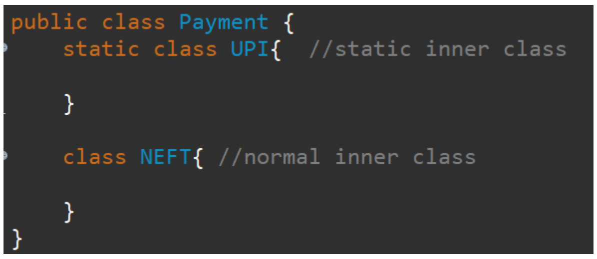
INNER CLASS

Type 1: Normal Inner class

Type 2: Static Inner Class

Example:



```
public class Payment {
    static class UPI{  //static inner class

    }

    class NEFT{ //normal inner class

    }
}
```

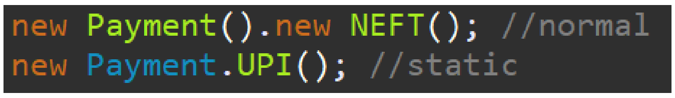
Creating Objects of Inner classes

Normal class:

<obj of outer class> . <obj of inner-class>

Static class:

<Outer-class-name.obj of inner class>

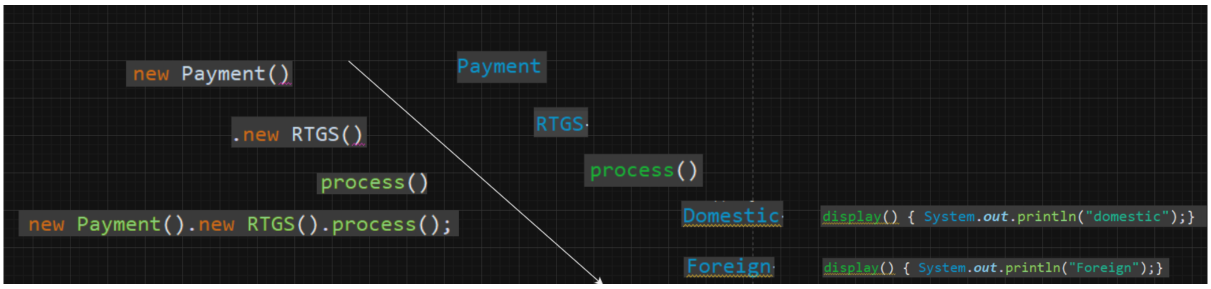


```
new Payment().new NEFT(); //normal
new Payment.UPI(); //static
```

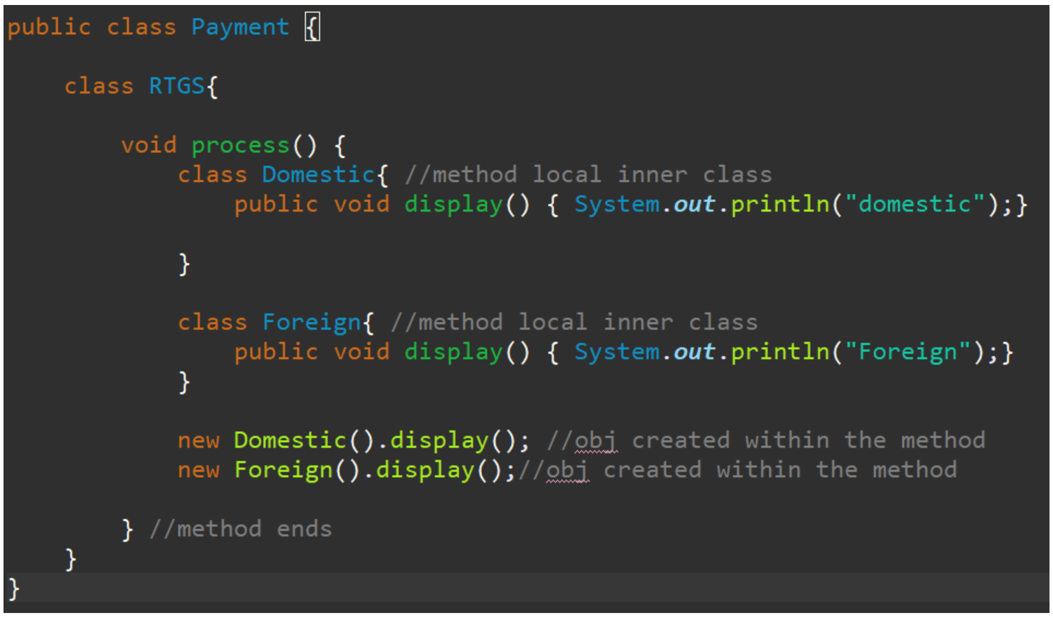
Conclusion: We need not create an object of Outer class to access inner class if we mark inner-class as static.

Type 3: Method local inner class

These classes are created inside the method which can be inside other inner class as shown below

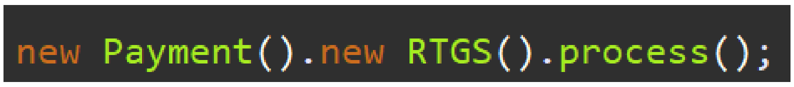


```
new Payment().new RTGS().process();
display() { System.out.println("domestic");}
display() { System.out.println("Foreign");}
```



```
public class Payment {

    class RTGS{

        void process() {
            class Domestic{ //method local inner class
                public void display() { System.out.println("domestic");}

            }

            class Foreign{ //method local inner class
                public void display() { System.out.println("Foreign");}
            }

            new Domestic().display(); //obj created within the method
            new Foreign().display();//obj created within the method

        } //method ends
    }
}
```

Calling from controller:



```
new Payment().new RTGS().process();
```

Type 4: Anonymous Inner class

If you have an interface and want to override its methods without using a separate class then you can do so as shown below:

```
public interface Payroll {
    public void processSalary(int empId);
}
```

```
Payroll payroll = new Payroll() { //Anonymous inner class

  @Override
  public void processSalary(int empId) {
       System.out.println("Salary calc for " + empId);

  }

};

payroll.processSalary(1);
```

That’s it for Type 4 class.

Note: We have used this class format for RowMapper interface in Spring JDBC implementation.