Nested Classes in Java

DT228/3

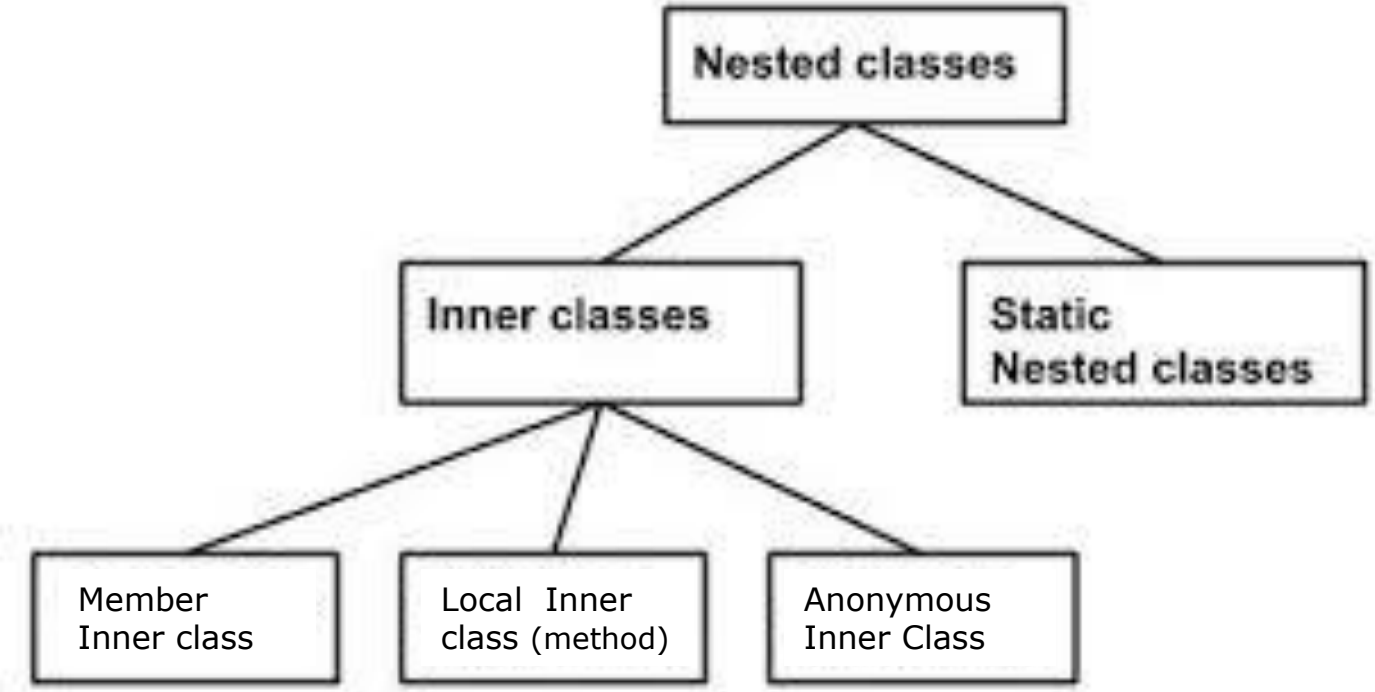
Dr. Susan McKeever

Nested class = class within a class

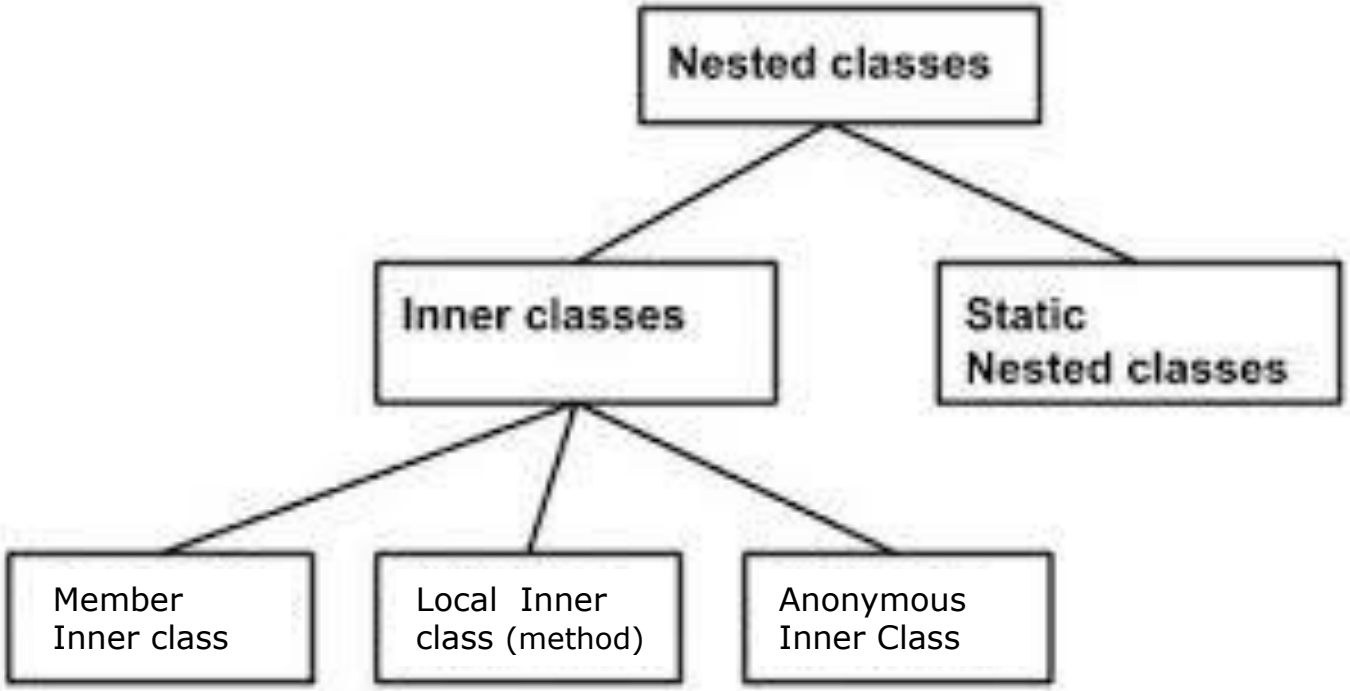
EG

```
public class OuterClass
{
    private String someVar;


     class InnerClass
     {
        private anotherVariable ;

     }

}
```

Types of nested classes



Types of nested classes



Already seen which types..?

Types of nested classes

Type	Description
Member Inner Class	A class created within class and outside method. E.g. custom Adapters in Android
Anonymous Inner Class	A class created for implementing interface or extending class. Its name is decided by the java compiler. E.g. event handlers in Android
Local Inner Class	A class created within method. Can only be instantiated by that method
Static Nested Class	A static class created within class. E.g. Database Helper class in Android

Nested interfaces exist too

Nested Interface	An interface created within class or interface.

Why use Inner Classes?

Logically grouping classes that are only used in one place.

if you want to create class which is only used by the enclosing class, then no sense in creating a separate file for that.

Increases encapsulation.

the **outer class can have private members**that the **inner class can access**

