National University of Computer & Emerging Sciences, Karachi

Computer Science Department

Fall 2023, Lab Manual – 06

Course Code: CL-1004	Course: Object Oriented Programming Lab
Instructor(s):	Shafique Rehman

LAB - 6

Inheritance and Aggregation

Inheritance:

Inheritance in Java is a mechanism in which one object acquires all the properties and behaviors of a parent object.

The idea behind inheritance in Java is that you can create new classes that are built upon existing classes. When you inherit from an existing class, you can reuse methods and fields of the parent class. Moreover, you can add new methods and fields in your current class also.

Inheritance represents the IS-A relationship which is also known as a parent-child relationship.

Terms used in Inheritance

- Class: A class is a group of objects which have common properties. It is a template or blueprint from which objects are created.
- Sub Class/Child Class: Subclass is a class which inherits the other class. It is also called a derived class, extended class, or child class.
- Super Class/Parent Class: Superclass is the class from where a subclass inherits the features. It is also called a base class or a parent class.
- Reusability: A mechanism which facilitates you to reuse the fields and methods of the existing class when you create a new class.

The syntax of Java Inheritance

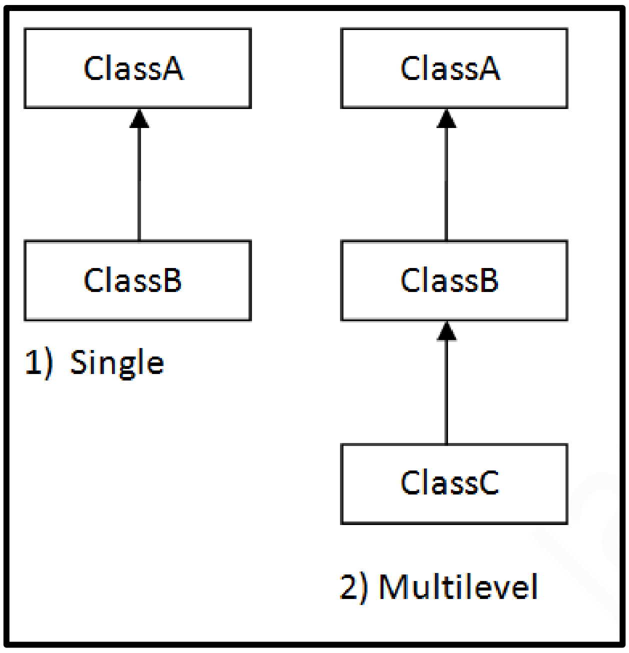
```
class Subclass-name extends Superclass-name
{
  //methods and fields
}
```

The extends keyword indicates that you are making a new class that derives from an existing class.

Types of inheritance in java

On the basis of class, there can be four types of inheritance in java: single, multilevel, hierarchical and multiple inheritance.

In java programming, multiple and hybrid inheritance is supported through interface only.



<u>Single Inheritance Example</u>

When a class inherits another class, it is known as a single inheritance. In the example given below, Dog class inherits the Animal class, so there is the single inheritance.

