**What is inner class?**

1. In **Java**, just like methods, variables of a **class**, can have another **class** as its member.
2. Writing a **class** within another is allowed in java. The **class** written within another is called the nested **class**.
3. The **class** that holds the **inner class** is called the outer **class**.

**Classification of Nested classes**

**Nested classes are divided into two categories:**

**1. Static Inner or Nested Class.**

1. **Static inner class.**

**2. Non-static Inner or Nested class.**

1. Member or non-static inner class.
2. Local inner class.
3. Anonymous inner class.

**1. Introduction of static inner class:**

1. **Static Inner class:**
2. Static inner class instance can be created outside the outer class within the following syntax

Outer.Inner obj=new Outer.Inner();

1. No need to create instance of outer class.

**Programming Example:**

**package** innerclassdemos;

**class** Outer {

**static** **class** Inner {

**public** **void** disp() {

System.***out***.println("Inner class");

}

}

}

**public** **class** StaticInnerClass {

**public** **static** **void** main(String[] args) {

Outer.Inner obj = **new** Outer.Inner();

obj.disp();

}

}

Output:

Inner class.

**2. Introduction of member non-static inner class:**

1. **Member or non-static Inner class.**
2. Step by step process to create a non-static Inner class object.
3. First need to create the instance of Outer class to the outside of Outer class.

Syntax: Outer ob1=new Outer();

1. After that, need to create the instance of Inner class to the outside of Outer class with depend on Outer class object’s reference variable.

Syntax: Outer.Inner ob2=ob1.new Inner();

**Programming Example:**

**package** innerclassdemos;

**class** Outer1 {

**class** Inner {

**public** **void** disp() {

System.***out***.println("Inner class");

}

}

}

**public** **class** NonStaticAnonymousClass {

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

Outer1 o1 = **new** Outer1();

Outer1.Inner o2 = o1.**new** Inner();

o2.disp();

}

}

**Output:**

Inner class

**ii. Local class:**

**1. When defining a class inside a block or method or constructor of the outer class, then defining inner class is called local non-static inner class.**

**Limitation**

**1. Local inner class functionality can be not accessed outside the outer class. We must to create object of inner class inside the inner class body.**

**2. Private , protected access modifier is not applicable to inner class.**

**3. Local inner class can not be static. This is not a common for all because it is local.**

**Advantage:**

**1. If we want to same classes with in a java application in same package then we should go for Local inner class concept. There is not any collision.**

**Compiler file are:**

**Outer.class**

**Outer$1LocalClass.class**

**Outer$2LocalClass.class**

**Non-static local inner class programming example:**

**package** innerclassdemos;

**class** Outer2 {

**public** **void** getData() {

**class** Inner2 {

**public** **void** method1() {

System.***out***.println("Method1 is printed");

}

**public** **void** method2() {

System.***out***.println("Method2 is printed");

}

**public** **void** method3() {

System.***out***.println("Method3 is printed");

}

}

Inner2 inner = **new** Inner2();

inner.method1();

inner.method2();

inner.method3();

}

}

**public** **class** NonStaticLocalInnerClass {

**public** **static** **void** main(String[] args) {

Outer2 outer = **new** Outer2();

outer.getData();

}

}

**Object creation in main method:**

**1. First need to create object of outer class out side the class.**

**Outer outer=new Outer();**

**2. Need to call outer class method with the help of outer class object.**

**outer.method();**

**when we call outer class method then automatically inner class functionality will be executed, because we are created object of inner class inside the method of outer class.**

**iii. Anonymous class:**

1. Anonymous classes in java are more accurately known as anonymous inner class.
2. There’s no such thing as anonymous classes without the “inner”.
3. They are defined inside the other class.
4. Defining a class without any identity, so that class name is not present.
5. We can define anonymous class inside the method of outer class.

**a. Introduction of anonymous class (using interface):**

**Program:**

1. **Here, First need to create instance of Outer class in which anonymous class is defined.**

**Outer outer =new Outer**

1. **After that with the help of outer class object, we are calling anonymous class defining body.**

**outer.annonymousobject.method();**

**package** innerclassdemos;

**interface** Greeting1 {

**void** sayHello();

}

**class** India1 {

Greeting1 gt = **new** Greeting1() {

**public** **void** sayHello() {

System.***out***.println("Namaste");

}

};

}

**public** **class** AnonymousInterfaceClass {

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

India1 india = **new** India1();

india.gt.sayHello();

}

}

**Output:**

Namaste

**b. Introduction of anonymous class(Without interface):**

**Program:**

**package** innerclassdemos;

**class** Greeting {

**void** sayHello() {

System.***out***.println("hello");

}

}

**class** India {

Greeting gt = **new** Greeting() {

**public** **void** sayHello() {

System.***out***.println("Namaste");

}

};

}

**public** **class** AnonymousClass {

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

India india = **new** India();

india.gt.sayHello();

}

}

**Output:**

Namaste

**c. Anonymous class in main method in java:**

**Program example:**

**package** innerclassdemos;

**interface** Abc{

**void** display();

}

**public** **class** AnonymousInMain {

**public** **static** **void** main(String[] args) {

Abc abc=**new** Abc() {

@Override

**public** **void** display() {

System.***out***.println(" Body of anonymous class");

}

};

abc.display();

}

}

**OUTPUT:**

Body of anonymous class

**Advantage of Inner class:**

**1. Setting dependencies of object. With the help of outer class object we can access the inner class functionality.**

**2. When we compile Outer.java then get two files**

**i. outer.class**

**ii. outer$inner.class.(for anonymous class outer$1.class where 1 is called that one anonymous class.)**