Inner Classes

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.

Advantage of inner classes

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

Types of Nested classes

Non-static nested class (inner class)

- **Member inner class**

```
package innerClass;

class MemberInnerClass {
	public int value= 10;
	class InnerClass{
		public void printData(){
			System.out.println("value : "+value);
		}
	}
	public static void main(String []args){
		MemberInnerClass outer = new MemberInnerClass();
		InnerClass inner = outer.new InnerClass();
		inner.printData();
	}
}
```

The java compiler creates two class files in case of inner class. The class file name of inner class is **"OuterClassName$InnerClassName".** 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.

- **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.

Anonymous inner class can be created by two ways:

Class

```
package innerClass;

abstract class ParentClass{
	abstract public void method();
}
class AnonymousInnerClass {
	public static void main(String []args){
		ParentClass obj = new ParentClass(){
			public void method(){
				System.out.println("AnonymousInnerClass...");
			}
		};
		obj.method();
	}
}
```

Interface

```
interface ParentInterface{
	abstract public void method();
}
class AnonymousInnerClassInterfaceImplementation {
	public static void main(String []args){
		ParentInterface obj = new ParentInterface(){
			public void method(){
				System.out.println("AnonymousInnerClass...");
			}
		};
		obj.method();
	}
}
```

- **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.

```
package innerClass;

class ParentClassTest{
	private int value = 10;
	public void method(){
		class LocalClass{
			void printDetails(){
				System.out.println("value : "+value);
			}
		}
		LocalClass local = new LocalClass();
//Local inner class cannot be invoked from outside the method. As scope is
defined limited to method only.
		local.printDetails();
	}
}
class LocalInnerClass {
	public static void main(String []args){
		ParentClassTest obj = new ParentClassTest();
		obj.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 local inner class.

```
class ParentClassTest1{
	private int value = 10;
	public void method(){
		class LocalClass1{
			String localVariable = "Test";
			void printDetails(){
```

```
                        System.out.println("value : "+value + " localVariable
: "+ localVariable);
                  }
            }
            LocalClass1 local = new LocalClass1();
            local.printDetails();
      }
}
class LocalInnerClass1 {
      public static void main(String []args){
            ParentClassTest1 obj = new ParentClassTest1();
            obj.method();
      }
}
```

Static nested class

A static class is created inside a class is called static nested class in java. It cannot access non-static data members and methods. It can be accessed by outer class name.

```
package innerClass;

class staticNestedClass {
      public static int value= 10;
      static class InnerClass{
            public void printData(){
                  System.out.println("value : "+value);
            }
      }
      public static void main(String []args){
            InnerClass inner = new staticNestedClass.InnerClass();
//No need to create the object of Outer class because nested class is static and static
properties, methods or classes can be accessed without object.
            inner.printData();
      }
}
```

Static nested class with static method

```
class staticNestedClass1 {
	public static int value= 10;
	static class InnerClass1{
		public static void printData(){
			System.out.println("value : "+value);
		}
	}
	public static void main(String []args){
		staticNestedClass1.InnerClass1.printData();
	}
}
```