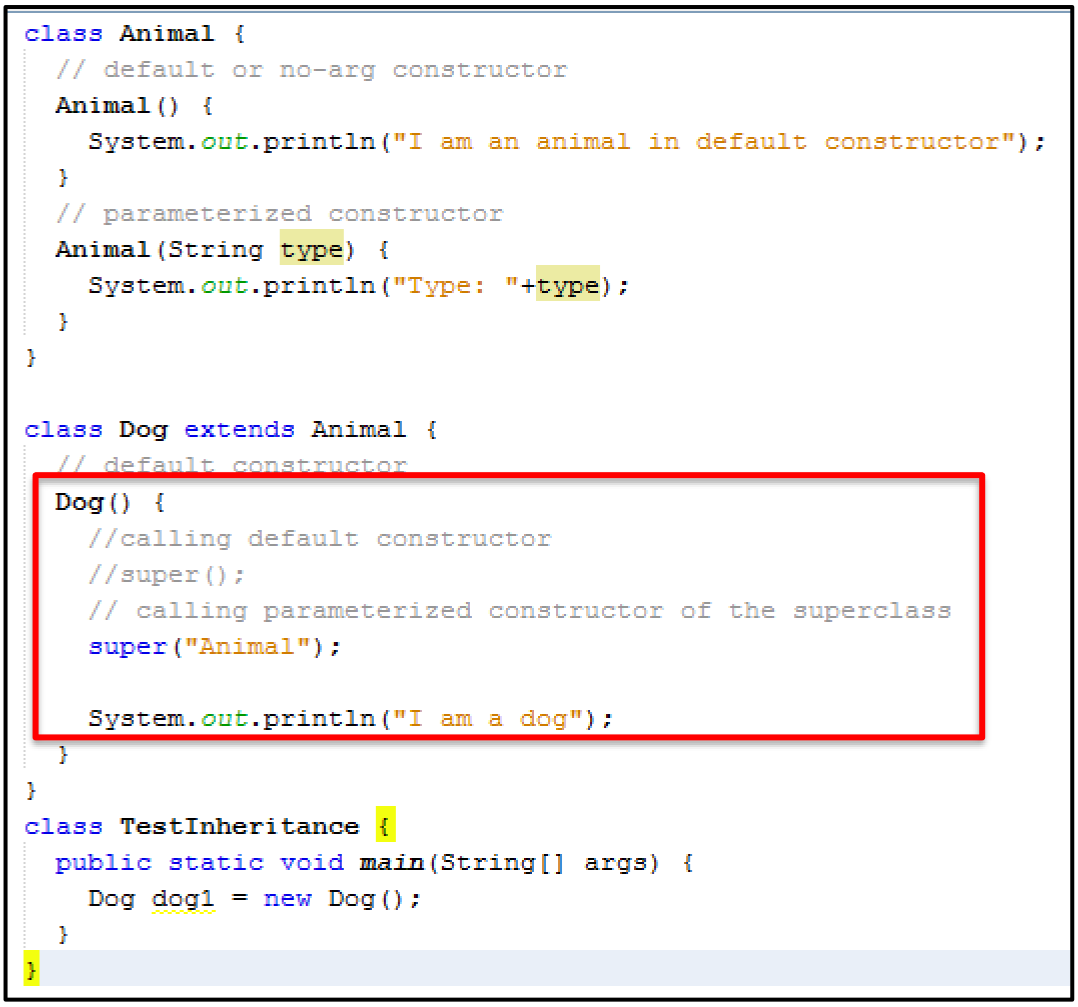
<u>Using super to invoke constructors:</u>

To explicitly call the superclass constructor from the subclass constructor, we use super(). It's a special form of the super keyword.

super() can be used only inside the subclass constructor and must be the first statement.

The compiler can automatically call the no-argument constructor. However, it cannot call parameterized constructors.

If a parameterized constructor has to be called, we need to explicitly define it in the subclass constructor.

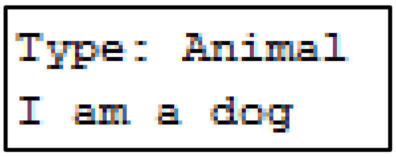


```
class Animal {
  // default or no-arg constructor
  Animal() {
    System.out.println("I am an animal in default constructor");
  }
  // parameterized constructor
  Animal(String type) {
    System.out.println("Type: "+type);
  }
}

class Dog extends Animal {
  // default constructor
  Dog() {
    //calling default constructor
    //super();
    // calling parameterized constructor of the superclass
    super("Animal");

    System.out.println("I am a dog");
  }
}
class TestInheritance {
  public static void main(String[] args) {
    Dog dog1 = new Dog();
  }
}
```

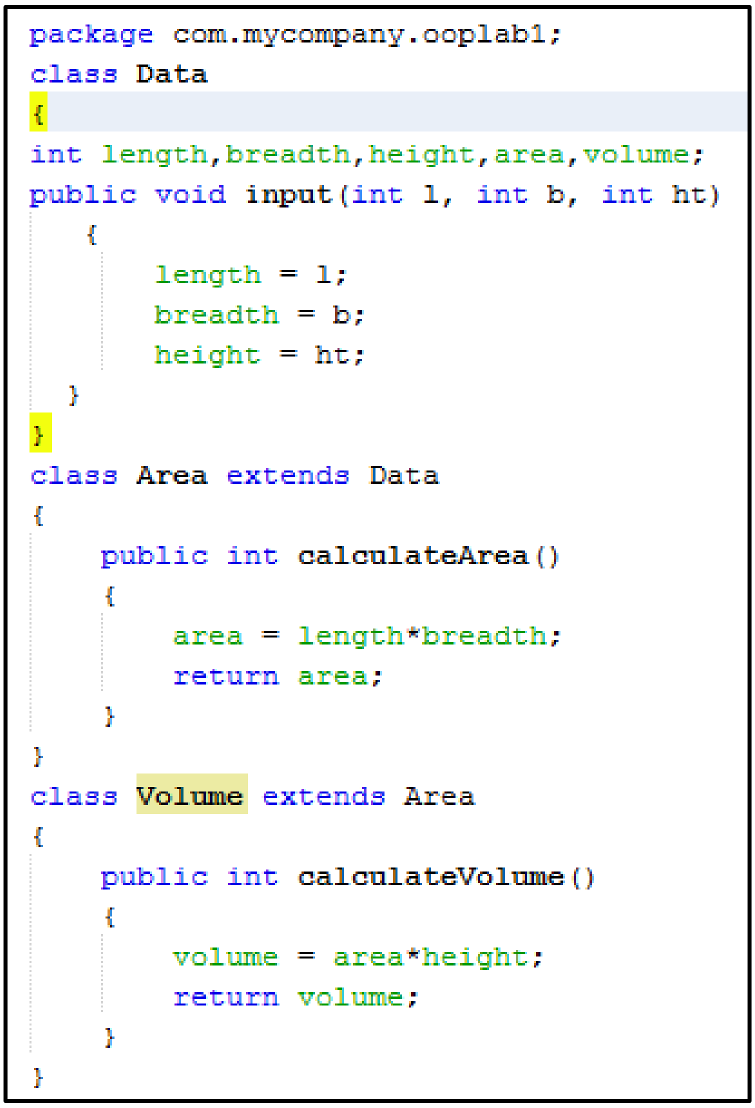
Output:



