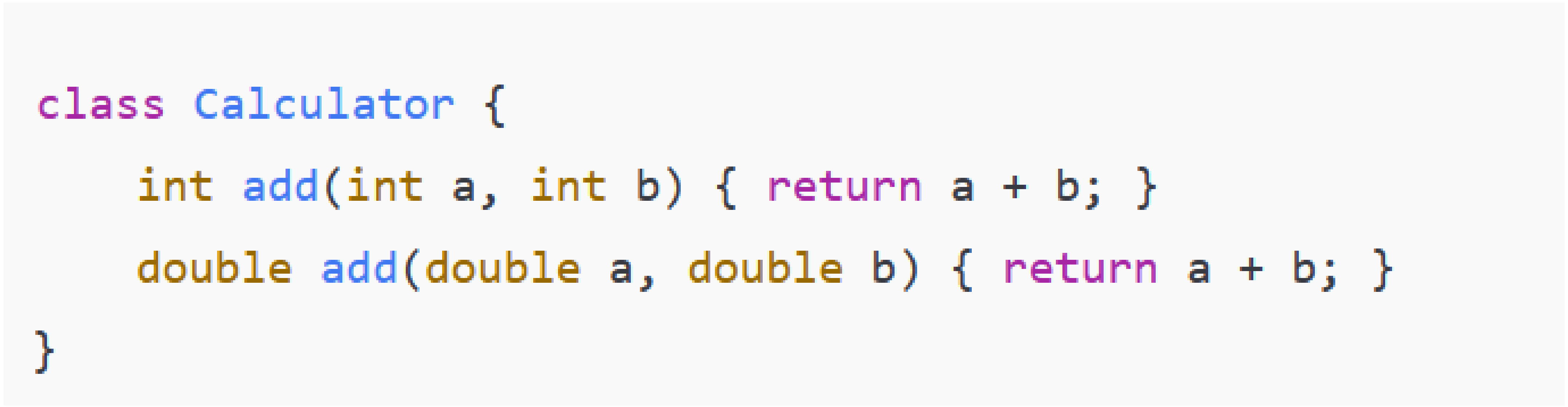
INTRODUCTION TO PROGRAMMING LANGUAGE II(JAVA)

Fariha Zahin
Lecturer
CSE, Southeast University

Method Overloading

Definition:

Same method name, but different parameters (different type or number) in the same class.

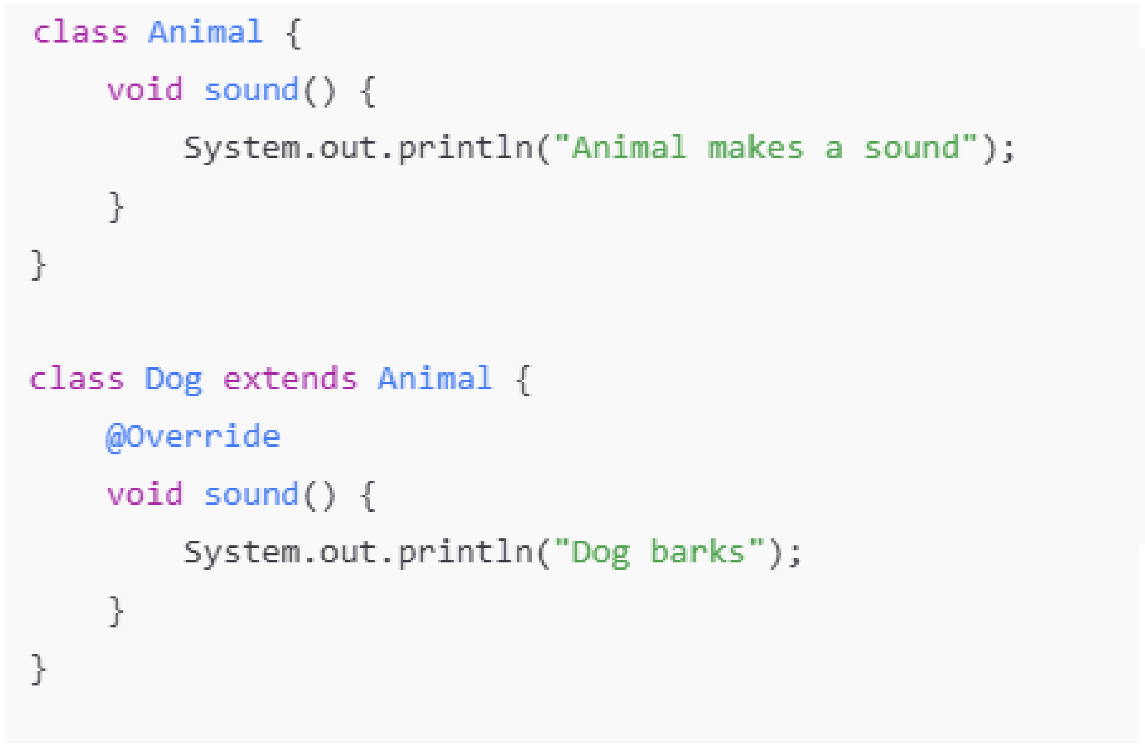


```
class Calculator {
    int add(int a, int b) { return a + b; }
    double add(double a, double b) { return a + b; }
}
```

