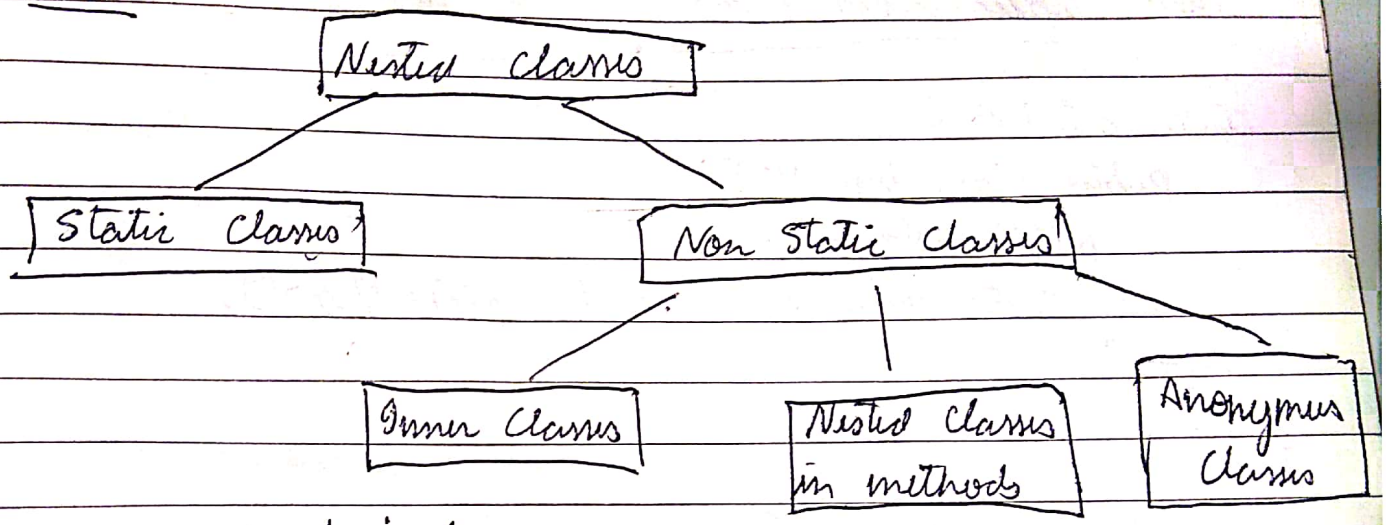
Nested Classes:



Non classes can be private and local to the method in they declared,

Static class:

```
class outerJS {
    public static class staticI {
        public static void show() {
            S.O.P("static class Demo");
        }
    }
}
class innerStaticClassDemo {
    public static void main(String[] args) {
        outerJS.staticI.show();
    }
}
```

byte code of both outer as well as inner class are formed like
outer.class
outer$inner.class

```
private class outer {
    public static void main(String[] arg){
        S.O.P(" Private demo");
    }
}
```

→ This will generate error as outer class is private

non static nested class cannot have static members.

— X —

```
class outer2NS {
    public class innerNS {
        public void show() {
            System.out.println(" Nested class show");
        }
    }
}
class innerClass2Demo {
    public static void main(String[] args) {
        outer2NS objNS = new outer2NS();
        outer2NS.innerNS objInner = objNS.new innerNS();
        objInner.show();
    }
}
```

— X —

★ Private Nested Class :-

```
class outer3Ns {

    Private class innerPri {
        Public void show() {
            SOP(" Private nested class show");
        }
    }

    Public void displayPri() {
        innerPri obj = new innerPri();
        obj.show();
    }
}

class innerPriClass3Demo {
    Public static void main(String[] args) {
        outer3Ns obj = new outer3Ns();
        obj.displayPri();
    }
}
```

* Method local class :

```
class outer4NS {
    public methodClass() {
        class innerclassMethod {
            Public void show() {
                System.out.println("method nested class");
            }
        }
        innerclassMethod obj = new innerclassMethod();
        obj.show();
    }
}
```

Anonymous class → with no physical existance (everything runtime) but still create object.

```
class innerMethodClassDemo {
    Public static void main(String[] args){
        outerNS objNS = new outerNS();
        objNS.methodClass();
    }
}
```

★ Anonymous Class: → by abstract class
→ by interface.

- may contain non-abstract methods, mainly used in eventhandling and action listener.

①
```
abstract class Anni {
    abstract void show();
}
class AnniDemo {
    Public static void main(String[] args) {
        Anni obj = new Anni() {
                    void show() {
                        S.O.P.("Hello from Anonymous class");
                    }
                };
        Obj.show();
    }
}
```

here in this code there are 3 byte codes:

① Anni.class
② AnniDemo.class
③ AnniDemo$1.class → (annonymus class byte code, the next annonymus class will be AnniDemo$2.class and so on.)

override all method or declare default in case of Interface or error will occur.

② interface An2 {
    void show();
    default void demo(){
        S.O.P.(10+6*7);
    }
}
class An2Demo {
    Public static void main(String[] args){
        An2 obj = new An2(){
            Public void show(){
                S.O.P("Demo of An2 using Interface");
            }
        };
        obj.show();
    }
}

---

interface An3 {
    void show();
    void demo();
}
class An3Demo {
    Public static void main(String[] args){
        An3 obj = new An3(){
            Public void show(){
                S.O.P.("Demo of show");
            }
            Public void demo(){
                S.O.P("Demo of demo");
            }
        };
    }

```
        obj.show();
        obj.demo();
    }
}
```

★ Anonymous class as a parameter:

```
interface marks {
    double per(int no);
}
class result {
    void showResult(marks obj) {
        s.o.p.("Percentage: " + obj.per(88));
    }
}
class printResult {
    public static void main(String[] args) {
        result objR = new Result();
        objR.showResult(new marks() {
            public double per(int no) {
                return((((double) m)*100)/100);
            }
        });
    }
}

interface max {
    int max2(int a, int b);
}
```

```
class result2{
    void printmax(max obj){
        S.O.P.(obj, max2(10,25));
    }
}
class printResult2 {
    public static void main(String[] args) {
        result2 objR2 = new result2();
        objR2.printMax(new max(){
            public int max2(int a, int b){
                int c;
                if(a>=b){
                    c=a;
                }else{
                    c=b;
                }
                return c;
            }
        });
    }
}
```

H.W

Q1 → create a anonymous class using abstract class to calculate simple interest.

Q2 → using Interface to calculate compound interest