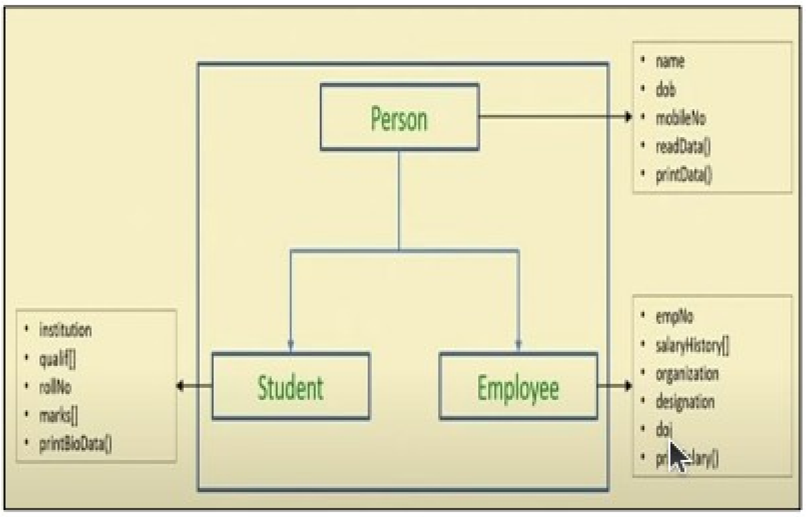
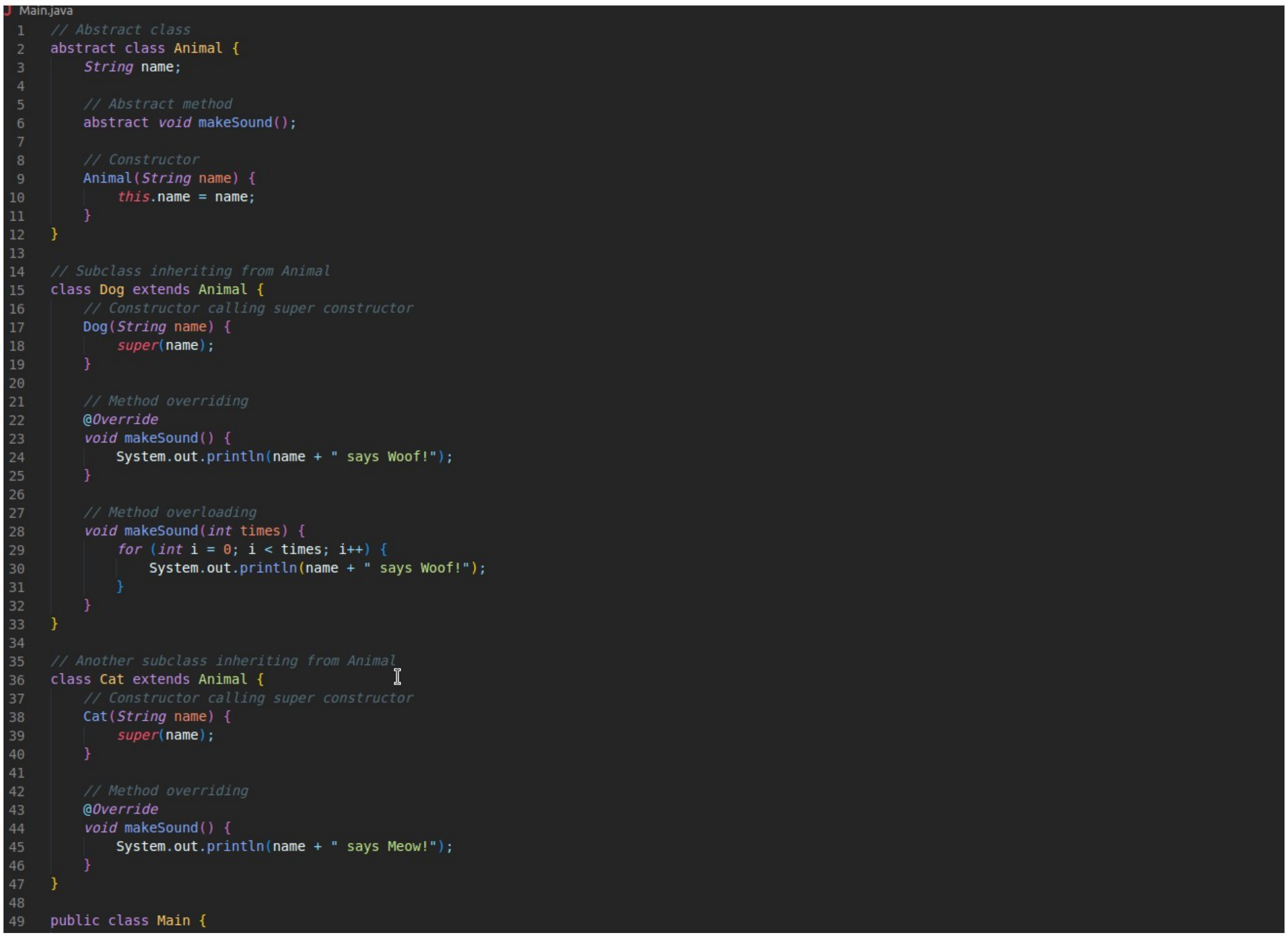
Nithin S
221IT085

