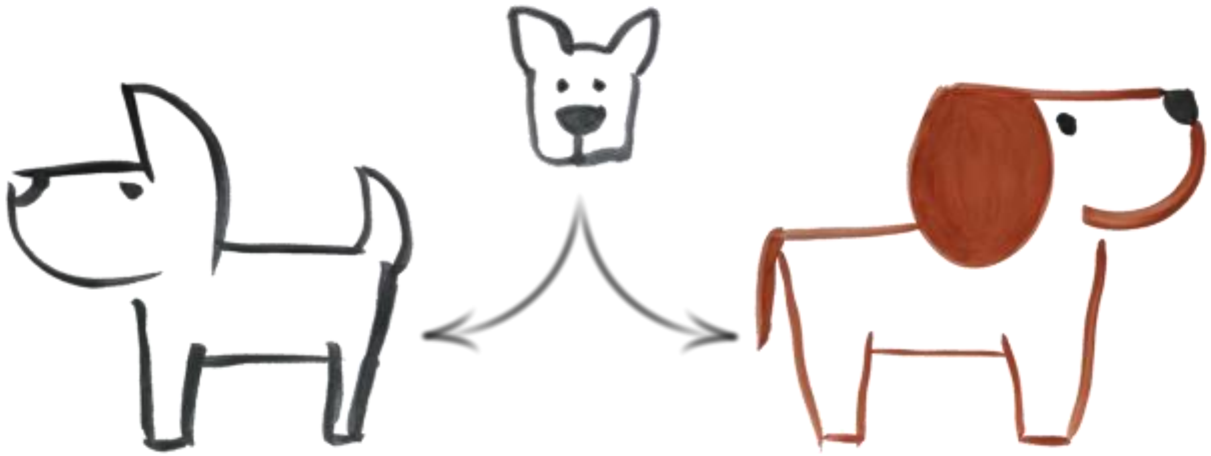
D.2.8 Describe the disadvantages of OOP

D.2.9 Discuss the use of programming teams

D.2.10 Explain the advantages of modularity in program development

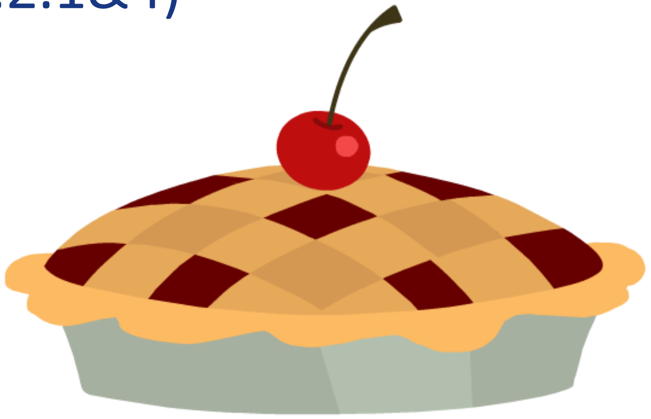
Topic D.2.6

