

**PRAKTIKUM PEMROGRAMAN BERORIENTASI
OBJEK**

untuk memenuhi tugas praktikum ke-7



Dibuat oleh :

Afandi Ikhsyan Al Karim (4522210032)

Kelas A

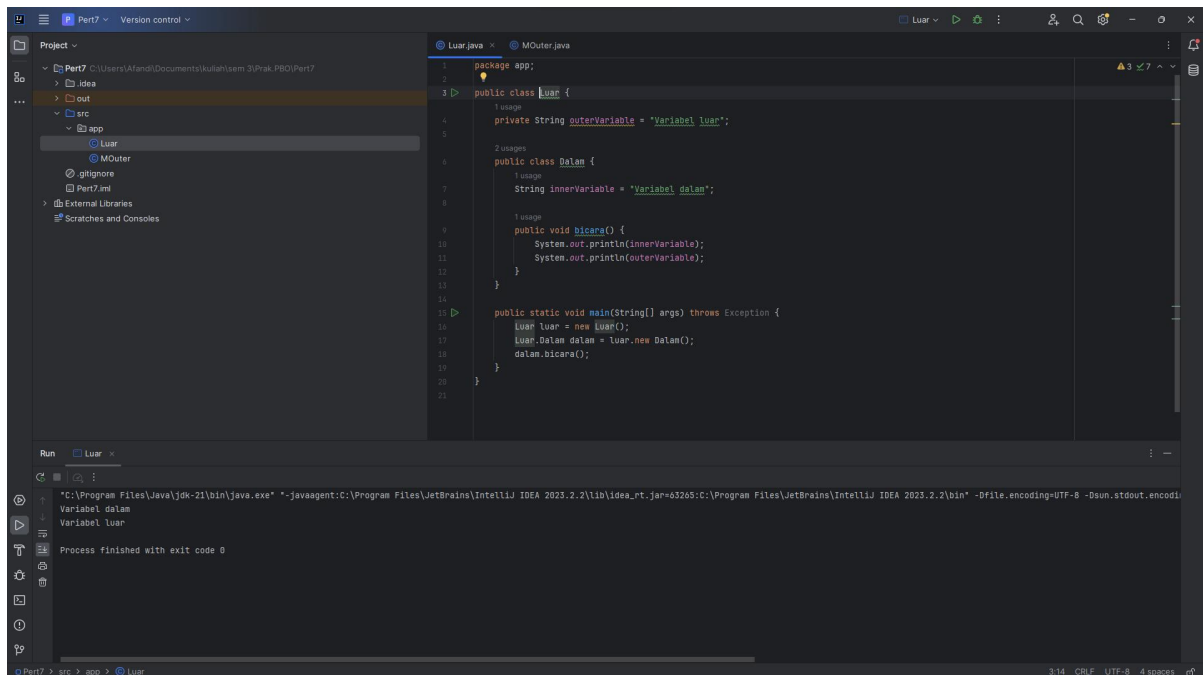
Dosen :

ADI WAHYU PRIBADI, S.Si., M.Kom

S-1 Teknik Informatika Universitas Pancasila

2023/2024

1. LATIHAN 1



```
1 package app;
2
3 public class Luar {
4     private String outerVariable = "Variabel luar";
5
6     public class Dalam {
7         String innerVariable = "Variabel dalam";
8
9         public void bicara() {
10             System.out.println(innerVariable);
11             System.out.println(outerVariable);
12         }
13     }
14
15     public static void main(String[] args) throws Exception {
16         Luar luar = new Luar();
17         Luar.Dalam dalam = luar.new Dalam();
18         dalam.bicara();
19     }
20 }
21 }
```

Run

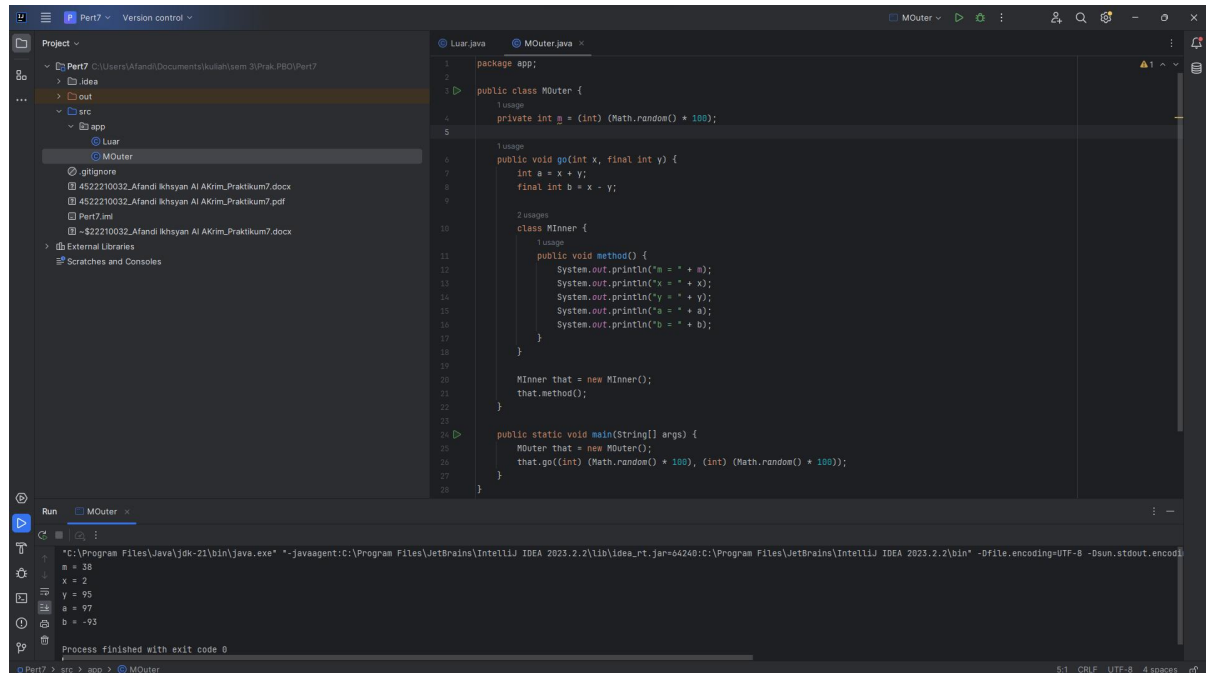
"C:\Program Files\Java\jdk-21\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2023.2.2\lib\idea_rt.jar=63265:C:\Program Files\JetBrains\IntelliJ IDEA 2023.2.2\bin" -Dfile.encoding=UTF-8 -Dsun.stdout.encoding=UTF-8