IT150 Lab Assignment 10

Q1.. Write a Java program demonstrating the concept of simple inheritance as shown below

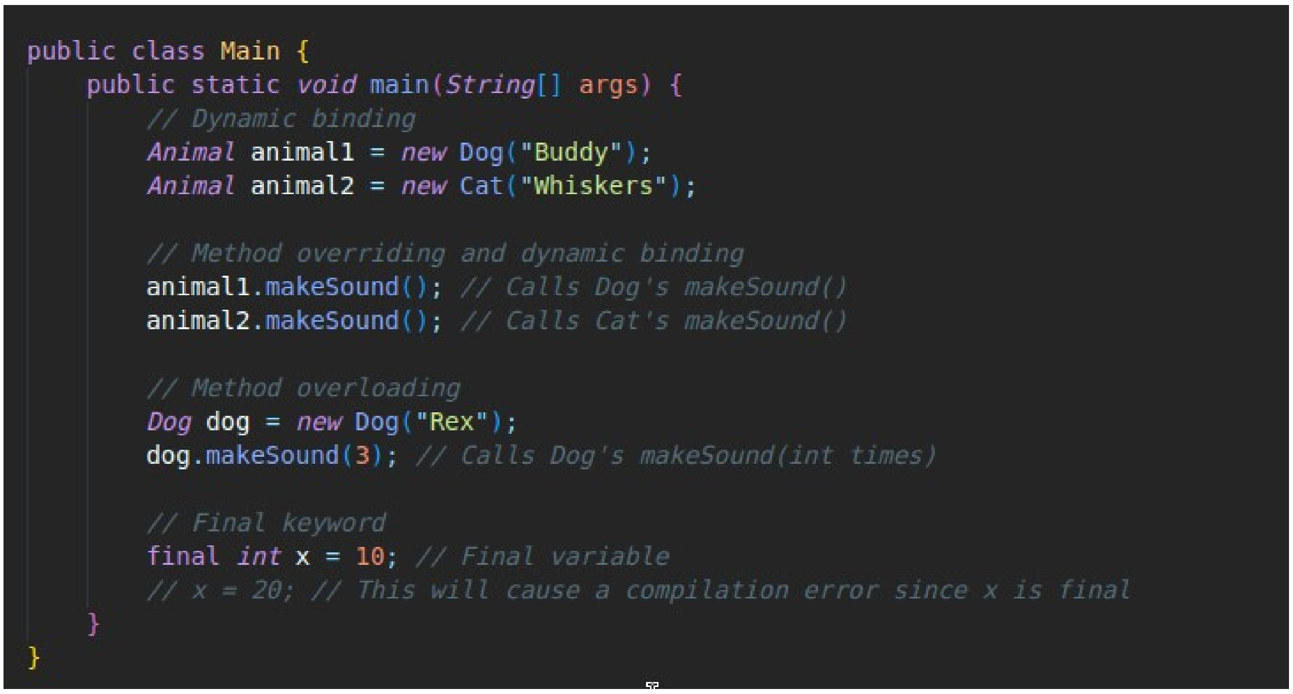


Code



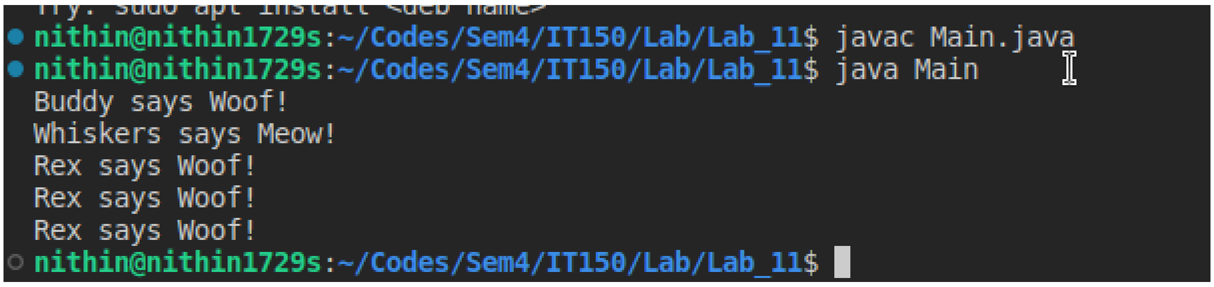
Main.java

```java
// Abstract class
abstract class Animal {
    String name;

    // Abstract method
    abstract void makeSound();

    // Constructor
    Animal(String name) {
        this.name = name;
    }
}

// Subclass inheriting from Animal
class Dog extends Animal {
    // Constructor calling super constructor
    Dog(String name) {
        super(name);
    }

    // Method overriding
    @Override
    void makeSound() {
        System.out.println(name + " says Woof!");
    }

    // Method overloading
    void makeSound(int times) {
        for (int i = 0; i < times; i++) {
            System.out.println(name + " says Woof!");
        }
    }
}

// Another subclass inheriting from Animal
class Cat extends Animal {
    // Constructor calling super constructor
    Cat(String name) {
        super(name);
    }

    // Method overriding
    @Override
    void makeSound() {
        System.out.println(name + " says Meow!");
    }
}

public class Main {
```



```java
public class Main {
    public static void main(String[] args) {
        // Dynamic binding
        Animal animal1 = new Dog("Buddy");
        Animal animal2 = new Cat("Whiskers");

        // Method overriding and dynamic binding
        animal1.makeSound(); // Calls Dog's makeSound()
        animal2.makeSound(); // Calls Cat's makeSound()

        // Method overloading
        Dog dog = new Dog("Rex");
        dog.makeSound(3); // Calls Dog's makeSound(int times)

        // Final keyword
        final int x = 10; // Final variable
        // x = 20; // This will cause a compilation error since x is final
    }
}
```

