Inner class

Friday, September 12, 2025 3:34 PM

Definition

An **inner class** is a class declared **inside another class**.
It acts as a member of the outer class and can access its fields and methods (even private ones).
They are used to logically group classes that are only used in one place, improve encapsulation, and make code more readable.

Why Use Inner Classes

- To logically group classes together.
- To increase **encapsulation** (hide inner workings).
- To write more **readable and maintainable** code.
- To directly access outer class's members.

Types of Inner Classes

Type	Syntax Example	Key Features
1. Non-static (Regular) Inner Class	class Outer { class Inner { … } }	- Needs an instance of outer class to be created. - Can access **all** members (including private) of the outer class.
2. Static Nested Class	class Outer { static class Inner { … } }	- Declared with static. - Does **not** need outer object to be created. - Can access only **static** members of outer class directly.
3. Local Inner Class	Defined **inside a method or block**	- Scope limited to the block/method. - Can access final/effectively final variables of that block.
4. Anonymous Inner Class	new Interface/Class() { … }	- **No name**. - Declared and instantiated in one step. - Used for quick, one-time implementation of interface or subclass.

Examples

1. Non static inner class

```java
// non-static inner class
class Home{  2 usages  new *
    String material ;  1 usage
    String type ;  1 usage

    public Home(String material , String type){  1 usage  new *
        this.material = material ;
        this.type = type ;
    }
}
```

```java
// non-static inner class object creation
Land.Home myHome = myLand.new Home( material: "concrete" ,  type: "villa") ;
```

2. Static inner class

```java
// static inner class
static class RegionDetails{  2 usages  new *
    String region ;  1 usage
    String precipitation ;  1 usage

    public RegionDetails(String region , String precipitation){
        this.region = region ;
        this.precipitation = precipitation ;
    }
}
```

```java
// non-static inner class object creation
Land.Home myHome = myLand.new Home( material: "concrete" ,  type: "villa") ;
```

3. Local inner class

```java
// local inner class
class Maintainance {  2 usages  new *
    int noOfWorker ;  1 usage
    String timeDuration ;  1 usage

    public Maintainance(int noOfWorker , String timeDuration){
        this.noOfWorker = noOfWorker ;
        this.timeDuration = timeDuration ;
    }

    public void cleanHome(){  1 usage  new *
        System.out.println("cleanning done !!");
    }

}
```

```java
// creation of a local inner class object
Maintainance service = new Maintainance( noOfWorker: 2,  timeDuration: "30 min") ;

// method
public void sunday(){  3 usages  1 override  new *
    service.cleanHome();
}
```

4. Anonymous inner class

```java
// anonymous inner class object creation
Land myNewLand = new Land( area: 268 ,  soilType: "Red soil"){  new *
    @Override  3 usages  new *
    public void sunday() {
        super.sunday();
        System.out.println("renovation of garden done !!");
    }
} ;
```