Inner Classes Overview

An **Inner Class** is a class defined within another class. Inner classes allow more modularization, abstraction, security, shareability, and reusability in Java. They can be categorized into the following types:

1. **Member Inner Class**
2. **Static Inner Class**
3. **Method Local Inner Class**
4. **Anonymous Inner Class**

Advantages of Inner Classes

1. **Modularization**: Inner classes allow a class to be split into smaller, specialized components.
    - Example: You can declare different types of math concepts like Algebra, Trigonometry, etc., within a `Maths` class.
2. **Abstraction**: Inner classes provide a level of abstraction. Variables and methods defined within an inner class are only accessible within the inner class itself, enhancing encapsulation.
    - Example: In `class A`, if an inner class `B` declares variables, they are not accessible in the outer class `A`.
3. **Security**: Inner classes can be declared private, enhancing security. This is not possible with outer classes.
    - Example: Inner classes can be declared `private`, restricting their access to other classes.
4. **Shareability**: Static inner classes can be used to share data without needing an instance of the outer class. This is not possible with outer classes.
5. **Reusability**: Inner classes support inheritance relationships and can be compiled into separate `.class` files, aiding in reusability.

Types of Inner Classes

1. Member Inner Class

A non-static class inside another class.

- **Syntax**:

```
java
CopyEdit
class Outer {
    class Inner {
        // Inner class implementation
    }
}
```

- To instantiate a member inner class:

```
java
CopyEdit
OuterClass.InnerClass refVar = new OuterClass().new
InnerClass();
```

- **Example**:

```
java
CopyEdit
class A {
    class B {
        void display() {
            System.out.println("Inside Inner class
B");
        }
    }
}

public class Main {
    public static void main(String[] args) {
        A a = new A();
        A.B b = a.new B();
        b.display(); // Output: Inside Inner class B
    }
}
```

2. Static Inner Class

A static class inside another class.

- **Syntax**:

```
java
CopyEdit
class Outer {
    static class Inner {
        // Inner class implementation
    }
}
```

- Static inner classes do not require an instance of the outer class to be instantiated.
- **Example**:

```
java
CopyEdit
class A {
    static class B {
        void display() {
            System.out.println("Inside Static Inner
class B");
        }
    }
}

public class Main {
    public static void main(String[] args) {
        A.B b = new A.B();
        b.display(); // Output: Inside Static Inner
class B
    }
}
```

3. Method Local Inner Class

A class defined within a method.

- **Scope**: The class is only accessible within the method where it is defined.
- **Example**:

```
java
CopyEdit
class A {
    void method() {
        class B {
```

```
            void display() {
                System.out.println("Inside Method
Local Inner class B");
            }
        }
        B b = new B();
        b.display(); // Output: Inside Method Local
Inner class B
    }
}

public class Main {
    public static void main(String[] args) {
        A a = new A();
        a.method();
    }
}
```

4. Anonymous Inner Class

An unnamed class that provides the implementation for methods of an abstract class or interface.

- **Syntax**:

```
java
CopyEdit
interface MyInterface {
    void method();
}

MyInterface obj = new MyInterface() {
    public void method() {
        System.out.println("Implementation of
method");
    }
};
```

- **Example**:

```
java
CopyEdit
interface I {
    void m1();
```

```
}

public class Main {
    public static void main(String[] args) {
        I i = new I() {
            public void m1() {
                System.out.println("Anonymous class
implementation");
            }
        };
        i.m1(); // Output: Anonymous class
implementation
    }
}
```

Common Questions and Concepts

Can Abstract Classes and Interfaces Be Inner Classes?

- Yes, both **abstract classes** and **interfaces** can be declared inside another class. Their implementation must be provided within the same outer class or by a subclass.

Can Inheritance Be Applied to Inner Classes?

- Yes, inheritance relationships can exist between inner classes, but they must follow certain conditions (e.g., both classes should reside in the same outer class, or the outer class must provide a subclass).

Declaring Inner Classes in Abstract Classes or Interfaces:

- It is possible to declare an inner **abstract class** or **interface** inside an abstract class or interface, and they can be implemented in the subclass of the outer class or interface.

Key Takeaways:

- Inner classes in Java help with **modularization**, **abstraction**, **security**, **shareability**, and **reusability**.

- Java provides different types of inner classes (member, static, method local, and anonymous), each with unique characteristics and usage scenarios.
- Inner classes are powerful tools for organizing code and can enhance functionality and readability.