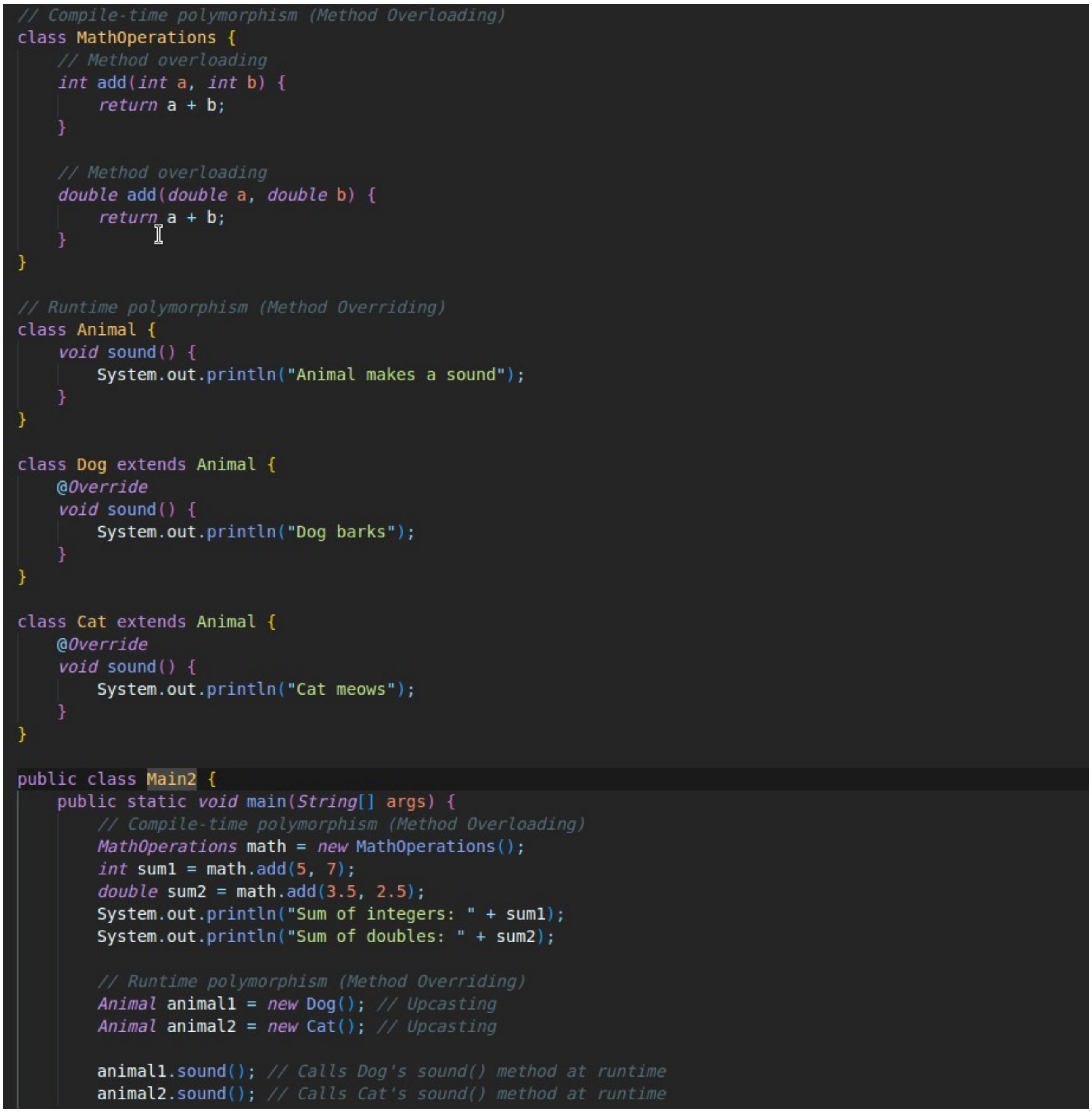
OUTPUT



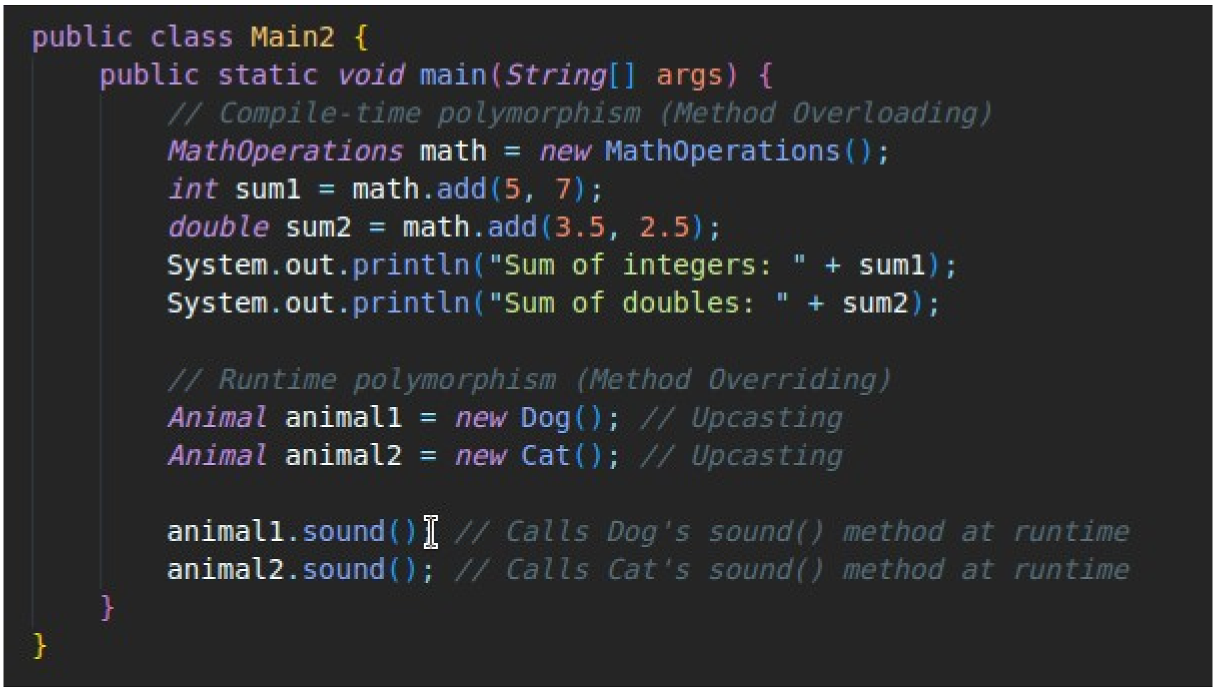
```
Try: sudo apt install <deb name>
nithin@nithin1729s:~/Codes/Sem4/IT150/Lab/Lab_11$ javac Main.java
nithin@nithin1729s:~/Codes/Sem4/IT150/Lab/Lab_11$ java Main
Buddy says Woof!
Whiskers says Meow!
Rex says Woof!
Rex says Woof!
Rex says Woof!
nithin@nithin1729s:~/Codes/Sem4/IT150/Lab/Lab_11$
```