Method Overriding

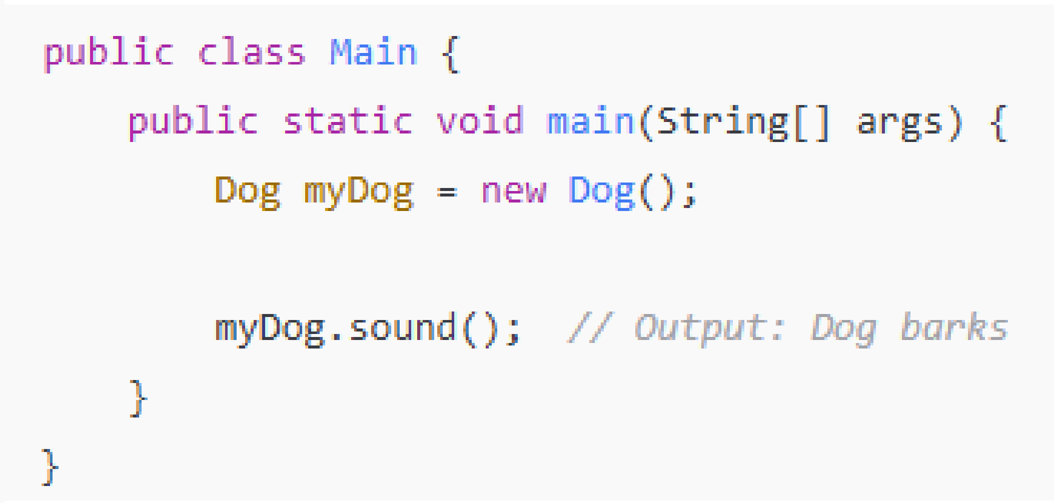
Definition:

Same method name and same parameters between parent and child classes; child class redefines behavior.



```java
class Animal {
    void sound() {
        System.out.println("Animal makes a sound");
    }
}

class Dog extends Animal {
    @Override
    void sound() {
        System.out.println("Dog barks");
    }
}
```



```java
public class Main {
    public static void main(String[] args) {
        Dog myDog = new Dog();

        myDog.sound();  // Output: Dog barks
    }
}
```

Polymorphism
Definition:
The ability of an object to take many forms.

**One interface, multiple implementations.

Types of Polymorphism:

- **Compile-time Polymorphism:**
  Achieved by **Method Overloading**.
- **Run-time Polymorphism:**
  Achieved by **Method Overriding**.

1. **Compile-Time Polymorphism (Static Polymorphism)**
   The method to be executed is determined at compile time.

How it's achieved:
Through **Method Overloading** (same method name, different parameter lists).
Characteristics:

- Faster execution since the method call is resolved during compilation.
- Happens within the same class.

2. Run-Time Polymorphism (Dynamic Polymorphism)
The method to be executed is determined during runtime.

How it's achieved:
Through **Method Overriding** (subclass provides specific implementation of a superclass method).
Characteristics:

- Slower than compile-time polymorphism (decision made at runtime).
- Requires inheritance (parent and child classes).