



# CSN-103: Fundamentals of Object Oriented Programming

**Instructor: Dr. Rahul Thakur**

Assistant Professor, Computer Science and Engineering, IIT Roorkee

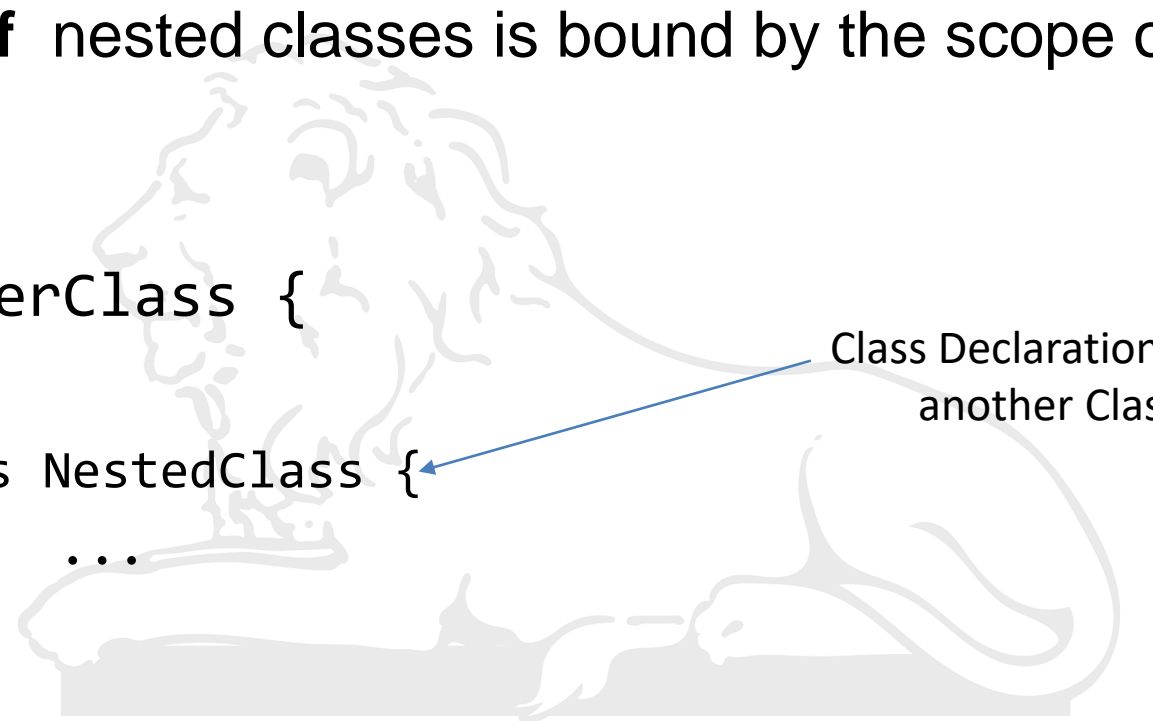


# Nested Class

- It is possible to declare a class **within** another class
  - Such classes are known as **Nested Classes**
- **Scope of** nested classes is bound by the scope of enclosing class

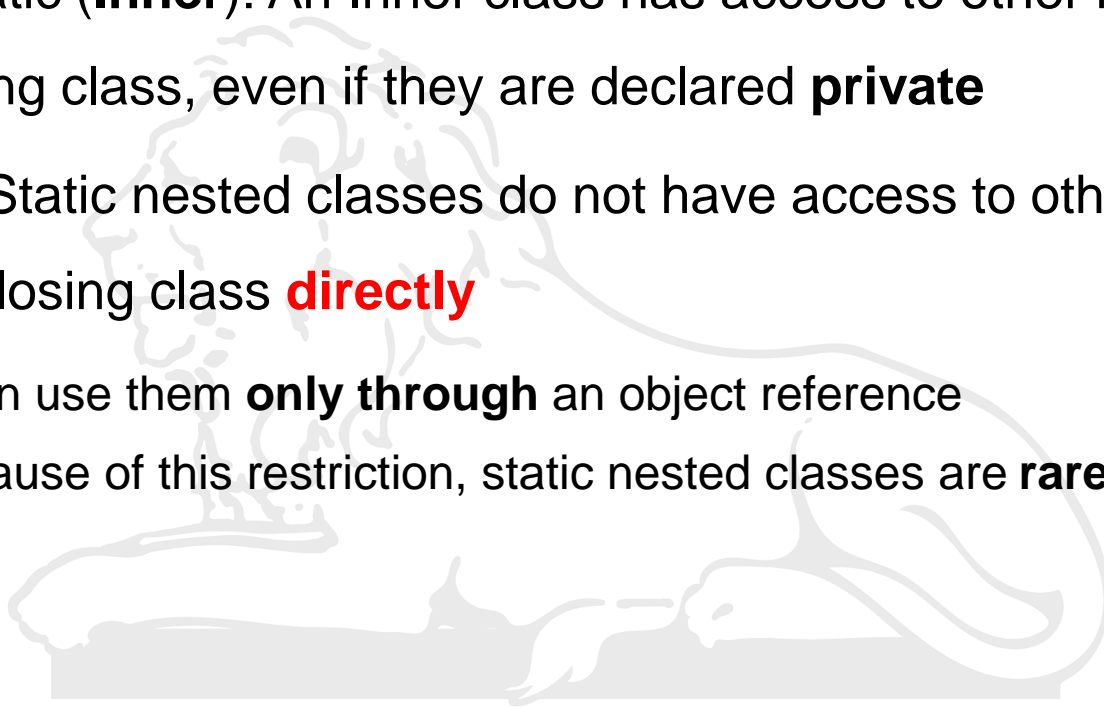
```
class OuterClass {  
    ...  
    class NestedClass {  
        ...  
    }  
}
```

Class Declaration Inside  
another Class

A faint, stylized illustration of a lion's head and body, serving as a background for the code snippet.

