JAVA NESTED CLASSES(INNER CLASSES)

- nested class is a class that is declared inside the class or interface.
- We use inner classes to logically group classes and interfaces in one place to be more readable and maintainable.
- **It can access all the members of the outer class, including private data members and methods.**

INTRO.

```
class Java_Outer_class{
    //code
    class Java_Inner_class{
        //code
    }
}
```

SYNTAX OF INNER CLASS

1. Nested classes represent a particular type of relationship that is **it can access all the members (data members and methods) of the outer class**, including private.
2. Nested classes are used **to develop more readable and maintainable code** because it logically group classes and interfaces in one place only.
3. **Code Optimization**: It requires less code to write.

ADVANTAGE OF JAVA INNER CLASSES

- users need to program a class in such a way so that no other class can access it

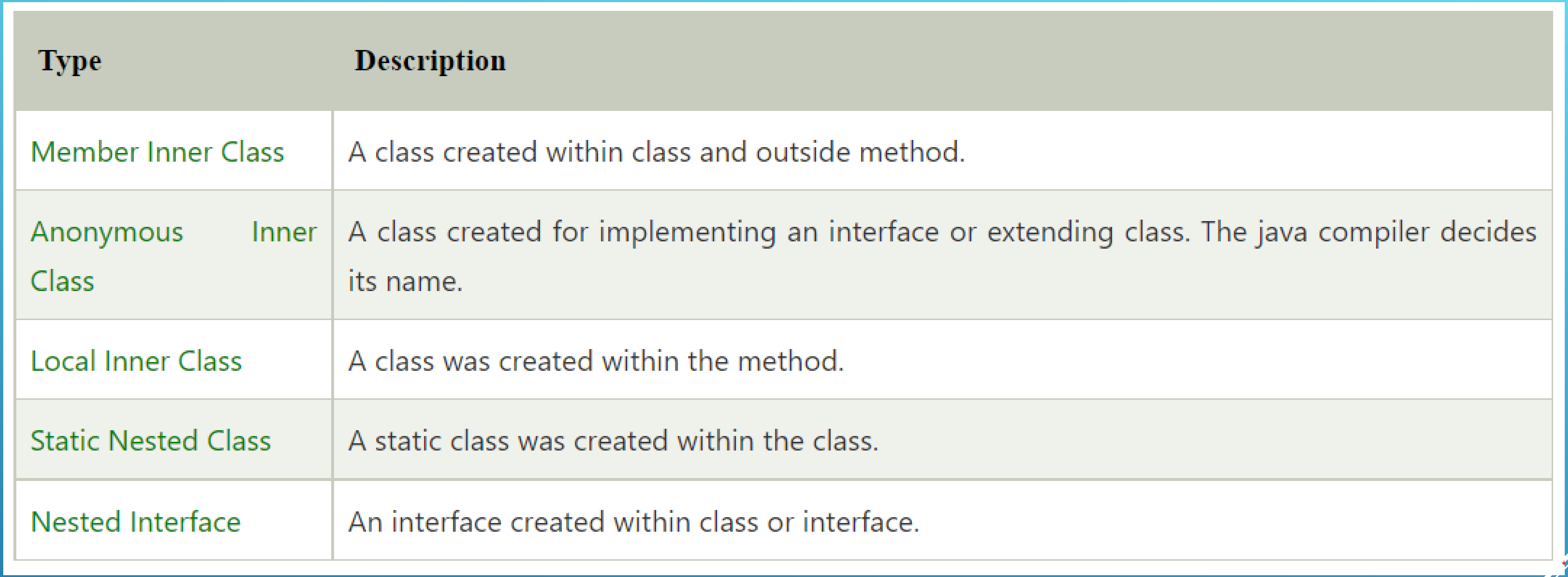
NEED

- An inner class is a part of a nested class.
- Non-static nested classes are known as inner classes.

DIFFERENCE BETWEEN NESTED CLASS AND INNER CLASS IN JAVA

- **Non-static nested class (inner class)**
  - Member inner class
  - Anonymous inner class
  - Local inner class
- **Static nested class**

TYPES OF NESTED CLASSES



Type	Description
Member Inner Class	A class created within class and outside method.
Anonymous Inner Class	A class created for implementing an interface or extending class. The java compiler decides its name.
Local Inner Class	A class was created within the method.
Static Nested Class	A static class was created within the class.
Nested Interface	An interface created within class or interface.

- A non-static class that is created inside a class.
- But outside a method.
- It is also known as a **regular inner class**.
- It can be declared with access modifiers like public, default, private, and protected.

