**Praktikum Nested Classes**

**PBO**

**Kelas A**

****

Oleh :

Adjie Firmansyah

4523210004

Dosen :

**Adi Wahyu Pribadi, S.Si., M.Kom**

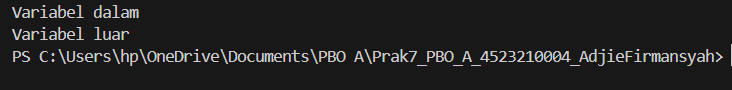
**S1 Teknik Informatika**

**Universitas Pancasila**

Latihan 1

package app;  
public class Luar {  
 private String outerVariable = "Variabel luar";  
  
 class Dalam {  
 String innerVariable = "Variabel dalam";  
  
 public void bicara() {  
 System.out.println(innerVariable);  
 System.out.println(outerVariable);  
 }  
 }  
}  
public class App {  
 public static void main(String[] args) throws Exception {  
 Luar l = new Luar();  
 Luar.Dalam d = l.new Dalam();  
  
 d.bicara();  
 }  
}

Output



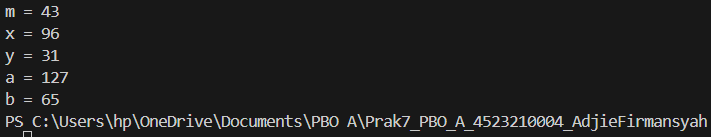
Penjelasan Output

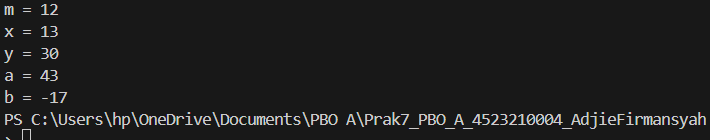
Inner class (Dalam) dapat mengakses variabelnya sendiri (innerVariable), sehingga mencetak "Variabel dalam". Inner class juga dapat mengakses variabel dari outer class (Luar) meskipun bersifat private (outerVariable), sehingga mencetak "Variabel luar".

Latihan 2

package app;  
public class Mouter {  
 private int m = (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("m = " + m);  
 System.out.println("x = " + x);  
 System.out.println("y = " + y);  
 System.out.println("a = " + a);  
 System.out.println("b = " + b);  
 }  
 }  
  
 MInner that = new MInner();  
 that.method();  
 }  
   
 public static void main(String[] args) throws Exception {  
 MOuter that = new MOuter();  
 that.go((int)(Math.random()\*100), (int)(Math.random()\*100));  
 }  
}

Output





Penjelasan Output

Outputnya mencetak m, x, y, a, dan b karena **local inner class** dapat mengakses variabel instance (m) dari outer class dan variabel lokal (x, y, a, b) yang bersifat final. Nilai m, x, dan y acak, jadi output bervariasi setiap kali dijalankan.

Latihan 3

Kelebihan Inner Class:

1. Enkapsulasi: Mengelompokkan logika terkait dalam satu tempat, menjaga kode tetap terstruktur.
2. Akses Praktis: Dapat langsung mengakses variabel dan metode outer class, termasuk yang private.
3. Modularitas Tinggi: Membantu dalam membangun struktur kode yang lebih kompleks dan modular.
4. Event Handling: Memudahkan penerapan callback atau listener dalam pemrograman berbasis event.

Kelemahan Inner Class:

1. Kode Lebih Rumit: Penggunaan inner class yang berlebihan dapat membuat kode menjadi lebih sulit dibaca dan dikelola.
2. Penggunaan Memori Lebih Besar: Menyimpan referensi ke outer class dapat meningkatkan penggunaan memori.
3. Pelanggaran Enkapsulasi: Kemudahan akses ke anggota private bisa mengurangi isolasi antar komponen.

Latihan 4

