Java Inner Classes

Java Inner Classes (Nested Classes)

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

Syntax

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

Advantage of Java inner classes

- 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.
- Nested classes are used to develop more readable and maintainable code because it logically group classes and interfaces in one place only.
- Code Optimization: It requires less code to write.

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.

i. Non-static nested class (inner class)
   a. Member inner class
   b. Anonymous inner class
   c. Local inner class

ii. 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.

Java Member Inner class

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

Syntax

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

Example

```
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();
 }
}
```

Output:

```
data is 30
```

Java Anonymous inner class

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.

In simple words, a class that has no name is known as an anonymous inner class in Java. 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

Example

```
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();
 }
}
```

```
nice fruits
```

Internal working

```
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 the Anonymous class is created that is referred to by 'p,' a reference variable of Person type.

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();
 }
}
```

Output:

```
nice fruits
```

Java Local inner class

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

If you want to invoke the methods of the local inner class, you must instantiate this class inside the method.

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();
 }
}
```

Rules for Java Local Inner class

- Local variables cannot be private, public, or protected.
- Local inner class cannot be invoked from outside the method.
- 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 the 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();
 }
}
```

Output:

```
50
```

Java static nested class

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.

1. It can access static data members of the outer class, including private.
2. The static nested class cannot access non-static (instance) data members.

Example

```
class TestOuter1{
  static int data=30;
  static class Inner{
   void msg(){System.out.println("data is "+data);}
  }
  public static void main(String args[]){
  TestOuter1.Inner obj=new TestOuter1.Inner();
  obj.msg();
  }
}
```

Output:

```
data is 30
```

Java static nested class example with a static method

```
public class TestOuter2{
  static int data=30;
  static class Inner{
   static void msg(){System.out.println("data is "+data);}
  }
  public static void main(String args[]){
  TestOuter2.Inner.msg();//no need to create the instance of static nested class
  }
}
```

Output:

```
data is 30
```

Java Nested Interface

An interface, i.e., declared within another interface or class, is known as a nested interface. The nested interfaces are used to group related interfaces so that they can be easy to maintain. The nested interface must be referred to by the outer interface or class. It can't be accessed directly.

1. The nested interface must be public if it is declared inside the interface, but it can have any access modifier if declared within the class.
2. Nested interfaces are declared static.

Syntax

Syntax of nested interface which is declared within the interface

```
interface interface_name{
 ...
 interface nested_interface_name{
  ...
 }
}
```

Syntax of nested interface which is declared within the class

```
class class_name{
 ...
 interface nested_interface_name{
  ...
 }
}
```

Example of nested interface which is declared within the interface

```
interface Showable{
  void show();
  interface Message{
   void msg();
  }
}
class TestNestedInterface1 implements Showable.Message{
 public void msg(){System.out.println("Hello nested interface");}

 public static void main(String args[]){
  Showable.Message message=new TestNestedInterface1();//upcasting here
  message.msg();
 }
}
```

Output:

```
hello nested interface
```

Example of nested interface which is declared within the class

```
class A{
  interface Message{
   void msg();
  }
}

class TestNestedInterface2 implements A.Message{
 public void msg(){System.out.println("Hello nested interface");}

 public static void main(String args[]){
  A.Message message=new TestNestedInterface2();//upcasting here
  message.msg();
 }
}
```

Output:

```
hello nested interface
```