Explain the **advantages** of **polymorphism**

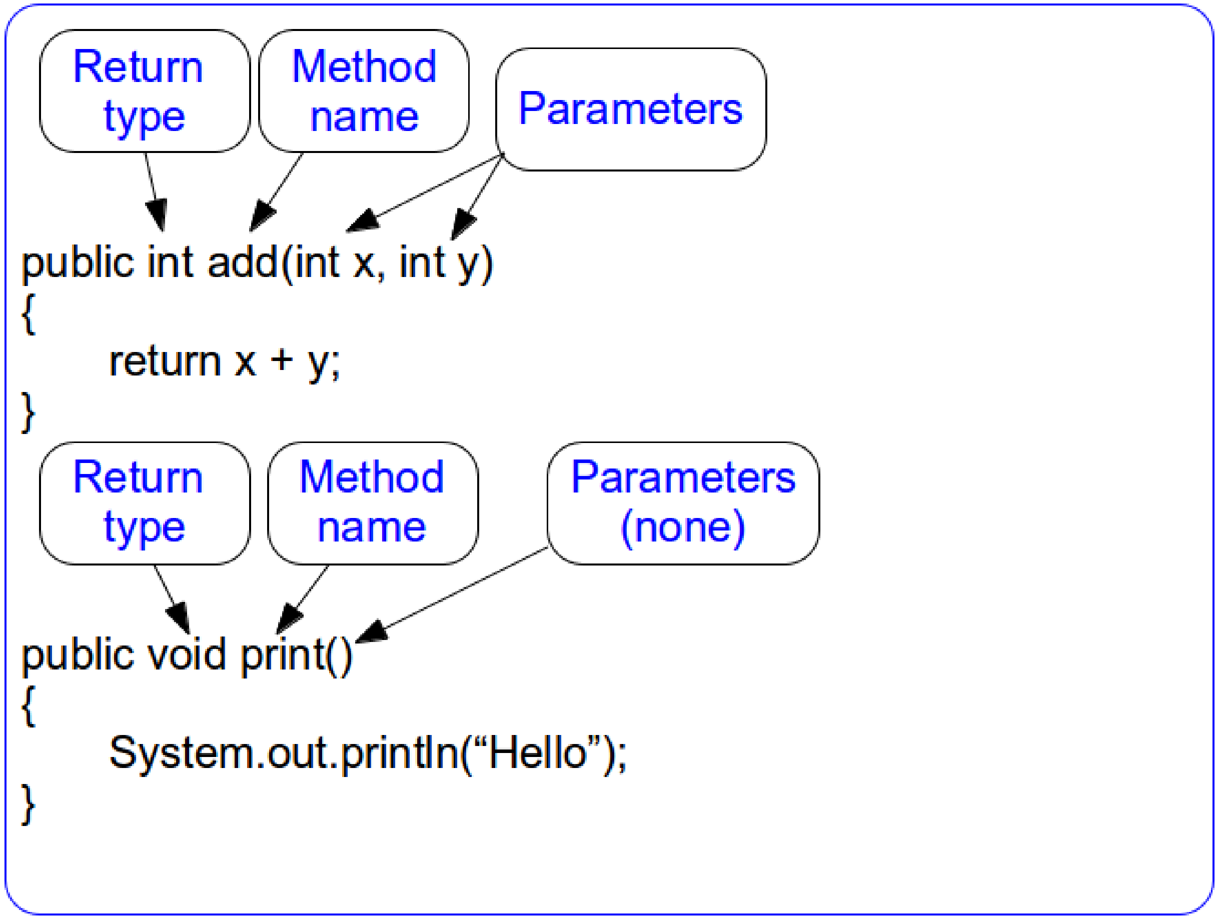


Four OOP fundamentals:

- **A**bstraction (*See* Topic 4.1.17-20)
- **P**olymorphism (*See* Topic D.2.3&6)
- **I**nheritance (*See* Topic D.2.2&5)
- **E**ncapsulation (*See* Topic D.2.1&4)



