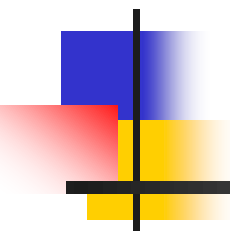


03. Class concepts – (Java)



Class fundamentals
Methods
Constructors
Inner Classes



Static Members

- A static class member can be accessed directly by the class name and doesn't need any object. A single copy of a static member is maintained throughout the program regardless of the number of objects created.
- Static variables are initialized only once and at the start of the execution during the lifetime of a class. These variables will be initialized first before the initialization of any instance variables.



Static Members

Methods declared as static (class methods) have several restrictions:

- They can only call other static methods.
- They must only access static data.
- They cannot refer to *this* or *super* in anyway.
- These methods can be accessed using the class name rather than a object reference.
- *main()* method should be always static because it must be accessible for an application to run, before any instantiation takes place.
- When *main()* begins, no objects are created, so if you have a member data, you must create an object to access it.



Static methods/Data members

```
public class Print {  
    public static String name = "default";  
    public static void printName() {  
        System.out.println(name);  
    }  
    public static void main(String arg[]) {  
        System.out.println(Print.name);  
        Print.printName();  
    }  
}
```



```
class TrackObj
```

```
{
```

```
    //class variable
```

```
    private static int counter = 0;
```

```
    //instance variable
```

```
    private int x = 0;
```

```
    TrackObj()
```

```
{
```

```
        counter++;
```

```
        x ++;
```

```
}
```

```
    //member method
```

```
    public int getX()
```

```
{
```

```
        return x;
```

```
}
```

```
    //class method
```

```
    public static int getCounter()
```

```
{
```

```
        return counter;
```

```
}
```

```
}
```



Access Modifiers

- Java provides a number of access modifiers to set the level of access for classes, fields, methods and constructors.
- A member has package or default accessibility when no accessibility modifier is specified.
- **Access Modifiers:**
 1. private
 2. protected
 3. default
 4. public



private access modifier

- The *private* (most restrictive) access modifier is used for fields or methods and cannot be used for classes and Interfaces.
- It also cannot be used for fields and methods within an interface.
- Field, method declared private are strictly controlled, and that member can be accessed only by other members of that class.
- A standard design strategy is to make all fields private and provide public getter methods for them.



protected access modifier

- Discussed later under the topic - Inheritance



default access modifier

- Java provides a default access modifier which is used when no access modifier is specified.
- Any class, field, method or constructor that has no declared access modifier is accessible only by classes in the same package.
- The default modifier is not used for fields and methods within an interface.



