Java Inner Class

Java inner class or nested class is a class i.e. declared inside the class or interface.

We use inner classes to logically group classes and interfaces in one place so that it can be more readable and maintainable.

Additionally, it can access all the members of outer class including private data members and methods.

Syntax of Inner class

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

Advantage of java inner classes

There are basically three advantages of inner classes in java. They are as follows:

1) Nested classes represent a special type of relationship that is **it can access all the members (data members and methods) of 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.

Do You Know

- What is the internal code generated by the compiler for member inner class ?
- What are the two ways to create annonymous inner class ?
- Can we access the non-final local variable inside the local inner class ?
- How to access the static nested class ?

- Can we define an interface within the class ?
- Can we define a class within the interface ?

Difference between nested class and inner class in Java

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

Types of Nested classes

There are two types of nested classes non-static and static nested classes.The non-static nested classes are also known as inner classes.

1. Non-static nested class(inner class)
   - a)Member inner class
   - b)Annomynous inner class
   - c)Local inner class
2. Static nested class

Type	Description
Member Inner Class	A class created within class and outside method.
Anonymous Inner Class	A class created for implementing interface or extending class. Its name is compiler.
Local Inner Class	A class created within method.
Static Nested Class	A static class created within class.
Nested Interface	An interface created within class or interface.

Java Member inner class

A non-static class that is created inside a class but outside a method is called member inner class.

Syntax:

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

Java Member inner class example

In this example, we are creating msg() method in member inner class that is accessing the private data member of outer class.

```java
class TestMemberOuter1{
 private int data=30;
 class Inner{
  void msg(){System.out.println("data is "+data);}
 }
 public static void main(String args[]){
  TestMemberOuter1 obj=new TestMemberOuter1();
  TestMemberOuter1.Inner in=obj.new Inner();
  in.msg();
 }
}
```

Test it Now

Output:

```
data is 30
```

Internal working of Java member inner class

The java compiler creates two class files in case of inner class. The class file name of inner class is "Outer$Inner". If you want to instantiate inner class, you must have to create the instance of outer class. In such case, instance of inner class is created inside the instance of outer class.

Internal code generated by the compiler

The java compiler creates a class file named Outer$Inner in this case. The Member inner class have the reference of Outer class that is why it can access all the data members of Outer class including private.

```java
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());
    }
}
```

Java Anonymous inner class

A class that have no name is known as anonymous inner class in java. It should be used if you have to override method of class or interface. Java Anonymous inner class can be created by two ways:

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

Java anonymous inner class example using class

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

Test it Now

Output:

```
nice fruits
```

Internal working of given code

```
Person p=new Person(){
void eat(){System.out.println("nice fruits");}
};
```

1. 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.
2. An object of Anonymous class is created that is referred by p reference variable of Person type.

Internal class generated by the compiler

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

Java anonymous inner class example using interface

```
interface Eatable{
 void eat();
}
class TestAnnonymousInner1{
 public static void main(String args[]){
 Eatable e=new Eatable(){
  public void eat(){System.out.println("nice fruits");}
 };
 e.eat();
 }
}
```

Test it Now

Output:

```
nice fruits
```

Internal working of given code

It performs two main tasks behind this code:

```java
Eatable p=new Eatable(){
void eat(){System.out.println("nice fruits");}
};
```

1. A class is created but its name is decided by the compiler which implements the Eatable interface and provides the implementation of the eat() method.
2. An object of Anonymous class is created that is referred by p reference variable of Eatable type.

Internal class generated by the compiler

```java
import java.io.PrintStream;
static class TestAnonymousInner1$1 implements Eatable
{
TestAnonymousInner1$1(){}
void eat(){System.out.println("nice fruits");}
}
```

Java Local inner class

A class i.e. created inside a method is called local inner class in java. If you want to invoke the methods of local inner class, you must instantiate this class inside the method.

Java local inner class example

```
public class localInner1{
 private int data=30;//instance variable
 void display(){
  class Local{
   void msg(){System.out.println(data);}
  }
  Local l=new Local();
  l.msg();
 }
 public static void main(String args[]){
  localInner1 obj=new localInner1();
  obj.display();
 }
}
```

Test it Now

Output:

```
30
```

Internal class generated by the compiler

In such case, compiler creates a class named Simple$1Local that have the reference of the outer class.

```
import java.io.PrintStream;
class localInner1$Local
{
    final localInner1 this$0;
    localInner1$Local()
    {
        super();
        this$0 = Simple.this;
    }
```

```
   void msg()
   {
       System.out.println(localInner1.access$000(localInner1.this));
   }
}
```

Rule: Local variable can't be private, public or protected.

Rules for Java Local Inner class

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

2) Local inner class cannot access non-final local variable till JDK 1.7. Since JDK 1.8, it is possible to access the non-final local variable in local inner class.

Example of local inner class with local variable

```
class localInner2{
 private int data=30;//instance variable
 void display(){
  int value=50;//local variable must be final till jdk 1.7 only
  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();
 }
}
```

Test it Now