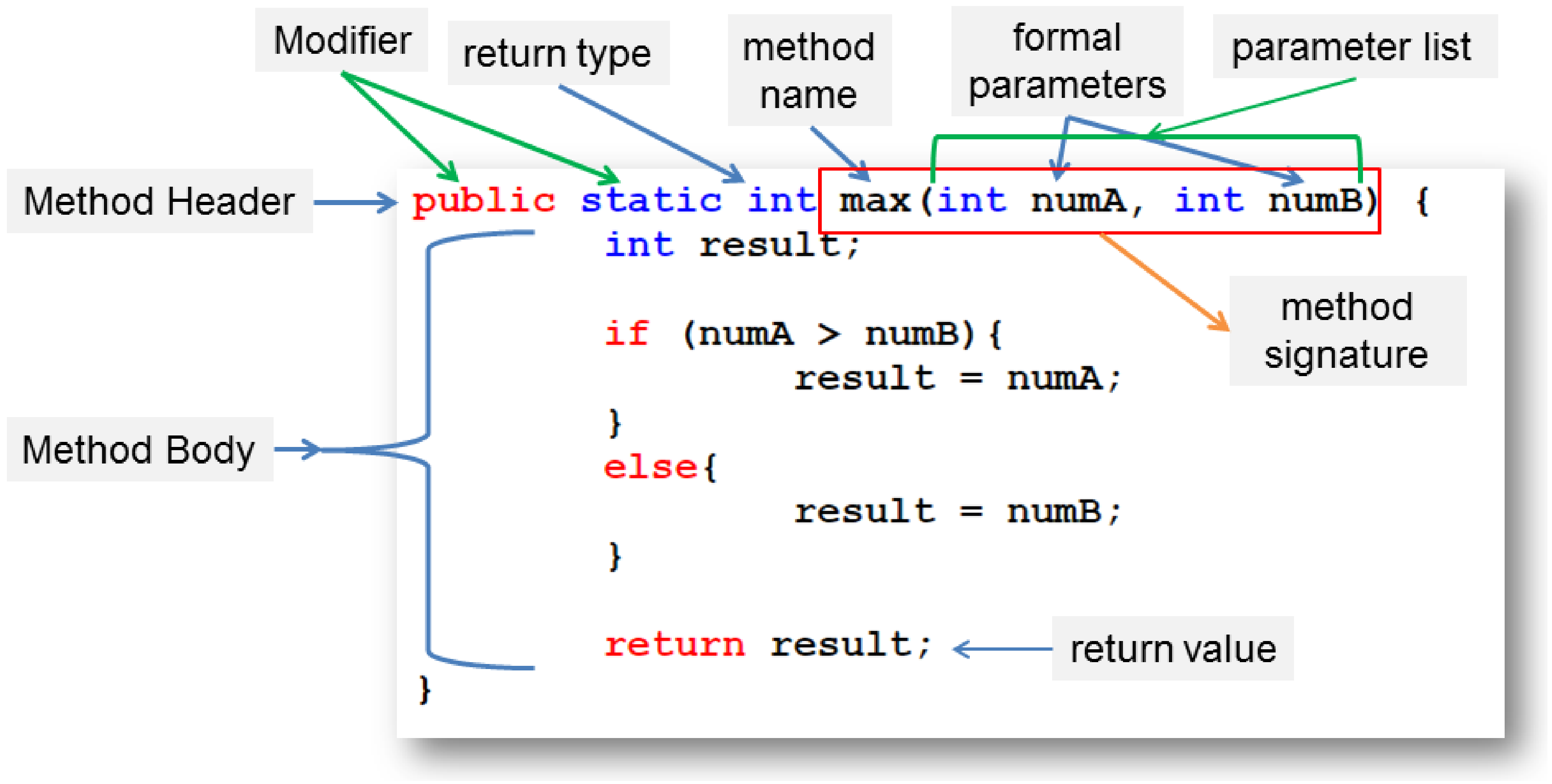
Reminder: Method signatures



```
public int add(int x, int y)
{
      return x + y;
}
```

```
public void print()
{
      System.out.println("Hello");
}
```

Reminder: Parts of a method



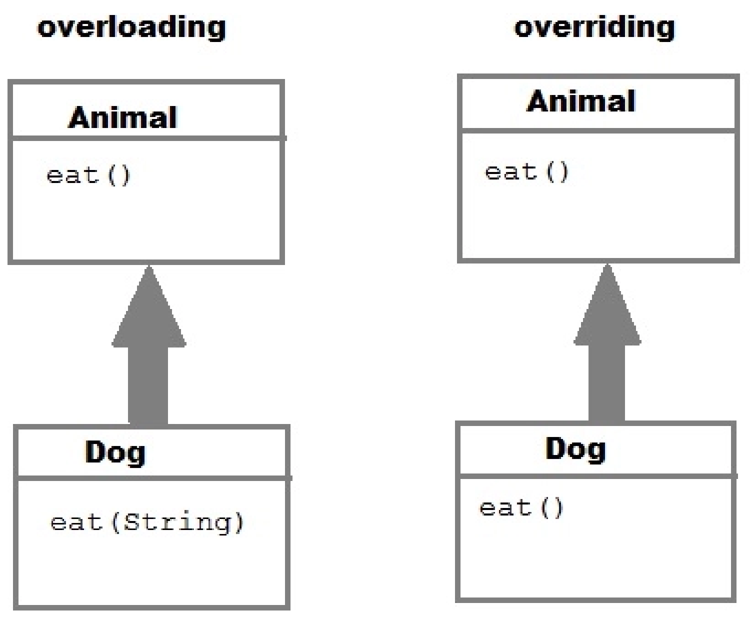
```
public static int max(int numA, int numB)  {
          int result;

          if (numA > numB){
                    result = numA;
          }
          else{
                    result = numB;
          }

          return result;
}
```

