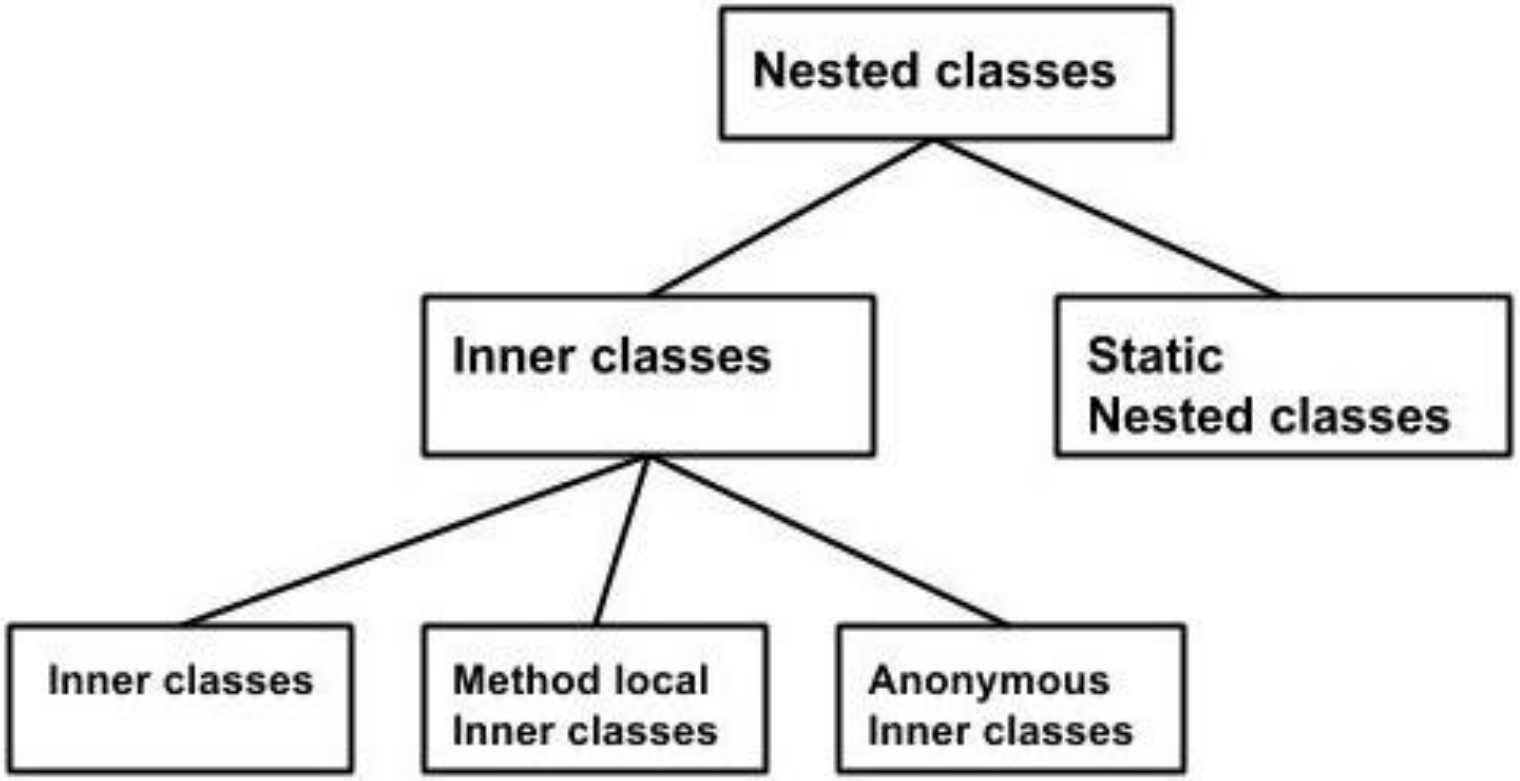
Java & JEE Training

Day 21 – Inner Classes

MindsMapped Consulting

Inner Classes



Inner Classes (Non-static Nested Classes)

```
class Outer_Demo {
   int num;

   // inner class
   private class Inner_Demo {
      public void print() {
         System.out.println("This is an inner class");
      }
   }

   // Accessing he inner class from the method within
   void display_Inner() {
      Inner_Demo inner = new Inner_Demo();
      inner.print();
   }
}

public class My_class {

   public static void main(String args[]) {
      // Instantiating the outer class
      Outer_Demo outer = new Outer_Demo();

      // Accessing the display_Inner() method.
      outer.display_Inner();
   }
}
```

Example: Accessing private members using Inner Class

```
class Outer_Demo {
   // private variable of the outer class
   private int num = 175;

   // inner class
   public class Inner_Demo {
      public int getNum() {
         System.out.println("This is the getnum method of the inner class");
         return num;
      }
   }
}

public class My_class2 {
   public static void main(String args[]) {
      // Instantiating the outer class
      Outer_Demo outer = new Outer_Demo();

      // Instantiating the inner class
      Outer_Demo.Inner_Demo inner = outer.new Inner_Demo();
      System.out.println(inner.getNum());
   }
}
```

Method-local Inner Class

```
public class Outerclass {
  // instance method of the outer class
  void my_Method() {
    int num = 23;

    // method-local inner class
    class MethodInner_Demo {
      public void print() {
        System.out.println("This is method inner class "+num);
      }
    } // end of inner class

    // Accessing the inner class
    MethodInner_Demo inner = new MethodInner_Demo();
    inner.print();
  }

  public static void main(String args[]) {
    Outerclass outer = new Outerclass();
    outer.my_Method();
  }
}
```

Anonymous Inner Class

```
abstract class AnonymousInner {
  public abstract void mymethod();
}

public class Outer_class {

  public static void main(String args[]) {
    AnonymousInner inner = new AnonymousInner() {
      public void mymethod() {
        System.out.println("This is an example of anonymous inner class");
      }
    };
    inner.mymethod();
  }
}
```

Example: Anonymous Inner Class as Argument

```
// interface
interface Message {
   String greet();
}

public class My_class {
   // method which accepts the object of interface Message
   public void displayMessage(Message m) {
      System.out.println(m.greet() +
         ", This is an example of anonymous inner class as an argument");
   }

   public static void main(String args[]) {
      // Instantiating the class
      My_class obj = new My_class();

      // Passing an anonymous inner class as an argument
      obj.displayMessage(new Message() {
         public String greet() {
            return "Hello";
         }
      });
   }
}
```

Static Nested Class

```
public class Outer {
   static class Nested_Demo {
      public void my_method() {
         System.out.println("This is my nested class");
      }
   }

   public static void main(String args[]) {
      Outer.Nested_Demo nested = new Outer.Nested_Demo();
      nested.my_method();
   }
}
```

Why use Nested classes?

- It is a way of **logically grouping** classes that are only used in one place.
- It increases **encapsulation**.
- Nested classes "can" lead to more **readable and maintainable code**.
- Child to parent class connection is simpler as it **visually illustrates** the variables and methods of each class.