Java Inner Classes

History:

- At 1.0 v of java Developers found GUI Bugs.
- To remove this error they introduced INNER CLASSES on 1.1v.
- After this inner classes introduced, it have advantages for developers to Write code in simple way.
- Inside a class we can declare another class i.e., Inner Class

Where we use ?

- With out existing one type of object there is no chance of another type of object, then we go for Inner Classes

Example:

Without existing car object there is no Engine object.

Based on position declaration & behaviour:

- Normal / Regular Inner Classes
- Method Local Inner Classes
- Anonymous Inner Classes
- Static Nested Classes

Normal / Regular Inner Classes

- Declaring any named class directly inside class without static modifier such type of classes is called Normal / Regular Inner Classes

```
class outer
{
        class inner
        {
                public void m1()
                {
                        System.out.print("CHIPL");
                }
        }
        public static void main(String args[])
        {
                outer o=new outer();
                outer.inner I=o.new inner();
                I.m1();

                // outer.inner I=new outer().new inner().m1();
        }
}
```

Output:

CHIPL

```
class outer
{
        class inner
        {
                public void m1()
                {
                        System.out.print("CH12");
                }
        }
        public void m2()
        {
                inner I=new inner();
                I.m1();
        }
        public static void main(String args[])
        {
                outer o=new outer();
                o.m2();
        }
}
```

Output:

CH12

```
class outer
{
        class inner
        {
                public void m1()
                {
                        System.out.print("GANESH");
                }
        }
}
class ganesh{
        public static void main(String args[])
        {
                outer.inner I=new outer().new inner().m1();
        }
}
```

Output:

GANESH

```
class outer
{
        int a=10;
        static int b=20;
        class inner
        {
                public void m1()
                {
                        System.out.print(a+" <---> "+b);
                }
        }
        public static void main(String args[])
        {
                outer o=new outer();
                outer.inner I=o.new inner();
                I.m1();

                // outer.inner I=new outer().new inner().m1();
        }
}
```

Output:

10 < --- > 20

```
class outer
{
        int a=10;
        class inner
        {
                int a=99;
                public void m1()
                {
                        int a=988;
                        System.out.println(a);
                        System.out.println(this.a);
                        System.out.print(outer.this.a);
                }
        }
        public static void main(String args[])
        {
                outer o=new outer();
                outer.inner I=o.new inner();
                I.m1();

                // outer.inner I=new outer().new inner().m1();
        }
}
```

Output:

988
99
10

Method Local Inner Classes

- Main purpose is to define method specific repeatedly required functionalit

```
class outer
{
        public void m1()
        {
                class inner
                {
                        public void sum(int x,int y)
                        {
                                System.out.print(x+y);
                        }
                }
        inner I=new inner();
        I.sum(11,22);
        }
        public static void main(String args[])
        {
                outer o=new outer();
                o.m1();
        }
}
```

Output:

33

```
class outer
{
        int a=10;
        static int b=20;
        public static void m1()
        {
                class inner
                {
                        public void m2()
                        {
                                System.out.print(a);
                                System.out.print(b);    // static
                        }
                }
                inner I=new inner();
                I.m2();
        }
        public static void main(String args[])
        {
                outer o=new outer();
                o.m1();
        }
}
```

Output:

Error at a because a is non static used in static method

Anonymous Inner Class:

- Used for Instant Use (One time usage)
- Anonymous class does not contain any Class Name
- Anonymous class extends a class
- Anonymous class extends a Interface
- Advanced Concept is “Lambda expression”

Adding Class for Anonymous class

```
1  class popcorn
2  {
3      public void taste()
4      {
5          System.out.print("Salty");
6      }
7  }
8  class outer
9  {
10     public static void main(String args[])
11     {
12         popcorn p=new popcorn()
13         {
14             public void taste()
15             {
16                 System.out.println("Spicy");
17             }
18         };
19         p.taste();
20         System.out.println(p.getClass().getName());
21
22         popcorn p1=new popcorn();
23         p1.taste();
24         System.out.print(p1.getClass().getName());
25     }
26 }
```

Output:

Spicy
outer$1
Saltypopcorn

Adding interface for Anonymous class

```
interface data{
    public void data();
}
class Anonymous
{
    public static void main(String args[])
    {
        data p=new data()
        {
            public void data()
            {
                System.out.println("Completed");
            }
        };
        p.data();
        System.out.println(p.getClass().getName());
    }
}
```

Output:

Completed
Anonymous$1

Static Nested Classes

- Define nested classes as static modifier is known as Static Nested Classes
- Not Strongly associated with outer class object

```
1  class outer
2  {
3      static class inner
4      {
5          public void m1()
6          {
7              System.out.print("Ongole");
8          }
9      }
10     public static void main(String args[])
11     {
12         inner I=new inner();
13         I.m1();
14     }
15 }
```

Output:

Ongole

- You can also write 2 main methods in this static nested class

```
1  class outer
2  {
3      static class inner
4      {
5          public static void main(String args[])
6          {
7              System.out.print("CH12");
8          }
9      }
10     public static void main(String args[])
11     {
12         System.out.print("Ganesh");
13     }
14 }
```

Output Console:

Ganesh

Output Cmd:

javac outer.java

java outer --- > Ganesh

java outer$inner --- > CH12


Thank
You