Variabel dalam
Variabel luar

Process finished with exit code 0

- a) Kelas Luar (Outer Clas):
 - Ini adalah kelas utama dalam kode. Itu memiliki satu variabel anggota `outerVariable`, yang merupakan variabel `String` yang berisi “Variabel luar”.
 - Kelas ini juga memiliki inner class yaitu `Dalam`.
- b) Kelas Dalam (Inner Class):
 - Ini adalah inner class yang dideklarasikan di dalam kelas luar. Inner class memiliki akses ke semua anggota kelas luar, termasuk variabel `outerVariable`.
 - Kelas ini memiliki satu variabel anggota `innerVariable`, yang merupakan variabel `String` yang berisi “Variabel dalam”.
 - Ada metode `bicara()` yang mencetak nilai dari `innerVariable` dan `outerVariable`. Karena `Dalam` adalah inner class, ia dapat mengakses `outerVariable` dari kelas luar.
- c) Metode main:
 - Metode main digunakan untuk memulai aplikasi.
 - Pertama, membuat objek `Luar` dengan nama `luar`.
 - Kemudian, membuat objek `Dalam` dengan nama `dalam` menggunakan `luar.new Dalam()`. Ini adalah cara untuk membuat objek inner class.
 - Akhirnya, memanggil metode `bicara()` pada objek `dalam`, yang mencetak nilai `innerVariable` dan `outerVariable`.

2. LATIHAN 2



a) Kelas MOuter:

- Ini adalah kelas utama.
- Kelas ini memiliki satu variabel anggota 'm', yang diinisialisasi dengan nilai acak antara 0 hingga 99 menggunakan 'Math.random() * 100'. Variabel ini adalah milik kelas 'MOuter'

b) Method 'go(int x, final int y)':

- Method ini menerima dua parameter 'x' dan 'y'.
- Di dalam method ini, ada dua variabel lokal: 'a' dan 'b'. Variabel 'a' diinisialisasi dengan penjumlahan 'x + y', sementara variabel 'b' diinisialisasi dengan pengurangan 'x - y'.
- Kemudian, ada deklarasi lokal inner class bernama 'MInner' yang memiliki satu metode, yaitu 'method()'.
- Dalam metode 'method()', mencetak nilai dari beberapa variabel:
 - 'm': ini adalah variabel anggota dari kelas 'MOuter' dan dapat diakses dari inner class.
 - 'x' dan 'y': ini adalah parameter method 'go()' yang juga dapat diakses dari inner class.
 - 'a' dan 'b': ini adalah variabel lokal dari method 'go()' dan dapat diakses dari inner class.

c) Method 'main(String[] args)':

- Method 'main' digunakan untuk memulai aplikasi.
- Pertama, membuat objek 'MOuter' dengan nama 'that'.
- Kemudian, memanggil method 'go()' pada objek 'that' dan memberikan dua nilai acak sebagai argumen. Ini akan menciptakan objek 'MInner' lokal dan memanggil method 'method()' pada objek tersebut.

3. LATIHAN 3

Jelaskan kelebihan dan kelemahan Inner Class!

Kelebihan inner class:

1. Privasi: Inner class dapat dideklarasikan sebagai private, yang berarti mereka hanya dapat diakses dari dalam kelas yang mengelilinginya.
2. Konteks Terkait :Inner class sering digunakan untuk menghubungkan inner class dengan outer class, sehingga kelas tersebut memiliki konteks yang berhubungan dengan kelas utamanya.
3. Abstraksi: Inner class membantu dalam menyembunyikan kompleksitas, sehingga dapat mengorganisasi kode dengan lebih baik.

Kelemahan Inner Class:

1. Kode Rumit: Terlalu banyak inner class dapat membuat kode menjadi lebih rumit dan sulit dibaca.
2. Ketergantungan Tinggi: Inner class sangat bergantung pada kelas utama, sehingga perubahan pada kelas utama dapat memengaruhi inner class.
3. Pembuatan Objek Sulit: Membuat objek inner class memerlukan objek kelas utama terlebih dahulu, yang bisa membuat proses pembuatan objek lebih rumit.