# Inner classes in java: Anonymous inner and static nested class

## ***What is an inner class?***

* **Inner class** are defined inside the body of another class (known as **outer class**).
* These classes can have access modifier or even can be marked as abstract and final.
* Inner classes have special relationship with outer class instances. This relationship allows them to have access to outer class members including private members too.

Inner classes can be defined in four different following ways as mentioned below:

1. **Inner class  
   2) Method – local inner class  
   3) Anonymous inner class  
   4) Static nested class**



## ***1) Inner class***

An inner class is declared inside the curly braces of another enclosing class. Inner class is coded inside a Top level class.

* **Inner class** acts as a member of the enclosing class and can have any access ***modifiers: abstract, final, public, protected, private, static***.  
  Inner class can access all members of the outer class including those marked private

//Top level class definition

class MyOuterClassDemo {

private int myVar= 1;

// inner class definition

class MyInnerClassDemo {

public void seeOuter () {

System.out.println("Value of myVar is :" + myVar);

}

} // close inner class definition

} // close Top level class definition

Inner Class Example 2:

class CPU {

double price;

// nested class

class Processor{

// members of nested class

double cores;

String manufacturer;

double getCache(){

return 4.3;

}

}

// nested protected class

protected class RAM{

// members of protected nested class

double memory;

String manufacturer;

double getClockSpeed(){

return 5.5;

}

}

}

public class Main {

public static void main(String[] args) {

// create object of Outer class CPU

CPU cpu = new CPU();

// create an object of inner class Processor using outer class

CPU.Processor processor = cpu.new Processor();

// create an object of inner class RAM using outer class CPU

CPU.RAM ram = cpu.new RAM();

System.out.println("Processor Cache = " + processor.getCache());

System.out.println("Ram Clock speed = " + ram.getClockSpeed());

}

}

In the above program, there are **two nested classes: Processor and RAM** inside the **outer class: CPU**. We can declare the inner class as protected. Hence, we have declared the RAM class as protected.

Inside the Main class, we first created an instance of an outer class CPU named cpu.

Using the instance of the outer class, we then created objects of inner classes:

* CPU.Processor processor = cpu.new Processor;
* CPU.RAM ram = cpu.new RAM();

**Note**: We use the dot (.) operator to create an instance of the inner class using the outer class.

***Accessing Members of Outer Class within Inner Class:***

We can access the members of the outer class by using this keyword.

class Car {

String carName;

String carType;

// assign values using constructor

public Car(String name, String type) {

this.carName = name;

this.carType = type;

}

// private method

private String getCarName() {

return this.carName;

}

// inner class

class Engine {

String engineType;

void setEngine() {

// Accessing the carType property of Car

if(Car.this.carType.equals("4WD")){

// Invoking method getCarName() of Car

if(Car.this.getCarName().equals("Crysler")) {

this.engineType = "Smaller";

} else {

this.engineType = "Bigger";

}

}else{

this.engineType = "Bigger";

}

}

String getEngineType(){

return this.engineType;

}

}

}

public class Main {

public static void main(String[] args) {

// create an object of the outer class Car

Car car1 = new Car("Mazda", "8WD");

// create an object of inner class using the outer class

Car.Engine engine = car1.new Engine();

engine.setEngine();

System.out.println("Engine Type for 8WD= " + engine.getEngineType());

Car car2 = new Car("Crysler", "4WD");

Car.Engine c2engine = car2.new Engine();

c2engine.setEngine();

System.out.println("Engine Type for 4WD = " + c2engine.getEngineType());

}

}

if(Car.this.carType.equals("4WD")) {...}

We are using this keyword to access the carType variable of the outer class. You may have noticed that instead of using this.carType we have used Car.this.carType.

# Static Nested Class

In Java, we can also define a static class inside another class. Such class is known as static nested class. Static nested classes are not called static inner classes.

Unlike inner class, a static nested class cannot access the member variables of the outer class. It is because the **static nested class** doesn't require you to create an instance of the outer class.

OuterClass.NestedClass obj = new OuterClass.NestedClass();

Here, we are creating an object of the **static nested class** by simply using the class name of the outer class. Hence, the outer class cannot be referenced using OuterClass.this.

### Example: Static Inner Class

class MotherBoard {

// static nested class

static class USB{

int usb2 = 2;

int usb3 = 1;

int getTotalPorts(){

return usb2 + usb3;

}

}

}

public class Main {

public static void main(String[] args) {

// create an object of the static nested class

// using the name of the outer class

MotherBoard.USB usb = new MotherBoard.USB();

System.out.println("Total Ports = " + usb.getTotalPorts());

}

}

In the above program, we have created a static class named USB inside the class MotherBoard. Notice the line,

MotherBoard.USB usb = new MotherBoard.USB();

Here, we are creating an object of USB using the name of the outer class.

Now, let's see what would happen if you try to access the members of the outer class:

### Example 4: Accessing members of Outer class inside Static Inner Class

class MotherBoard {

String model;

public MotherBoard(String model) {

this.model = model;

}

// static nested class

static class USB{

int usb2 = 2;

int usb3 = 1;

int getTotalPorts(){

// accessing the variable model of the outer classs

if(MotherBoard.this.model.equals("MSI")) {

return 4;

}

else {

return usb2 + usb3;

}

}

}

}

public class Main {

public static void main(String[] args) {

// create an object of the static nested class

MotherBoard.USB usb = new MotherBoard.USB();

System.out.println("Total Ports = " + usb.getTotalPorts());

}

}

When we try to run the program, we will get an error:

error: non-static variable this cannot be referenced from a static context

This is because we are not using the object of the outer class to create an object of the inner class. Hence, there is no reference to the outer class Motherboard stored in Motherboard.this.

### Key Points to Remember

* Java treats the inner class as a regular member of a class. They are just like methods and variables declared inside a class.
* Since inner classes are members of the outer class, you can apply any access modifiers like private, protected to your inner class which is not possible in normal classes.
* Since the nested class is a member of its enclosing outer class, you can use the dot (.) notation to access the nested class and its members.
* Using the nested class will make your code more readable and provide better encapsulation.
* Non-static nested classes (inner classes) have access to other members of the outer/enclosing class, even if they are declared private.