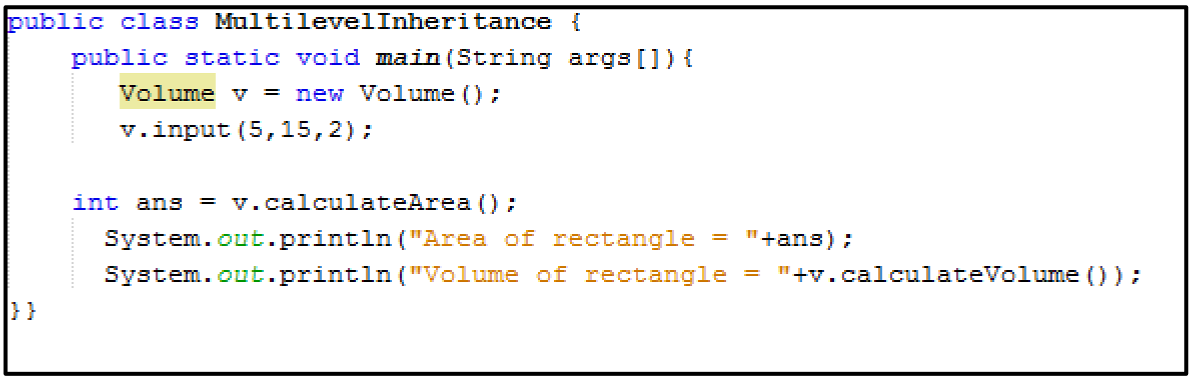
```
Type: Animal
I am a dog
```

Multilevel Inheritance Example

When there is a chain of inheritance, it is known as multilevel inheritance. As you can see in the example given below, BabyDog class inherits the Dog class which again inherits the Animal class, so there is a multilevel inheritance.



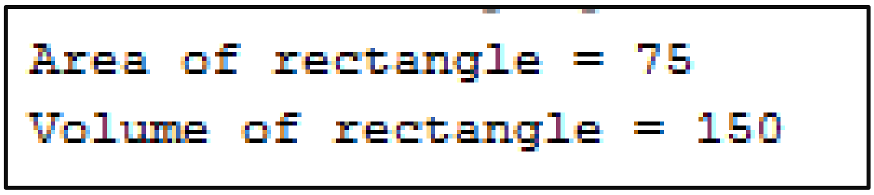
```
package com.mycompany.ooplab1;
class Data
{
int length,breadth,height,area,volume;
public void input(int l, int b, int ht)
   {
       length = l;
       breadth = b;
       height = ht;
  }
}
class Area extends Data
{
    public int calculateArea()
    {
        area = length*breadth;
        return area;
    }
}
class Volume extends Area
{
    public int calculateVolume()
    {
        volume = area*height;
        return volume;
    }
}
```



```
public class MultilevelInheritance {
    public static void main(String args[]){
       Volume v = new Volume();
       v.input(5,15,2);

    int ans = v.calculateArea();
      System.out.println("Area of rectangle = "+ans);
      System.out.println("Volume of rectangle = "+v.calculateVolume());
}}
```

