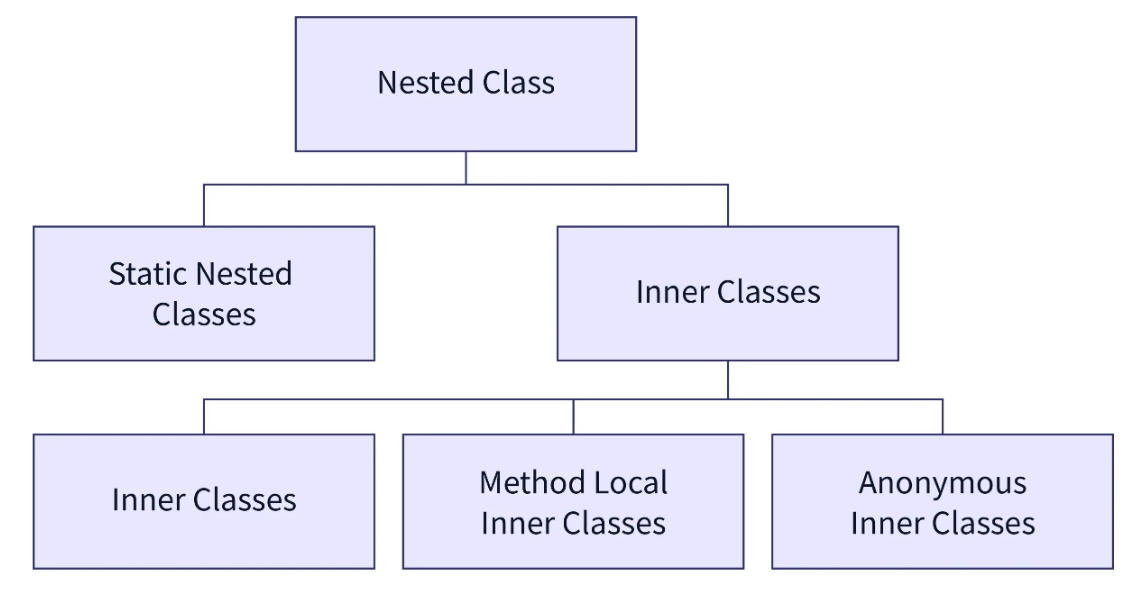
In Java, an anonymous class is a local class without a name that can be used to define and instantiate a class at the same time. An anonymous inner class is one that has no name and produces only one object.

Anonymous classes can be useful for providing quick implementations of interfaces or abstract classes in situations where creating a named class would be overly verbose or cumbersome.

An Anonymous Class is a nested class that doesn't have any name. An anonymous class must be defined within another class, thats why it is also known as an Anonymous Inner Class in Java.



**Syntax:**

// Test can be interface,abstract/concrete class

Test t = new Test()

{

// data members and methods

public void test\_method()

{

........

........

}

};

**Now let us do discuss the difference between regular class(normal classes) and Anonymous Inner class**

* **A normal class can implement any number of interfaces but the anonymous inner class can implement only one interface at a time.**
* **A regular class can extend a class and implement any number of interfaces simultaneously. But anonymous Inner class can extend a class or can implement an interface but not both at a time.**
* **For regular/normal class, we can write any number of constructors but we can’t write any constructor for anonymous Inner class because the anonymous class does not have any name and while defining constructor class name and constructor name must be same.**

**Types of Anonymous Class in Java**

Based on declaration and behavior, there are 3 types of anonymous Inner classes:

1. Anonymous Inner class that extends a class
2. Anonymous Inner class that implements an interface
3. Anonymous Inner class that defines inside method/constructor argument

**Type 1:**Anonymous Inner class that extends a class

We can have an anonymous inner class that extends a class. For example, we know that we can create a thread by extending a Thread class. Suppose we need an immediate thread but we don’t want to create a class that extends [Thread class](https://www.geeksforgeeks.org/java-lang-thread-class-java/) all the time. With the help of this type of Anonymous Inner class, we can define a ready thread.

**Example:**

// Java program to illustrate creating an immediate thread  
// Using Anonymous Inner class that extends a Class  
  
// Main class  
class MyThread {  
  
 // Main driver method  
 public static void main(String[] args)  
 {  
 // Using Anonymous Inner class that extends a class  
 // Here a Thread class  
 Thread t = new Thread() {  
  
 // run() method for the thread  
 public void run()  
 {  
 // Print statement for child thread  
 // execution  
 System.*out*.println("Child Thread");  
 }  
 };  
  
 // Starting the thread  
 t.start();  
  
 // Displaying main thread only for readability  
 System.*out*.println("Main Thread");  
 }  
}

**Output:**

Main Thread

Child Thread

**Type 2:**Anonymous Inner class that implements an interface

We can also have an anonymous inner class that implements an interface. For example, we also know that by implementing Runnable interface we can create a Thread. Here we use an anonymous Inner class that implements an interface.

**Example:**

// Java program to illustrate defining a thread  
// Using Anonymous Inner class that implements an interface  
  
// Main class  
class MyThread {  
  
 // Main driver method  
 public static void main(String[] args)  
 {  
 // Here we are using Anonymous Inner class  
 // that implements a interface i.e. Here Runnable  
 // interface  
 Runnable r = new Runnable() {  
  
 // run() method for the thread  
 public void run()  
 {  
 // Print statement when run() is invoked  
 System.*out*.println("Child Thread");  
 }  
 };  
  
 // Creating thread in main() using Thread class  
 Thread t = new Thread(r);  
  
 // Starting the thread using start() method  
 // which invokes run() method automatically  
 t.start();  
  
 // Print statement only  
 System.*out*.println("Main Thread");  
 }  
}

**Output:**

Main Thread

Child Thread

**Type 3:**Anonymous Inner class that defines inside method/constructor argument

Anonymous inner classes in method/constructor arguments are often used in graphical user interface (GUI) applications. To get you familiar with syntax lets have a look at the following program that creates a thread using this type of Anonymous Inner class

**Example:**

// Java program to illustrate defining a thread  
// Using Anonymous Inner class that define inside argument  
  
// Main class  
class MyThread {  
 // Main driver method  
 public static void main(String[] args)  
 {  
 // Using Anonymous Inner class that define inside  
 // argument  
 // Here constructor argument  
 Thread t = new Thread(new Runnable() {  
  
 public void run()  
 {  
 System.*out*.println("Child Thread");  
 }  
 });  
  
 t.start();  
  
 System.*out*.println("Main Thread");  
 }  
}

**Output:**

Main Thread

Child Thread