Java Inner Class:

In Java, an inner class is a class defined within another class. Inner classes have access to the members (fields and methods) of the outer class, including its private members. Inner classes are often used to logically group classes and can improve encapsulation and code organization. There are several types of inner classes in Java, including:

1. **Non-static (Inner) Class**: These are inner classes that are not marked as `static`. They have access to the instance members of the outer class and can be instantiated only within an instance of the outer class.

2. **Static Nested Class**: These are inner classes that are marked as `static`. They do not have access to instance members of the outer class and can be instantiated without creating an instance of the outer class.

Here are two examples illustrating each type of inner class:

Example 1: Non-static (Inner) Class

```java
public class OuterClass {
    private int outerField = 10;

    public class InnerClass {
        public void display() {
            System.out.println("Value of outerField: " + outerField);
        }
    }

    public static void main(String[] args) {
        OuterClass outer = new OuterClass();
        OuterClass.InnerClass inner = outer.new InnerClass();
        inner.display();
    }
}
```

In this example, we have an `OuterClass` containing an inner class `InnerClass`. The `InnerClass` has access to the `outerField` of the `OuterClass`, which is a private field.

To create an instance of the inner class, we first create an instance of the outer class and then create an instance of the inner class using `outer.new InnerClass()`. The `display()` method of `InnerClass` can access the `outerField` of the outer class.

Example 2: Static Nested Class

```java
public class OuterClass {
    private static int staticOuterField = 20;

    public static class StaticNestedClass {
        public void display() {
            System.out.println("Value of staticOuterField: " + staticOuterField);
        }
    }

    public static void main(String[] args) {
        OuterClass.StaticNestedClass nested = new OuterClass.StaticNestedClass();
        nested.display();
    }
}
```

In this example, we have an `OuterClass` containing a `StaticNestedClass`, which is marked as `static`. The `StaticNestedClass` does not have access to instance members of the `OuterClass`, but it can access static members.

To create an instance of the static nested class, we can do so directly using `OuterClass.StaticNestedClass`. The `display()` method of `StaticNestedClass` can access the `staticOuterField` of the `OuterClass`.

These examples demonstrate the basic concepts of Java inner classes and the difference between non-static and static nested classes. Inner classes can be a powerful tool for organizing and encapsulating code in Java applications.