Definition: Polymorphism

- **Polymorphism** is derived from 2 Greek words: *poly* and *morph*. The word "poly" means **many** and "morphs" means **forms**.
- So **polymorphism** means **many forms**; specifically in Java it means that two methods can have the same name but different contents/functions.
- **In short**: **methods (behaviours) have the same name but different parameter lists and processes**

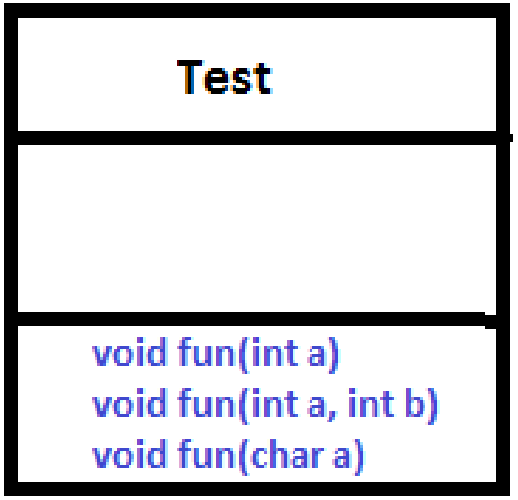
Types of Polymorphism

A. Overloading (*same class*)
B. Overriding (*different classes/inheritance*)



A. Overloading

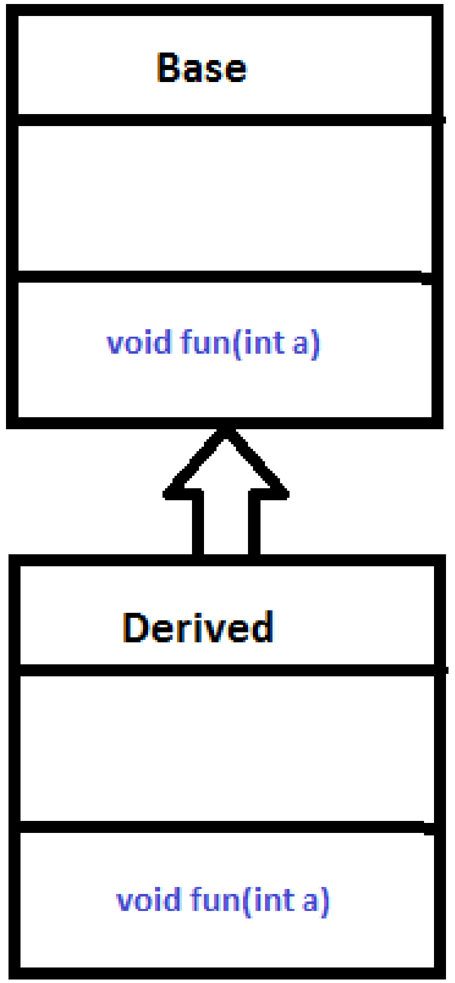
- Overloading allows different methods to have **same name**, but **different signatures** where signature can differ by number of input parameters or type of input parameters or both.



Overloading

B. Overriding

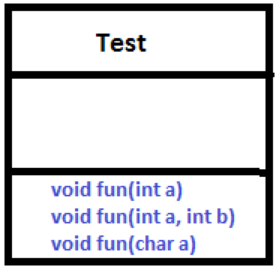
- Overriding allows a sub class to provide a **specific implementation** of a method that is already provided by one of its super classes.
- When a method in a subclass has the **same name**, **same parameters or signature** and **same return type** as a method in its super-class, then the method in the subclass is said to **override** the method in the super-class.



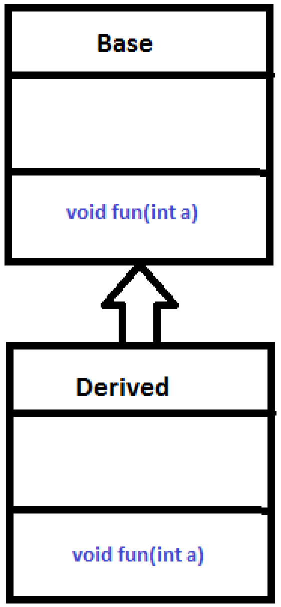
Overriding

Overloading vs Overriding

- Overloading is about **same function** have **different signatures** (usually in the **same class**).
- Overriding is about **same function**, **same signature** but **different classes** connected through **inheritance**.