MEMBER INNER CLASS

```
class Outer{
    //code
    class Inner{
        //code
    }
}
```

SYNTAX

```
class Outer{
    private int data=30;
    class Inner{
        void msg(){     System.out.println("data is "+data);  }
    }
    public static void main(String args[]){
    Outer obj=new Outer();
    Outer.Inner in=obj.new Inner();
    in.msg();
    } }
```

EXAMPLE

- An object or instance of a member's inner class always exists within an object of its outer class.
- The new operator is used to create the object of member inner class with slightly different syntax.
- **SYNTAX:** OuterClassRef.**new** InnerClassConstructor();
- **Example:** obj.**new** Inner();

INTERNAL CODE

- The java compiler creates two class files in the case of the inner class.
- The Java compiler creates a class file named **Outer$Inner** in this case.
- We must have to create the instance of the outer class.
- The inner class has the reference of Outer class that is why it can access all the data members of Outer class including private

INTERNAL CODE

```
import java.io.PrintStream;
class Outer$Inner
{
    final Outer this$0;
    Outer$Inner()
    {   super();
        this$0 = Outer.this;
    }
    void msg()
    {
        System.out.println((new StringBuilder()).append("data is ")
                    .append(Outer.access$000(Outer.this)).toString());
    }
}
```

INTERNAL CODE EXAMPLE

- Java anonymous inner class is an inner class without a name.
- And for which only a single object is created.
- An anonymous inner class can be useful when making an instance of an object with certain "extras" such as overloading methods of a class or interface, without having to actually subclass a class.
- A class that has no name is known as an anonymous inner class in Java.

ANONYMOUS INNER CLASS

- It should be used if you have to override a method of class or interface. Java Anonymous inner class can be created in two ways:

1. Class (may be abstract or concrete).
2. Interface.

```
abstract class Person{
  abstract void eat();
}
class Test {
 public static void main(String args[]){
        Person p=new Person(){
        void eat(){System.out.println("nice fruits");}
        };
  p.eat();
 }
}
```

EXAMPLE

- A class is created, but its name is decided by the compiler, which extends the Person class and provides the implementation of the eat() method.
- An object of the Anonymous class is created that is referred to by 'p,' a reference variable of Person type.

```
import java.io.PrintStream;
static class Test$1 extends Person
{
   TestAnonymousInner$1(){}
   void eat()
    {
        System.out.println("nice fruits");
    }
}
```

INTERNAL CODE

```
interface Eatable{
 void eat();
}
class Test1{
        public static void main(String args[]){
                Eatable e=new Eatable(){
                public void eat(){System.out.println("nice fruits");}
        };
        e.eat();
 }
}
```

USING AN INTERFACE

- A class i.e., created inside a method, is called local inner class in java.
- Created inside a block.
- Sometimes this block can be a for loop, or an if clause.
- Local Inner classes are not a member of any enclosing classes.
- Local inner classes cannot have any access modifiers associated with them.
- They can be marked as final or abstract.
- These classes have access to the fields of the class enclosing it.

LOCAL INNER CLASS

```
public class local_o {
        private int data=30;
        void display(){
                class Local{
                        void msg(){System.out.println(data);}
                }
                Local l=new Local();
                l.msg();
        }
 public static void main(String args[]){
  local_o obj=new local_o();
  obj.display();
 }
}
```

EXAMPLE

```
import java.io.PrintStream;
class local_o$Local
{
    final local_o this$0;
    local_o$Local()
    {
        super();
        this$0 = Simple.this;
    }
    void msg()
    {
        System.out.println(localInner1.access$000(local_o.this));
    }
}
```

INTERNAL CODE

1) Local inner class cannot be invoked from outside the method.

RULES FOR JAVA LOCAL INNER CLASS

```
class localInner2{
 private int data=30;//instance variable
 void display(){
        int value=50
        class Local{
                void msg(){System.out.println(value);
                }
        }
  Local l=new Local();
  l.msg();
 }
 public static void main(String args[]){
  localInner2 obj=new localInner2();
  obj.display();
 }
}
```

EXAMPLE

- A static class is a class that is created inside a class, is called a static nested class in Java.
- It cannot access non-static data members and methods.
- It can be accessed by outer class name.
- It can access static data members of the outer class, including private.
- The static nested class cannot access non-static (instance) data members

STATIC NESTED CLASS

```
class Outer{
  static int data=30;
  static class Inner{
   void msg(){System.out.println("data is "+data);}
  }
  public static void main(String args[]){
  Outer.Inner obj=new Outer.Inner();
  obj.msg();
  }
}
```

EXAMPLE

```
public class Outer2{
 static int data=30;
 static class Inner{
   static void msg(){System.out.println("data is "+data);}
  }
 public static void main(String args[]){
 Outer2.Inner.msg();
//no need to create the instance of static nested class
 }
}
```

EXAMPLE

```
import java.io.PrintStream;
static class TestOuter1$Inner
{
TestOuter1$Inner(){}
void msg(){
System.out.println((new StringBuilder()).append("data is ")
.append(TestOuter1.data).toString());
}
}
```

INTERNAL CODE