e.g. class A whose attributes are declared private; Class B needs to access them. We can hide class B in A and B can access members of A in spite of the fact that they are private. Also B can be hidden from outside world when declared private.

Lead to more readable and maintainable code.

Instance of the inner class can only be created from an instance of the outer class

Why use Inner Classes?

Logically grouping classes that are only used in one place. e.g. Custom Adapter used by a listActivity

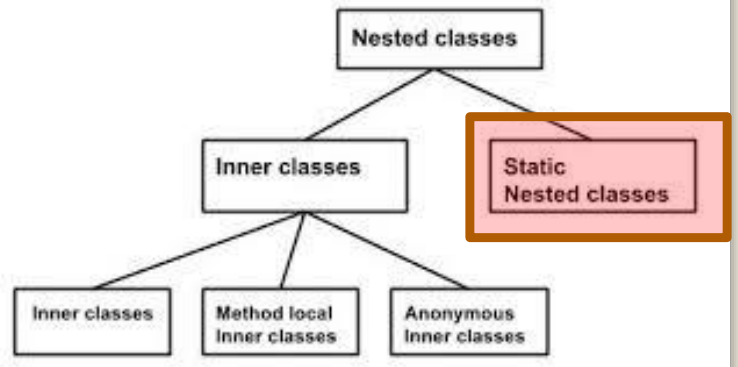
Inner class is a member of the outer class, similar to member variables/ methods – can get us around the multiple inheritance problem of Java

Increases encapsulation

- Variables can be accessed between outer and inner classes (even private);
- Saves alot of data passing between constructors etc

Lead to more readable and maintainable code.

Why use Static Nested Classes?




```
public class OuterClass
{
    private String someVar;

     class static NestedClass
    {
       private anotherVariable

    }

}
```

No special relationship or variable scope between outer and nested classes

Nested class is just like any other outer class – just happens to be nested!

Provides a naming / readability convenience where the Nested class only makes sense in the context of the Outer class