PRAKTIKUM 14

NESTED CLASS

A. TUJUAN PEMBELAJARAN

1. Mampu mengimplementasikan konsep inner class

B. DASAR TEORI

Java membolehkan menyisipkan suatu kelas ke dalam kelas lainnya. Kelas sisipan ini disebut kelas Inner. Kelas Inner berguna untuk mendukung suatu proses yang akan dijalankan oleh kelas luarnya. Beberapa ketentuan kelas Inner :

- Kelas Luar yang mengandung kelas Inner, bila dikompilasi akan menghasilkan dua file *.class, yaitu *Luar.class* dan *Luar$Inner.class*
- Kelas Inner boleh tidak diberi nama, yang disebut Anonymous Inner.
- Kelas Inner dapat diberi modifier akses public, atau protected, atau default, ataupun private.
- Untuk mengakses referensi this dari kelas luar digunakan bentuk NamaKelasLuar.this.
- Kelas Luar ikut bertanggung-jawab dalam instansiasi kelas Inner (yang non static) . Kalau objek kelas Luar adalah a, dan objek kelas Inner adalah b, maka sintaks yang benar adalah :

  ```
  Luar a = new Luar();
  Luar.Inner b = a.new Inner();
  ```

  - Jika kelas Inner bersifat static, maka objek milik kelas Inner dapat dibuat sendiri tanpa melalui kelas Luarnya, (Artinya kelas Inner tidak dapat mengakses attribute ataupun method non static milik kelas Luarnya).

Inner Class adalah kelas yang disisipkan di dalam kelas yang lain. Fungsi kelas sisipan ini adalah mendukung suatu proses yang akan dijalankan oleh kelas utamanya. Inner Class bersifat

tersarang terhadap kelas – kelas utamanya, seperti halnya blok penyeleksian (if, for) yang tersarang pada blok penyeleksian lainnya atau method yang tersarang pada method lainnya.

Analogi Inner Class

Inner Class dapat dianalogikan sebagi hubungan antara manusia dan paru – paru. Setiap manusia pasti bernafas dengan menggunakan paru – paru. Dalam hal ini berarti kinerja dari paru – paru turut mendukung/menetukan kinerja dari manusia. Dalam bahasa pemrograman berorientasi objek manusia dapat dikatakan sebagai outer class dan paru – paru sebagai inner class-nya.

C. TUGAS PENDAHULUAN

Jelaskan kembali pengertian inner class menurut bahasa anda!

D. PERCOBAAN

Percobaan 1 : Non-Static Inner class yang dideklarasikan di dalam class

```
public class Luar{
    private String variabelLuar = "Variabel Luar";

    class Dalam{
        String variabelDalam = "Variabel Dalam";

        public void bicara(){
            System.out.println(variabelDalam);
            System.out.println(variabelLuar);
        }
    }
}

class TestDalam{
    public static void main(String args[]){
        Luar l = new Luar();
        Luar.Dalam d = l.new Dalam();
        d.bicara();
    }
}
```

Percobaan 2 : Inner class yang dideklarasikan di dalam method.

```
public class MOuter{
    private int m = (int)(Math.random()*100);

    public static void main(String args[]){
        MOuter that = new MOuter();
        that.go((int)(Math.random()*100),(int)(Math.random()*100));
    }

    public void go(int x, final int y){
        int a = x+y;
        final int b = x-y;
        class MInner{
            public void method(){
                System.out.println("Nilai m adalah: " + m);
                System.out.println("Nilai x adalah: " + x);
                System.out.println("Nilai y adalah: " + y);
                System.out.println("Nilai a adalah: " + a);
                System.out.println("Nilai b adalah: " + b);
            }
        }

        MInner that = new MInner();
        that.method();
    }
}
```

Percobaan 3 : Static Inner class. Perbaiki error yang terjadi.

```
class TestStaticInnerClass1 {

    static String test = "Outer class static field";
    String instFld = "This is an instance field";

    public static void main(String[] args) {
        System.out.println(Inner.value);
        new Inner();
    }

    static class Inner {

        static int value = 100;

        Inner() {
            System.out.println("New static inner class");
            System.out.println(test);
            System.out.println(instFld);
            // can create an instance of the outer class and then
            // access instance fields
            TestStaticInnerClass tsi = new TestStaticInnerClass();
            System.out.println(tsi.instFld);
        }
    }
}
```