Q2.Write a Java program (use Inheritance) demonstrating the concept of :
a. Constructors
b. Method overriding
c. Method overloading
d. usage of super keyword
e. Dynamic Binding
f. Abstract Class and abstract method
g. Final Keyword

Code

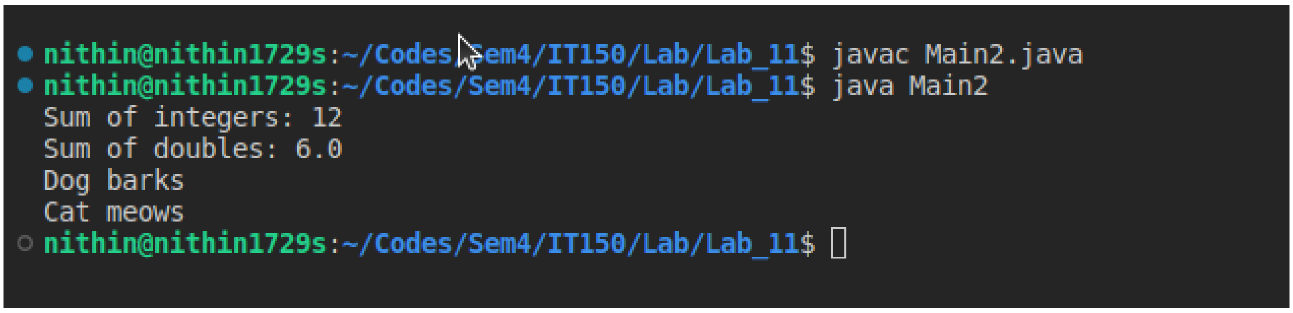


```java
// Compile-time polymorphism (Method Overloading)
class MathOperations {
    // Method overloading
    int add(int a, int b) {
        return a + b;
    }

    // Method overloading
    double add(double a, double b) {
        return a + b;
    }
}

// Runtime polymorphism (Method Overriding)
class Animal {
    void sound() {
        System.out.println("Animal makes a sound");
    }
}

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

class Cat extends Animal {
    @Override
    void sound() {
        System.out.println("Cat meows");
    }
}

public class Main2 {
    public static void main(String[] args) {
        // Compile-time polymorphism (Method Overloading)
        MathOperations math = new MathOperations();
        int sum1 = math.add(5, 7);
        double sum2 = math.add(3.5, 2.5);
        System.out.println("Sum of integers: " + sum1);
        System.out.println("Sum of doubles: " + sum2);

        // Runtime polymorphism (Method Overriding)
        Animal animal1 = new Dog(); // Upcasting
        Animal animal2 = new Cat(); // Upcasting

        animal1.sound(); // Calls Dog's sound() method at runtime
        animal2.sound(); // Calls Cat's sound() method at runtime
```



```java
public class Main2 {
    public static void main(String[] args) {
        // Compile-time polymorphism (Method Overloading)
        MathOperations math = new MathOperations();
        int sum1 = math.add(5, 7);
        double sum2 = math.add(3.5, 2.5);
        System.out.println("Sum of integers: " + sum1);
        System.out.println("Sum of doubles: " + sum2);

        // Runtime polymorphism (Method Overriding)
        Animal animal1 = new Dog(); // Upcasting
        Animal animal2 = new Cat(); // Upcasting

        animal1.sound() // Calls Dog's sound() method at runtime
        animal2.sound(); // Calls Cat's sound() method at runtime
    }
}
```

