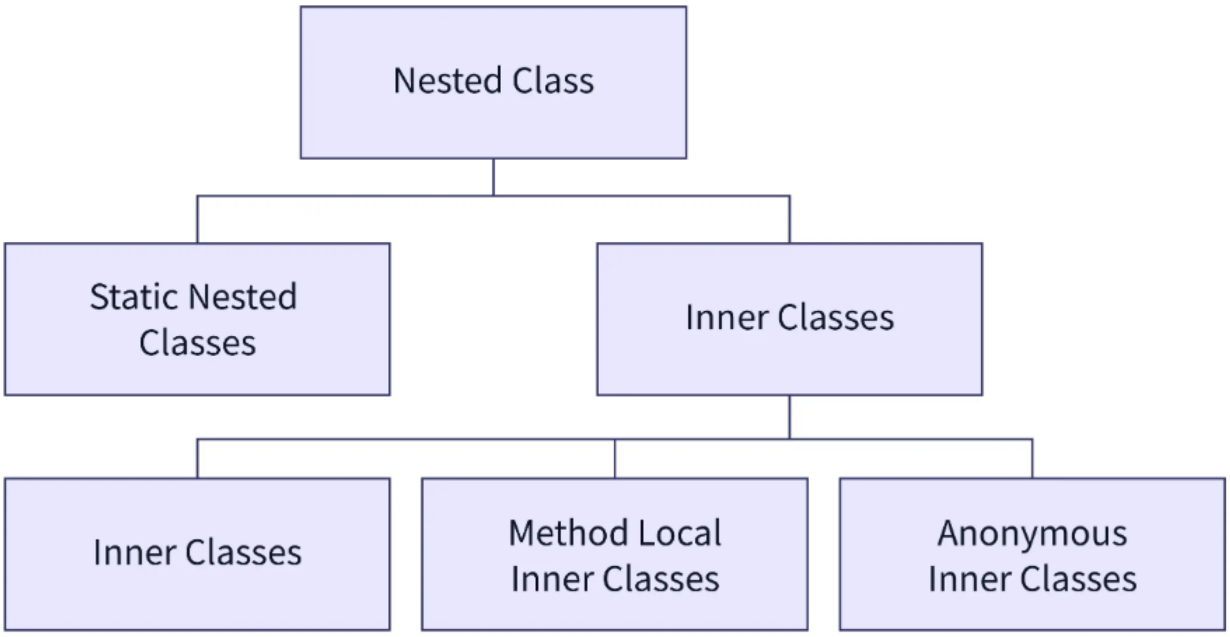
27-Nested Class

Author: Vincent Lau

Learning Objectives

> Understand and able to code all types of nested classes
>
> Understand the difference between Inner Class and Static Nested Class
>
> Understand the purpose of using Anonymous Inner Class
>
> Understand how Anonymous Inner Class applied to Interface and Class



Introduction

A nested class is a class defined within the scope of another class. It can be any class, including inner classes and static nested classes.

Static Nested Class

- A static nested class is defined within another class and marked as static. It **does not have access to the instance members of the outer class, but accessible to static variables.**
- It behaves like a top-level class in terms of access and instantiation.

```java
class Outer {
    private int x = 10;
    private static int y = 100;

    static class StaticNested {
        // does not have access to the instance members of the outer class (x)
        void display() {
            System.out.println("I am Static Nested Class, static variable=" + y);
        }
    }
}
```

```java
// You DON'T NEED to create Outer Object before creating object of static
```

```
nested class
Outer.StaticNested nestedObject = new Outer.StaticNested();
nestedObject.display(); // I am Static Nested Class, static variable=100
```

Inner Class

Instance Inner Class

- An instance inner class (or Inner Class) is defined within another class and **has access to the instance members of the outer class**.
- It requires an instance of the outer class to be created.

```
class Outer {
    private int x = 10;

    class Inner {
        // has access to the instance members of the outer class
        void display() {
            System.out.println("Instance Inner Class, variable=" + x);
        }
    }
}
```

```
// You NEED to create Outer Object before creating object of inner class
Outer outer = new Outer();
Outer.Inner inner = outer.new Inner(); // use outer object to create another object
inner.display(); // Instance Inner Class, variable=10
```

Local Inner Class

A local inner class is defined within a method or scope block. It's only accessible within the block where it's defined and has access to the local variables of that block.

```
class OuterClass {

    void methodWithLocalInner() {
        int localVar = 5;

        class LocalInner {
            int x;

            void display() {
                System.out.println("Local Inner Class: " + localVar + this.x);
            }
        }

        LocalInner localInner = new LocalInner();
        localInner.display();
    }
}
```

Anonymous Inner Class

An anonymous inner class is a type of **inner class that does not have a name**. It's typically used for providing an implementation of an interface or extending a class inline.

One of the typical examples using Anonymous Inner Class is Thread & Runnable. We will practice Anonymous Inner Class again in Thread Chapter.

Interface Example

```
interface Action {
    void read();
    void run();
}

public class AnonymousInnerClassExample {
    public static void main(String[] args) {
        // Skip creating a new Class implements Interface Reading
        Action person = new Action() {
            int count;

            @Override
            public void run() {
                // codes ...
                this.count++;
                System.out.println("x=" + this.count);
            }

            @Override
            public void read() {
                System.out.println("Anonymous Inner Class for Interface Reading");
```

```java
            }
        };

        person.read();
        person.run();
    }
}
```

Class Example

```java
class Parent {
    void read() {
        System.out.println("Parent Class");
    }
}

public class AnonymousInnerClassExample {
    public static void main(String[] args) {
        // Skip creating a SubClass extends Class Parent
        Parent parent = new Parent() {
            @Override
            void read() {
                System.out.println("Anonymous Inner Class for Class Person");
            }
        };
        parent.read();
    }
}
```

These are the different types of nested classes in Java. They allow you to organize and encapsulate your code, providing varying levels of encapsulation and access to the members of the outer class.

Key Differences & Summary

- **Static Nested Class**: Acts like a top-level class. No access to instance members, but able to access static variables.
- **Instance Inner Class**: Requires an instance of the outer class, can access instance members.
- **Local Inner Class**: Defined within a method, can access local variables of the method.
- **Anonymous Inner Class**: Inline implementation, often for interfaces or subclasses.

Each type of nested class serves different purposes and offers different levels of encapsulation and access to the members of the outer class.