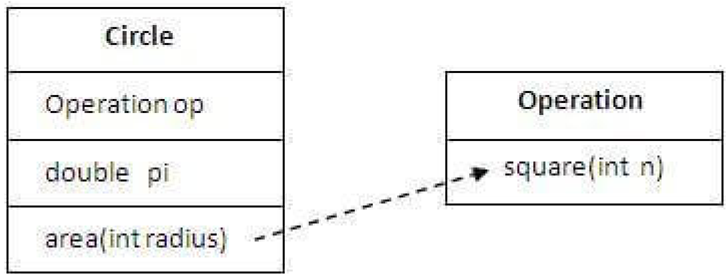
Output:



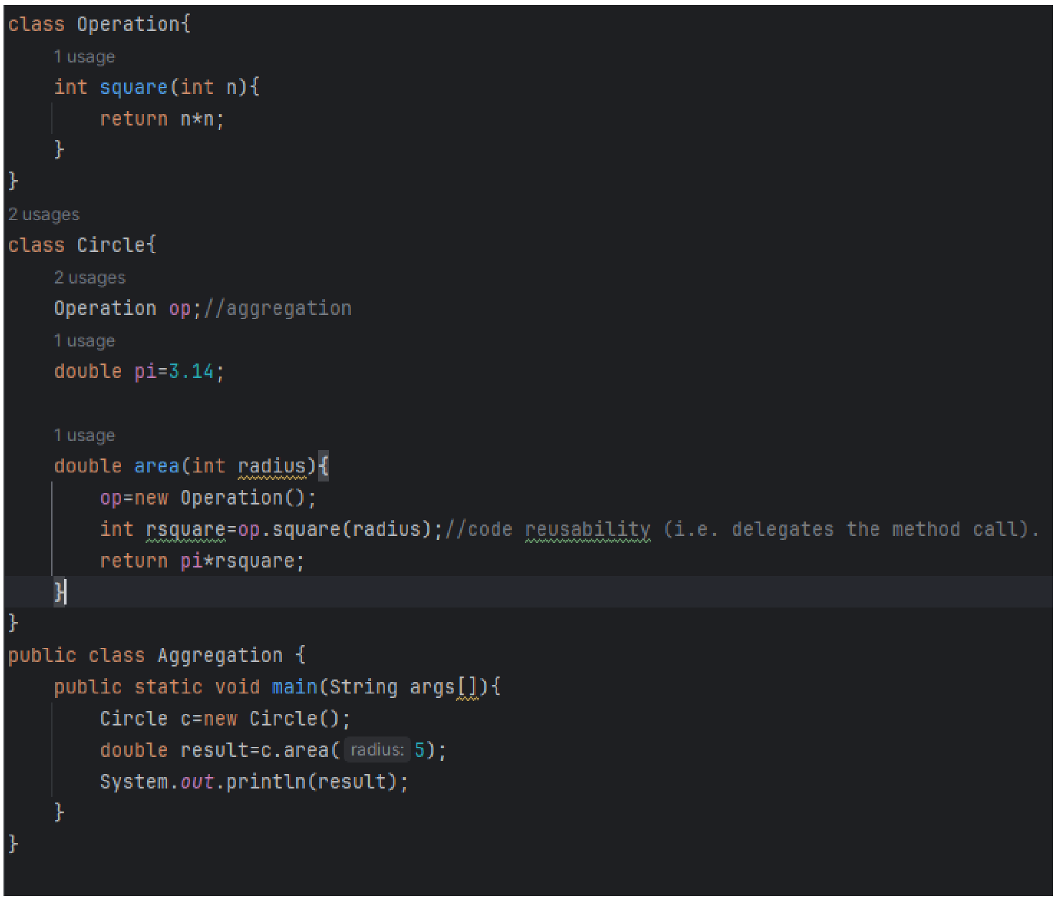
```
Area of rectangle = 75
Volume of rectangle = 150
```

Aggregation

If a class has an entity reference, it is known as Aggregation. Aggregation represents HAS-A relationship. It is used for code reusability.