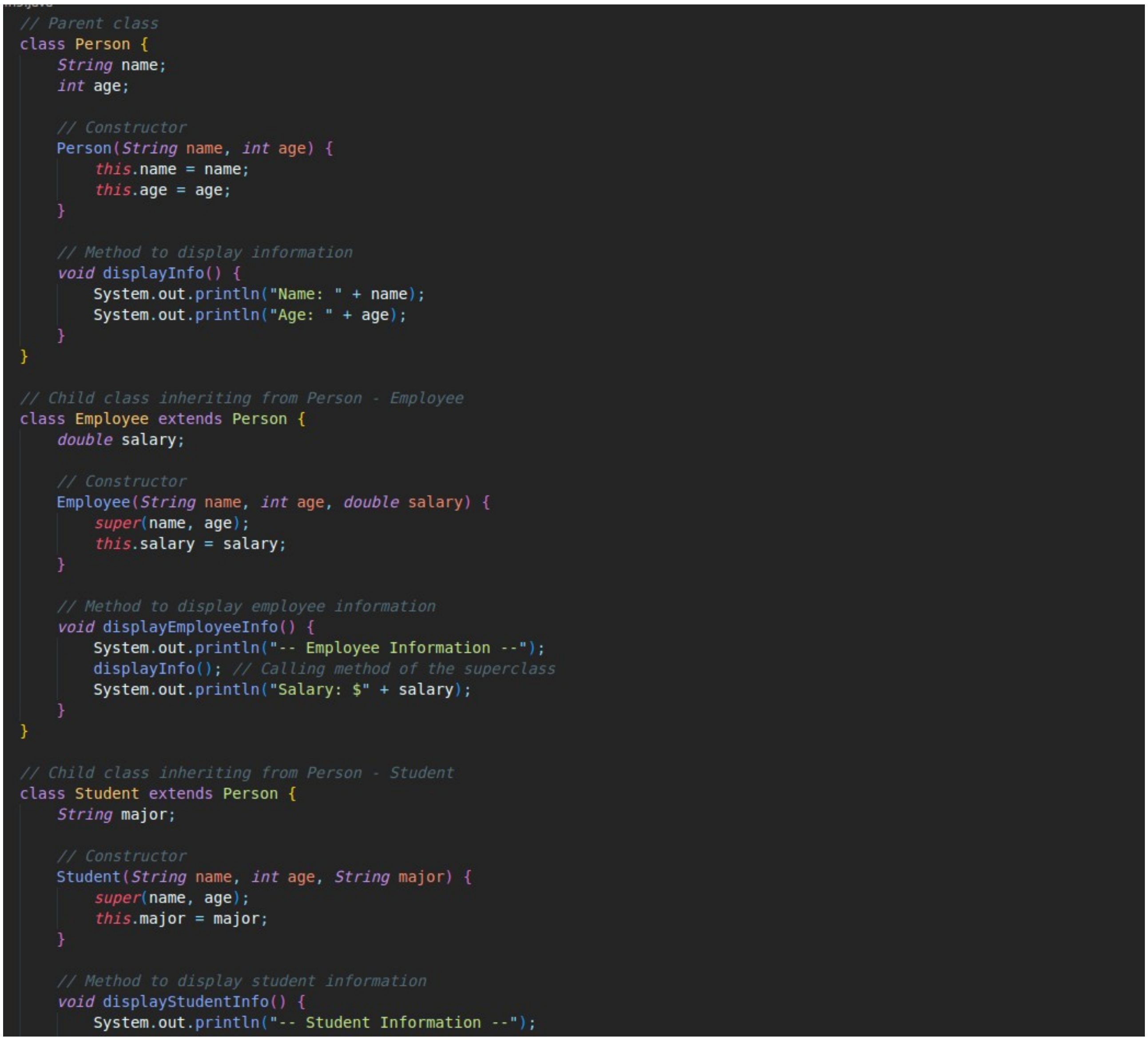
OUTPUT



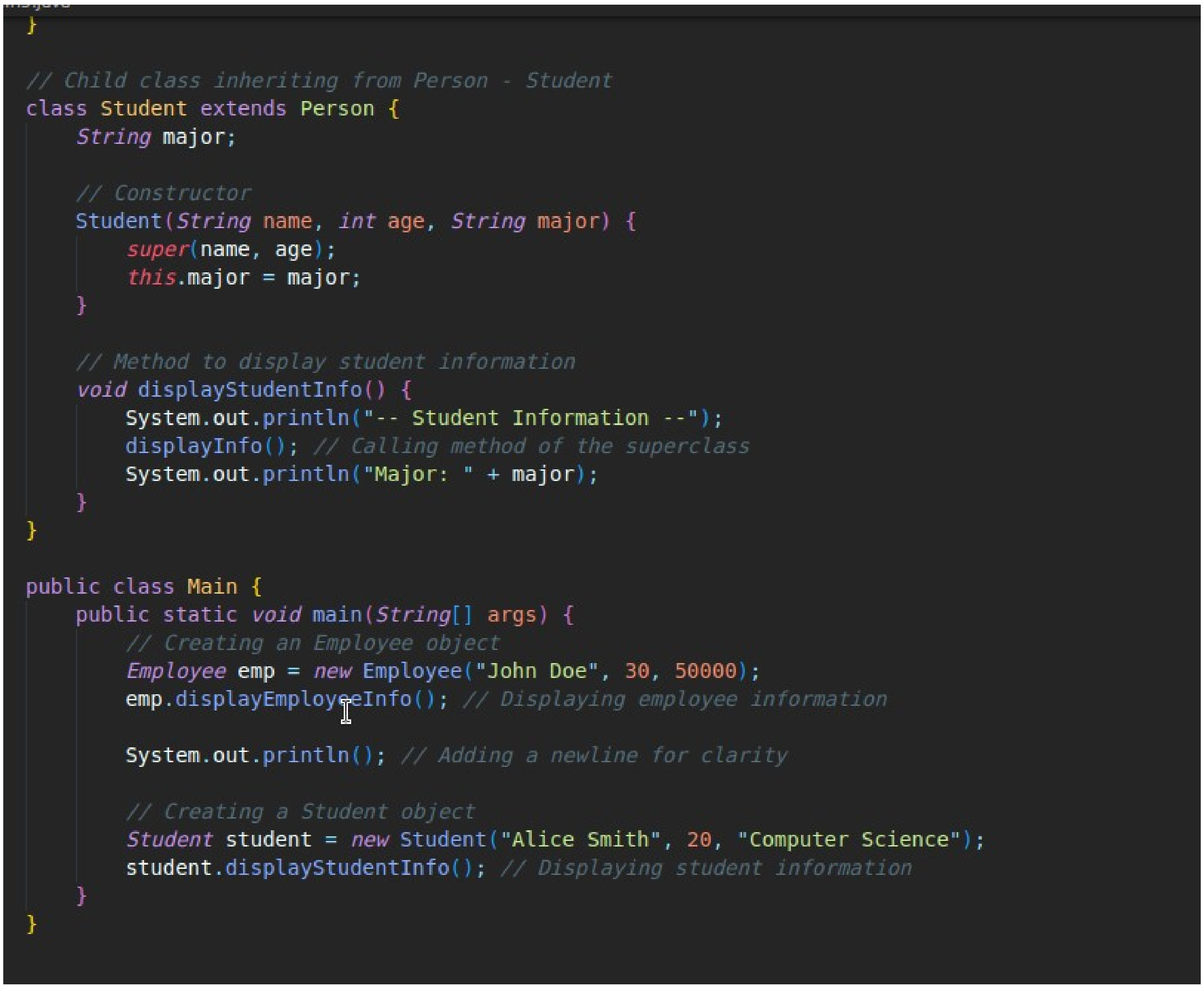
```
● nithin@nithin1729s:~/Codes/Sem4/IT150/Lab/Lab_11$ javac Main2.java
● nithin@nithin1729s:~/Codes/Sem4/IT150/Lab/Lab_11$ java Main2
  Sum of integers: 12
  Sum of doubles: 6.0
  Dog barks
  Cat meows
○ nithin@nithin1729s:~/Codes/Sem4/IT150/Lab/Lab_11$ 
```

Q3.Write a Java program demonstrating the concept of compile time polymorphism and runtime
polymorphism.