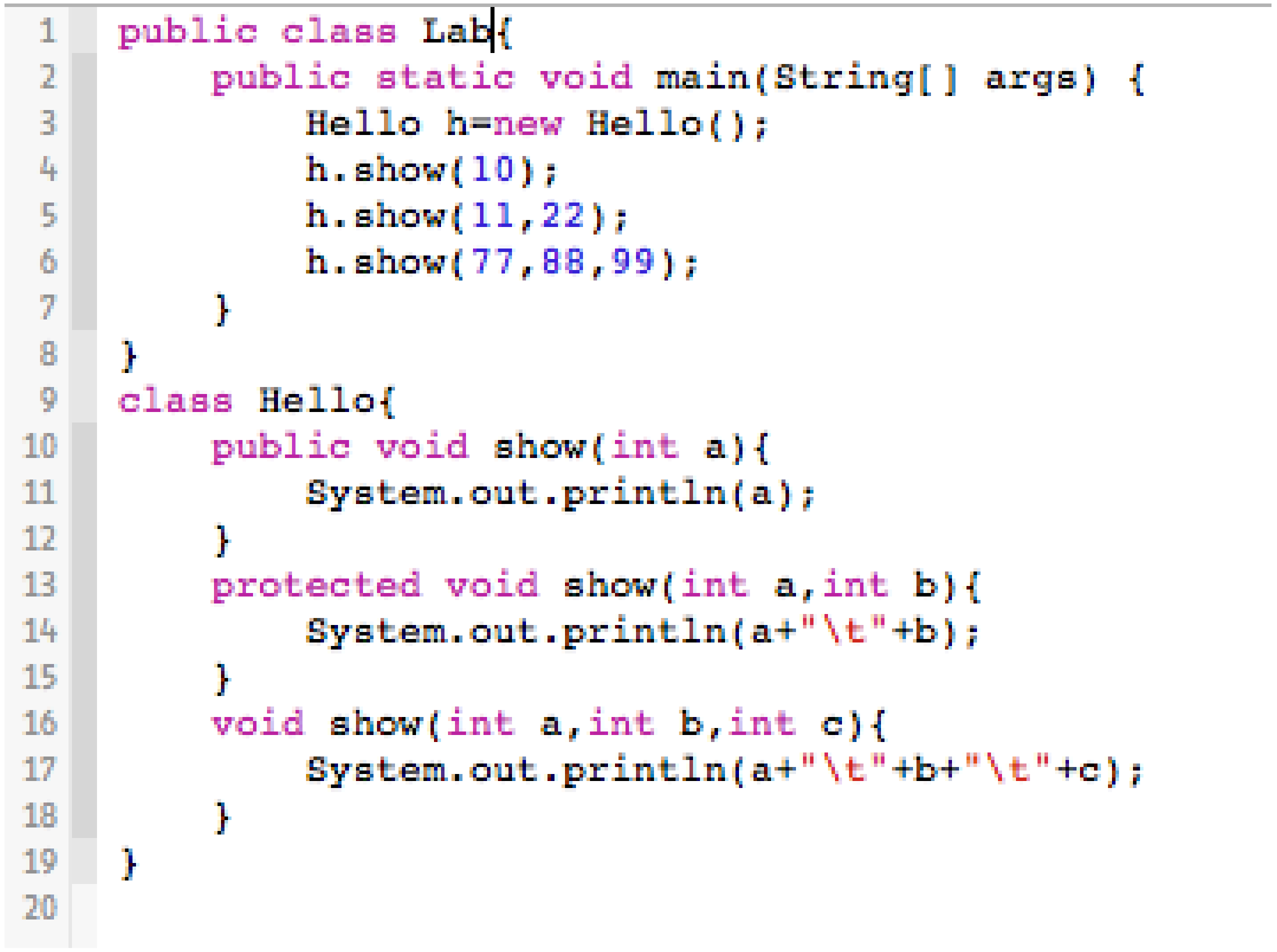


Overloading



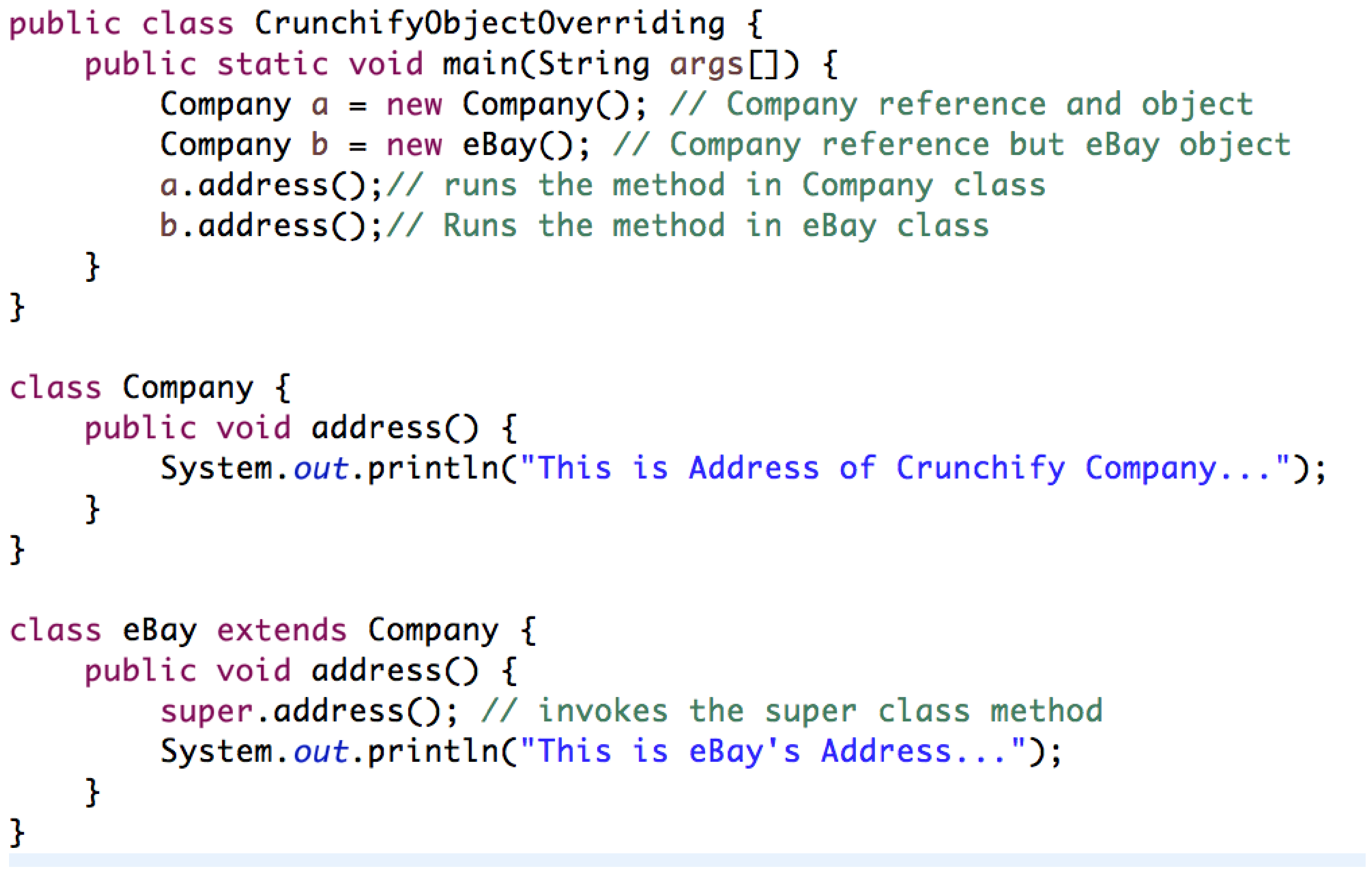
Overriding

Overloading Example: Java



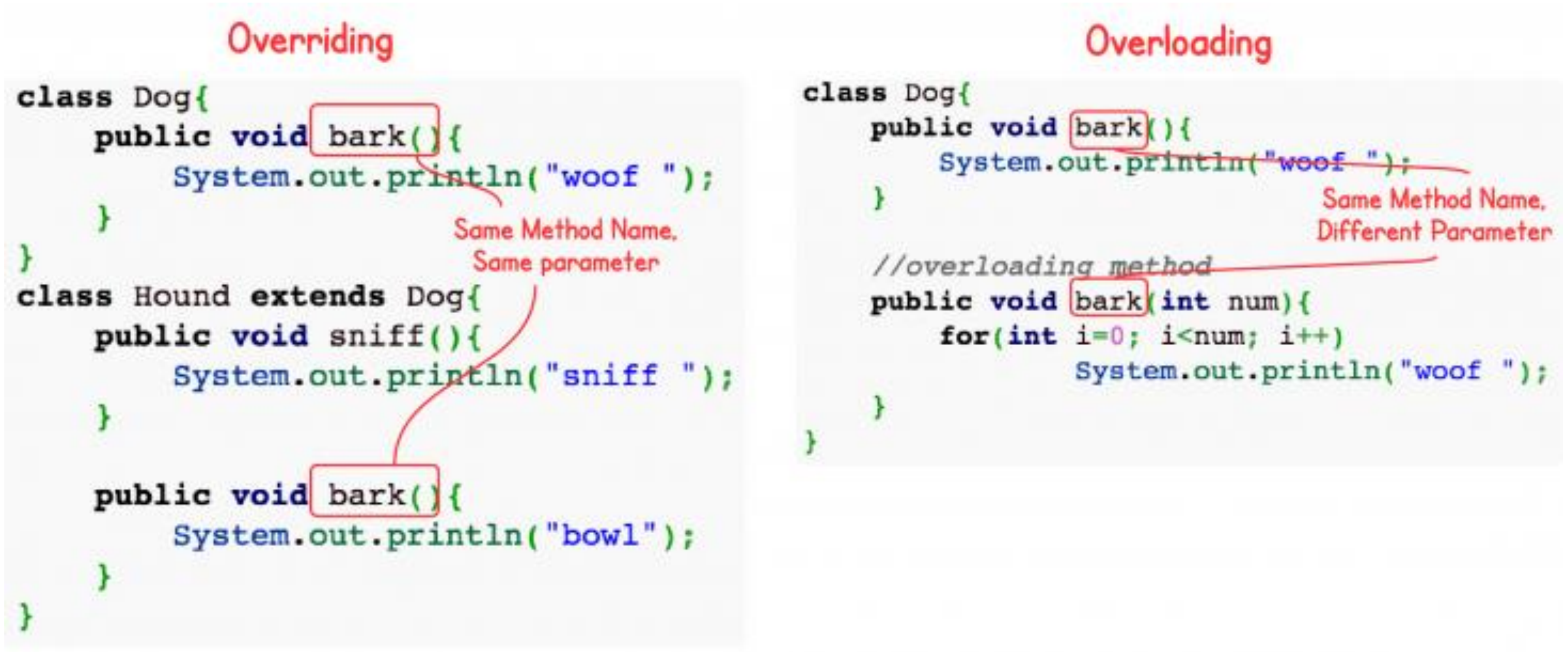
```
public class Lab{
	public static void main(String[] args) {
		Hello h=new Hello();
		h.show(10);
		h.show(11,22);
		h.show(77,88,99);
	}
}
class Hello{
	public void show(int a){
		System.out.println(a);
	}
	protected void show(int a,int b){
		System.out.println(a+"\t"+b);
	}
	void show(int a,int b,int c){
		System.out.println(a+"\t"+b+"\t"+c);
	}
}
```

Overriding Example: Java



```java
public class CrunchifyObjectOverriding {
    public static void main(String args[]) {
        Company a = new Company(); // Company reference and object
        Company b = new eBay(); // Company reference but eBay object
        a.address();// runs the method in Company class
        b.address();// Runs the method in eBay class
    }
}

class Company {
    public void address() {
        System.out.println("This is Address of Crunchify Company...");
    }
}

class eBay extends Company {
    public void address() {
        super.address(); // invokes the super class method
        System.out.println("This is eBay's Address...");
    }
}
```

Polymorphism Comparison: Java



Overriding

```
class Dog{
    public void bark(){
        System.out.println("woof ");
    }
}
class Hound extends Dog{
    public void sniff(){
        System.out.println("sniff ");
    }

    public void bark(){
        System.out.println("bowl");
    }
}
```

Same Method Name,
Same parameter

Overloading

```
class Dog{
    public void bark(){
        System.out.println("woof ");
    }

    //overloading method
    public void bark(int num){
        for(int i=0; i<num; i++)
            System.out.println("woof ");
    }
}
```

Same Method Name,
Different Parameter