Percobaan 4 : Non Static Inner class yang dideklarasikan dalam class

```
class Outer {
    static String test;
    String str = "Outer class field";

    Outer() {
        new Inner();
    }

    class Inner {            // class defined within the body of Outer
//      static String str;          // compile-error
//      static final String str;    // compile-error
        static final String str = "Constant is ok";
//      static {}           // static initializer not allowed
        Inner() {
            System.out.println(Outer.this.str);
            test = "Inherits static member";
            System.out.println(test);
        }
//      interface NeverInner();     // compile-error
    }

    public class PublicInner{}
    protected class ProtectedInner {}
    private class PrivateInner{}
    abstract class AbstractInner {}
    final class FinalInner {}
    static class StaticInner {}
}
```

Percobaan 5 : Local Inner class

```
class TestLocalInner {
    public static void main(String[] args) {
        Outer o = new Outer();
        o.display();
//      new Outer.new Local();   // doesn't compile
    }
}

class Outer {
    static String classFld = "static class members are accessible";
    String instFld = "instance flds of enclosing class are accessible";

    void display(){
        final String str = "only final method variables are accessible";

        class Local {
            Local() {
                System.out.println(str);
                System.out.println(classFld);
                System.out.println(instFld);
            }
        }

        // can only be created within the enclosing block
        new Local();
    }
    // this won't work
//    new Local();
}
```

Percobaan 6 : Anonymous class

```
class TestAnonymous{
   public static void main(String str[]){
      final int d = 10;
      father f = new father(d);
      father fanon = new father(d){

         // override method parent
         void method (int x){
            System.out.println("Anonymous : " + x);
         }

         // overload method parent
         void method (String str){
            System.out.println("Anonymous : " + str);
         }

         // penambahan method baru di anonymous
         void newMethod(){
            System.out.println("New Method in Anonymous : ");
         }
      };

      //fanon.method("New Number"); // ini akan error
      //fanon.newMethod(); // ini akan error
   }
}
```

Percobaan 7 : Inner class yang dideklarasikan Inner class yang dideklarasikan di di dalam dalam method.

```
public class Parcel1{
    public Tujuan ke(String s){
        class Tujuannya implements Tujuan{
            private String label;

            Tujuannya(String tuj){
                label = tuj;
            }

            public String bacaLabel(){
                return label;
            }
        }
        return new Tujuannya(s);
    }

    public static void main(String args[]){
        Parcel1 p1 = new Parcel1();
        Tujuan t = p1.ke("Bali");
        System.out.println(t.bacaLabel());
    }
}

interface Tujuan{
    String bacaLabel();
}
```

Percobaan 8 : Anonymous Inner Class.

```
import java.awt.*;
import java.awt.event.*;

public class X extends Frame{
   public static void main(String arg[]){
      X x = new X();
      x.pack();
      x.setVisible(true);
   }

   private int count;
   public X(){
      final Label l = new Label("Count = " + count);
      add(l,BorderLayout.SOUTH);
      add(
         new Button("Hello " + 1){
            {
               addActionListener(
                  new ActionListener(){
                     public void actionPerformed(ActionEvent ev) {
                        count++;
                        l.setText("Count = " + count);
                     }
                  }
               );
            }
         }, BorderLayout.NORTH );
   }
}
```

E. LATIHAN

Latihan 1 : Program berikut tidak dapat dikompile. Perbaiaki agar dapat dikompile

```
public class Problem {
        String s;
        static class Inner {
                void testMethod() {
                   s = "Set from Inner";
                }
        }
}
```

Latihan 2 : Gunakan Java API documentation untuk Box class (di javax.swing package) untuk menjawab pertanyaan berikut :

1. What static nested class does `Box` define?
2. What inner class does `Box` define?
3. What is the superclass of `Box`'s inner class?
4. Which of `Box`'s nested classes can you use from any class?

5. How do you create an instance of `Box`'s `Filler` class?

F. TUGAS

Jelaskan kelebihan dan kelemahan penggunaan inner kelas!

G. LAPORAN RESMI

Kumpulkan hasil percobaan di atas dan tambahkan analisa untuk tiap percobaan, latihan, dan tugas yang telah dibuat.