# Anonymous Inner Class in Java

4.1

Prerequisites :- Nested Classes in Java

It 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.

Anonymous inner classes are useful in writing implementation classes for listener interfaces in graphics programming.

Anonymous inner class are mainly created in two ways:

- Class (may be abstract or concrete)
- Interface

**Syntax:** The syntax of an anonymous class expression is like the invocation of a constructor, except that there is a class definition contained in a block of code.

```
// Test can be interface,abstract/concrete class
Test t = new Test()
{
   // data members and methods
   public void test_method()
   {
      .........
      ........
   }
};
```

To understand anonymous inner class, let us take a simple program

```
//Java program to demonstrate need for Anonymous Inner class
interface Age
{
    int x = 21;
    void getAge();
}
class AnonymousDemo
{
    public static void main(String[] args)
    {
        // Myclass is implementation class of Age interface
        MyClass obj=new MyClass();

        // calling getage() method implemented at Myclass
        obj.getAge();
    }
}

// Myclass implement the methods of Age Interface
class MyClass implements Age
{
    @Override
    public void getAge()
    {
        // printing the age
        System.out.print("Age is "+x);
    }
}
```

Run on IDE

In the program, interface Age is created with getAge() method and x=21. Myclass is written as implementation class of Age interface. As done in Program, there is no need to write a separate class Myclass. Instead, directly copy the code of Myclass into this parameter, as shown here:

```
Age obj = new Age() {
            @Override
            public void getAge() {
                System.out.print("Age is "+x);
            }
        };
```

Here, an object to Age is not created but an object of Myclass is created and copied in the entire class code as shown above. This is possible only with anonymous inner class. Such a class is called 'anonymous inner class', so here we call 'Myclass' as anonymous inner class.

**Anonymous inner class version of the above Program**

```
//Java program to demonstrate Anonymous Inner class
interface Age
{
    int x = 21;
    void getAge();
}
class AnonymousDemo
{
    public static void main(String[] args) {

        // Myclass is hidden inner class of Age interface
        // whose name is not written but an object to it
        // is created.
        Age obj = new Age() {
            @Override
            public void getAge() {
                // printing  age
                System.out.print("Age is "+x);
            }
        };
        obj.getAge();
    }
}
```

Run on IDE

**Types of anonymous inner class :** Based on declaration and behavior, there are 3 types of anonymous Inner classes:

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 extend Thread class all the time. By the help of this type of Anonymous Inner class we can define a ready thread as follows:

```
//Java program to illustrate creating an immediate thread
//Using Anonymous Inner class that extends a Class
class MyThread
{
    public static void main(String[] args)
    {
        //Here we are using Anonymous Inner class
        //that extends a class i.e. Here a Thread class
        Thread t = new Thread()
        {
            public void run()
            {
                System.out.println("Child Thread");
            }
        };
        t.start();
        System.out.println("Main Thread");
    }
}
```

Run on IDE

Output:

```
Main Thread
Child Thread
OR
Child Thread
Main Thread
```

2. **Anonymous Inner class that implements a 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 anonymous Inner class that implements an interface.

```
//Java program to illustrate defining a thread
//Using Anonymous Inner class that implements an interface
class MyThread
{
    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()
        {
            public void run()
            {
                System.out.println("Child Thread");
            }
        };
        Thread t = new Thread(r);
        t.start();
        System.out.println("Main Thread");
    }
}
```

Run on IDE

Output:

```
Main Thread
Child Thread
OR
Child Thread
Main Thread
```

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 on the following program that creates a thread using this type of Anonymous Inner class :

```
//Java program to illustrate defining a thread
//Using Anonymous Inner class that define inside argument
class MyThread
{
    public static void main(String[] args)
    {
        //Here we are 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");
    }
}
```

Run on IDE

Output:

```
Main Thread
Child Thread
OR
Child Thread
Main Thread
```

**Difference between Normal/Regular class and Anonymous Inner class:**

- A normal class can implement any number of interfaces but anonymous inner class can implement only one interface at a time.
- A regular class can extend a class and implement any number of interface 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 cant write any constructor for anonymous Inner class because anonymous class does not have any name and while defining constructor class name and constructor name must be same.

**Accessing Local Variables of the Enclosing Scope, and Declaring and Accessing Members of the Anonymous Class**

Like local classes, anonymous classes can capture variables; they have the same access to local variables of the enclosing scope:

- An anonymous class has access to the members of its enclosing class.
- An anonymous class cannot access local variables in its enclosing scope that are not declared as final or effectively final.
- Like a nested class, a declaration of a type (such as a variable) in an anonymous class shadows any other declarations in the enclosing scope that have the same name.

Anonymous classes also have the same restrictions as local classes with respect to their members:

- We cannot declare static initializers or member interfaces in an anonymous class.
- An anonymous class can have static members provided that they are constant variables.

Note that you can declare the following in anonymous classes:

- Fields
- Extra methods (even if they do not implement any methods of the supertype)
- Instance initializers
- Local classes