Code

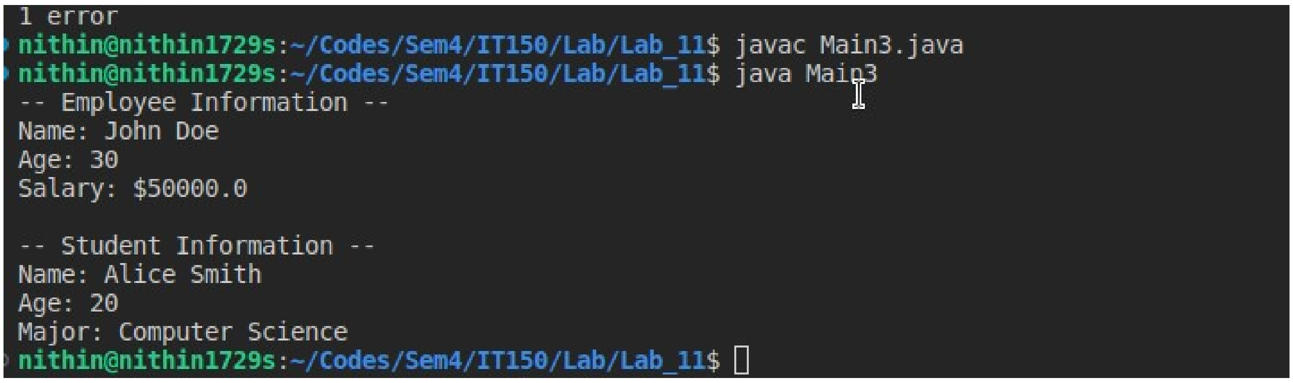


```java
// Parent class
class Person {
    String name;
    int age;

    // Constructor
    Person(String name, int age) {
        this.name = name;
        this.age = age;
    }

    // Method to display information
    void displayInfo() {
        System.out.println("Name: " + name);
        System.out.println("Age: " + age);
    }
}

// Child class inheriting from Person - Employee
class Employee extends Person {
    double salary;

    // Constructor
    Employee(String name, int age, double salary) {
        super(name, age);
        this.salary = salary;
    }

    // Method to display employee information
    void displayEmployeeInfo() {
        System.out.println("-- Employee Information --");
        displayInfo(); // Calling method of the superclass
        System.out.println("Salary: $" + salary);
    }
}

// Child class inheriting from Person - Student
class Student extends Person {
    String major;

    // Constructor
    Student(String name, int age, String major) {
        super(name, age);
        this.major = major;
    }

    // Method to display student information
    void displayStudentInfo() {
        System.out.println("-- Student Information --");
```



```java
}

// Child class inheriting from Person - Student
class Student extends Person {
    String major;

    // Constructor
    Student(String name, int age, String major) {
        super(name, age);
        this.major = major;
    }

    // Method to display student information
    void displayStudentInfo() {
        System.out.println("-- Student Information --");
        displayInfo(); // Calling method of the superclass
        System.out.println("Major: " + major);
    }
}

public class Main {
    public static void main(String[] args) {
        // Creating an Employee object
        Employee emp = new Employee("John Doe", 30, 50000);
        emp.displayEmployeeInfo(); // Displaying employee information

        System.out.println(); // Adding a newline for clarity

        // Creating a Student object
        Student student = new Student("Alice Smith", 20, "Computer Science");
        student.displayStudentInfo(); // Displaying student information
    }
}
```

OUTPUT



```
1 error
nithin@nithin1729s:~/Codes/Sem4/IT150/Lab/Lab_11$ javac Main3.java
nithin@nithin1729s:~/Codes/Sem4/IT150/Lab/Lab_11$ java Main3
-- Employee Information --
Name: John Doe
Age: 30
Salary: $50000.0

-- Student Information --
Name: Alice Smith
Age: 20
Major: Computer Science
nithin@nithin1729s:~/Codes/Sem4/IT150/Lab/Lab_11$
```