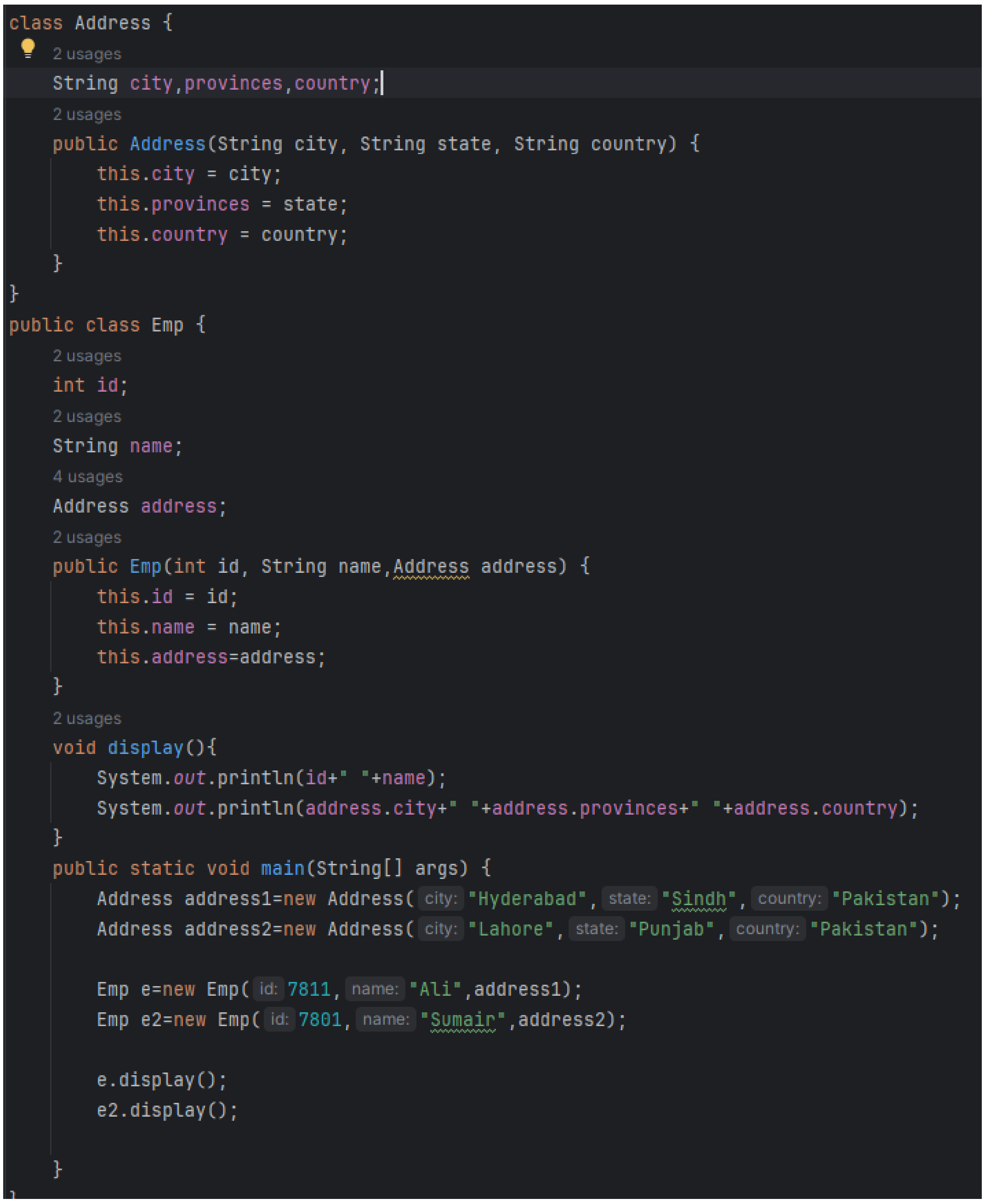
Example 1:



```
class Operation{
    1 usage
    int square(int n){
        return n*n;
    }
}
2 usages
class Circle{
    2 usages
    Operation op;//aggregation
    1 usage
    double pi=3.14;

    1 usage
    double area(int radius){
        op=new Operation();
        int rsquare=op.square(radius);//code reusability (i.e. delegates the method call).
        return pi*rsquare;
    }
}
public class Aggregation {
    public static void main(String args[]){
        Circle c=new Circle();
        double result=c.area( radius: 5);
        System.out.println(result);
    }
}
```

Example 2:



```
class Address {
    2 usages
    String city,provinces,country;
    2 usages
    public Address(String city, String state, String country) {
        this.city = city;
        this.provinces = state;
        this.country = country;
    }
}
public class Emp {
    2 usages
    int id;
    2 usages
    String name;
    4 usages
    Address address;
    2 usages
    public Emp(int id, String name,Address address) {
        this.id = id;
        this.name = name;
        this.address=address;
    }
    2 usages
    void display(){
        System.out.println(id+" "+name);
        System.out.println(address.city+" "+address.provinces+" "+address.country);
    }
    public static void main(String[] args) {
        Address address1=new Address( city: "Hyderabad", state: "Sindh", country: "Pakistan");
        Address address2=new Address( city: "Lahore", state: "Punjab", country: "Pakistan");

        Emp e=new Emp( id: 7811, name: "Ali",address1);
        Emp e2=new Emp( id: 7801, name: "Sumair",address2);

        e.display();
        e2.display();

    }
}
```