public access modifier

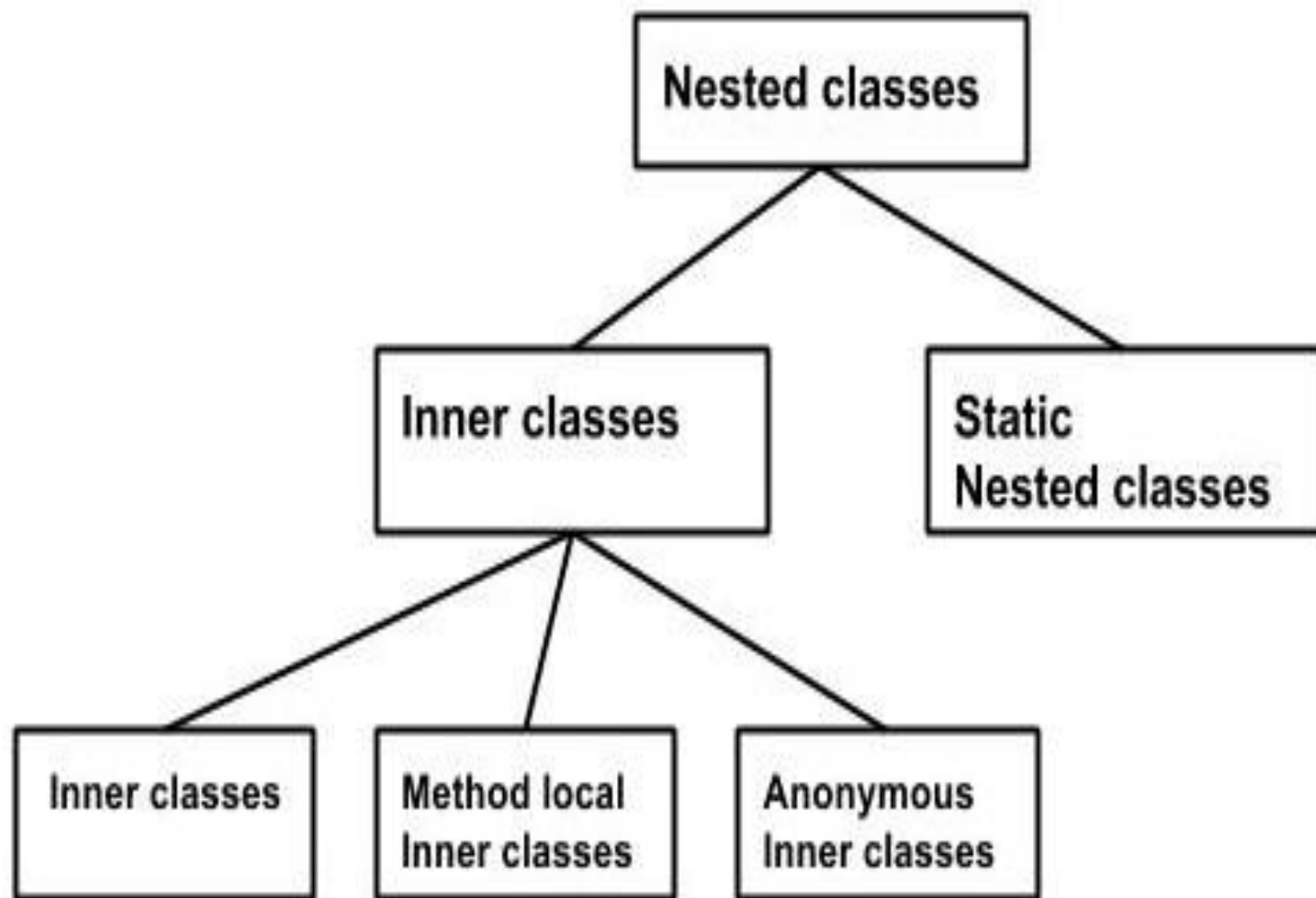
- Fields, methods and constructors declared public (least restrictive) within a public class are visible to any class in the Java program, whether these classes are in the same package or in another package.



Access Levels

The following table shows the access to members permitted by each modifier.

Modifier	Class	Package	Subclass	World
<i>public</i>	Y	Y	Y	Y
<i>protected</i>	Y	Y	Y	N
<i>default</i>	Y	Y	N	N
<i>private</i>	Y	N	N	N





Nested Classes

- The Java programming language allows you to define a class within another class. Such a class is called a *nested class*.

```
class OuterClass {  
    ...  
    class NestedClass {  
        ...  
    }  
}
```



Nested Classes

- Nested classes are divided into two categories: static and non-static. Nested classes that are declared static are simply called *static nested classes*. Non-static nested classes are called *inner classes*



Nested Classes

```
class OuterClass {  
    ...  
    static class StaticNestedClass {  
        ...  
    }  
    class InnerClass {  
        ...  
    }  
}
```



Static Nested Classes

- As with class methods and variables, a static nested class is associated with its outer class. And like static class methods, a static nested class cannot refer directly to instance variables or methods defined in its enclosing class — it can use them only through an object reference.



Inner Classes

- As with instance methods and variables, an inner class is associated with an instance of its enclosing class and has direct access to that object's methods and fields. Also, because an inner class is associated with an instance, it cannot define any static members itself.



Local and Anonymous Inner Classes

- There are two additional types of inner classes. You can declare an inner class within the body of a method. Such a class is known as a *local inner class*. You can also declare an inner class within the body of a method without naming it. These classes are known as *anonymous inner classes*.