L40

Friday, February 4, 2022 12:57 AM

Non-static Inner Classes, Static Inner Classes

Non-static inner classes

Q1. In Java, we can write a class inside another class. Such classes are called nested classes.

We can classify the nested classes into **four** types :

1. **Inner classes**
2. **Static nested classes**
3. **Local classes**
4. **Anonymous classes**

We will learn about inner classes here and the rest in the later sections.

Nested classes which are not declared as static are called Inner classes.

For example:

class A { *//this is called a top-level class*class B { *//this is an inner class*}
}

Let us consider two top-level classes A and B such that class B needs access to private members of class A and class B is not used by any other class. In such a scenario it makes sense to make B as an inner class of A as given in the above example. By making it an inner class, class B will gain access to all the private members of class A, since B has also become an member of A.

Since **inner classes** are similar to instance fields, create an instance of an inner class we need to first have an instance of the outer class.

See and retype the below code. Note how we are able to access the private field name inside the method getCompleteName() of the inner class Prefixer.

Also note how the instance of the inner class is created by using the new keyword on an instance of the outer class in the line:
Namer.Prefixer prefixer = namer.new Prefixer("Mr.");

```
package q11294;
public class Namer {
    private String name;
    public Namer(String name) {
        this.name = name;
    }
    class Prefixer {
        private String prefix;
        private Prefixer(String prefix) {
            this.prefix = prefix;
        }
        public String getCompleteName() {
            return prefix + " " + name;
        }
    }
    public static void main(String[] args) {
        Namer namer = new Namer("Doodle");
        Namer.Prefixer prefixer = namer.new Prefixer("Mr.");
        System.out.println(prefixer.getCompleteName());
    }
}
```

Static nested classes

Q1. A nested class marked as static is called a **static nested class** and not an inner class.

Nested classes which are not declared as static are called inner classes.

For example:

class A { *//this is called a top-level class*class B { *//this is an inner class*}
 static class C { *//this is a nested class*}
}

In the above code,

1. A is called a **top-level class**
2. B is called an **inner class** and
3. C is called a **static nested class**

Note that the **static nested class** C can only access **static members** of class A and not A's instance members.

Unlike an **inner class** where we need an instance of **outer class** to create an instance of **inner class**, a **static nested class** can be directly created using the **new** keyword.

See and retype the below code to understand the difference in creation of an instance of an **inner class** and a **static nested class.**

Note: A static nested class is almost equal to a top-level class, except that it is always referenced under the context on its outer class.

```java
package q11295;
public class A {
    class B {
        private B() {
            System.out.println("In inner class B's constructor");
        }
    }
    static class C {
        private C() {
            System.out.println("In static nested class C's constructor");
        }
    }
    public static void main(String[] args) {
        A.B b = new A().new B();
        A.C c = new A.C();
    }
}

```