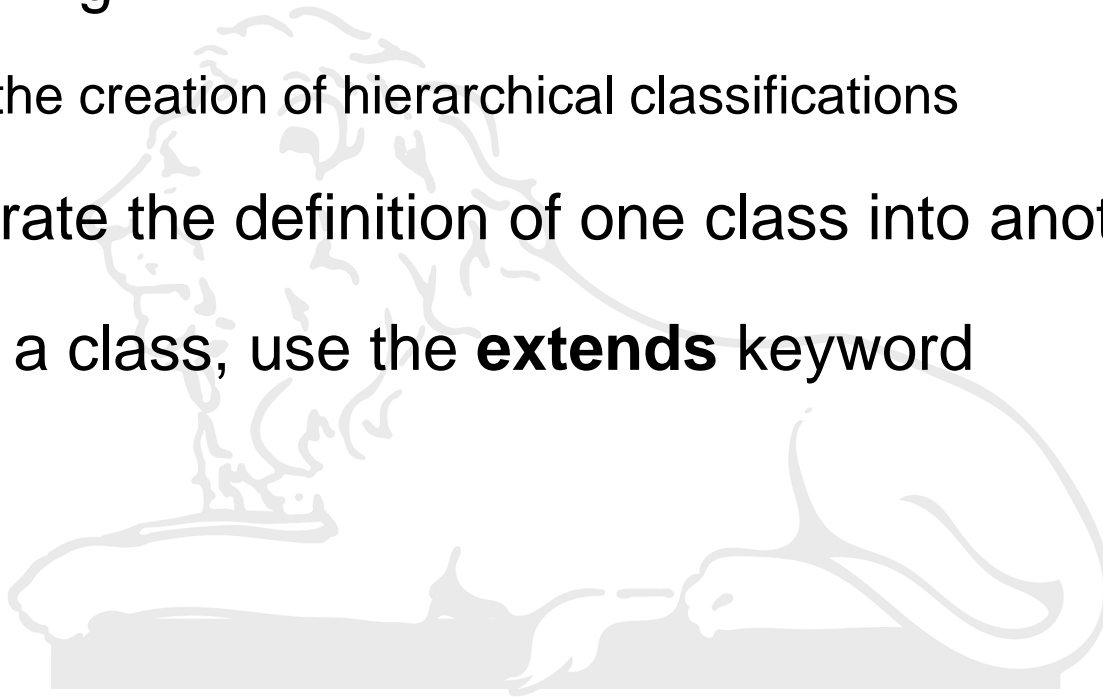
# Types of Nested Class

- Two types of nested class
  - Non-static (**Inner**): An Inner class has access to other members of the enclosing class, even if they are declared **private**
  - Static: Static nested classes do not have access to other members of the enclosing class **directly**
    - It can use them **only through** an object reference
    - Because of this restriction, static nested classes are **rarely** used



# Inheritance

- **Inheritance** is one of the **cornerstones** of object-oriented programming
  - Allows the creation of hierarchical classifications
- It incorporate the definition of one class into another class
- To inherit a class, use the **extends** keyword



# Inheritance Basics

```
class A {  
    int i, j;  
    void showij() {  
        System.out.println("i and j:"+i+" "+j);  
    }  
}
```

```
class B extends A {  
    int k;  
    void showk() {  
        System.out.println("k: " + k);  
    }  
    void sum() {  
        System.out.println("i+j+k: "+(i+j+k));  
    }  
}
```

# Subclass and Superclass

- As you can see, the **subclass B** includes all of the members of its **superclass, A**
  - **subOb** can access *i* and *j* and call **showij( )**
  - Inside **sum( )**, *i* and *j* can be referred to directly
- Superclass **or** Base Class, Subclass **or** Derived Class
- General form of a **class** declaration that inherits a superclass

```
class subclass-name extends superclass-name {  
    // body of class  
}
```

# Subclass and Superclass

- Even though **A** is a **superclass** for **B**
  - It is also a completely independent, stand-alone class
  - A subclass can be a superclass for another subclass
  - Can create a hierarchy of inheritance: A subclass becomes a superclass of another subclass
  - Java does not support the inheritance of multiple superclasses into a single subclass
    - **Only one superclass for any subclass**
  - No class can be a superclass of itself

# Member Access and Inheritance

- A subclass includes all of the members of its superclass
  - Cannot access the **private** members of the superclass
  - Members should be public or **protected**

*A class member that has been declared as **private** will remain **private to its class**. It is not accessible by **any code outside its class**, including **subclasses***



# Private Members and Inheritance

```
class A {  
    int i;                // package-private by default  
    private int j;        // private to A  
    void setij(int x, int y) {  
        i = x;  
        j = y;  
    }  
}
```

```
class B extends A {  
    int total;  
    void sum() {  
        total = i + j;    // ERROR  
    }  
}
```

Use **protected** keyword