**LAPORAN PRAKTIKUM PEMROGRAMAN BEORIENTASI OBJEK**

“Tugas 07 Nested Class*”*

****

Oleh:

Nama : Mohammad Sayifullah

NPM : 4523210066

Kelas : A

Dosen:

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

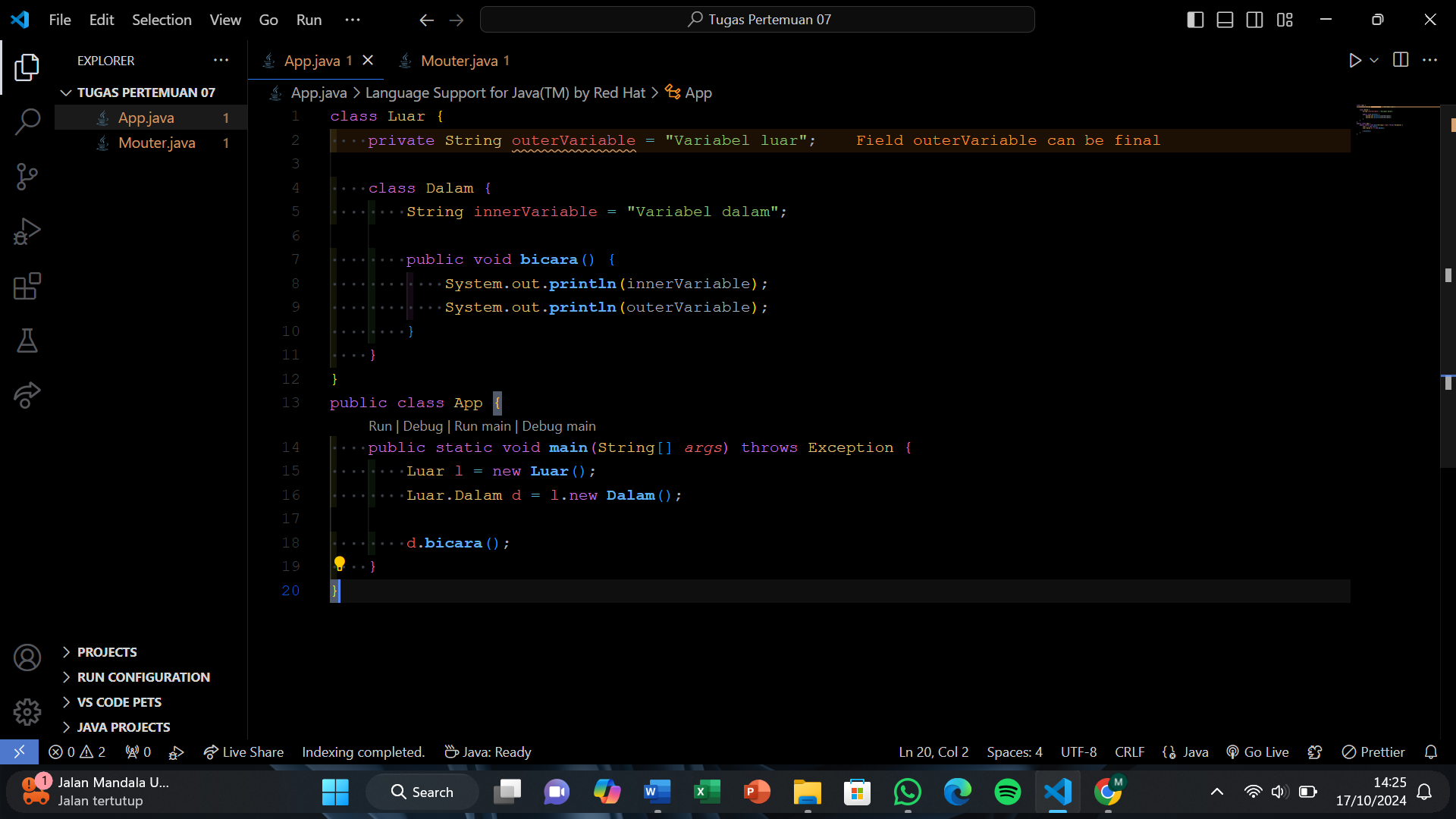
**S1-Teknik Informatika**

**Fakultas Teknik Universitas Pancasila**

**2024**

**LATIHAN 1**

Source Code App.java

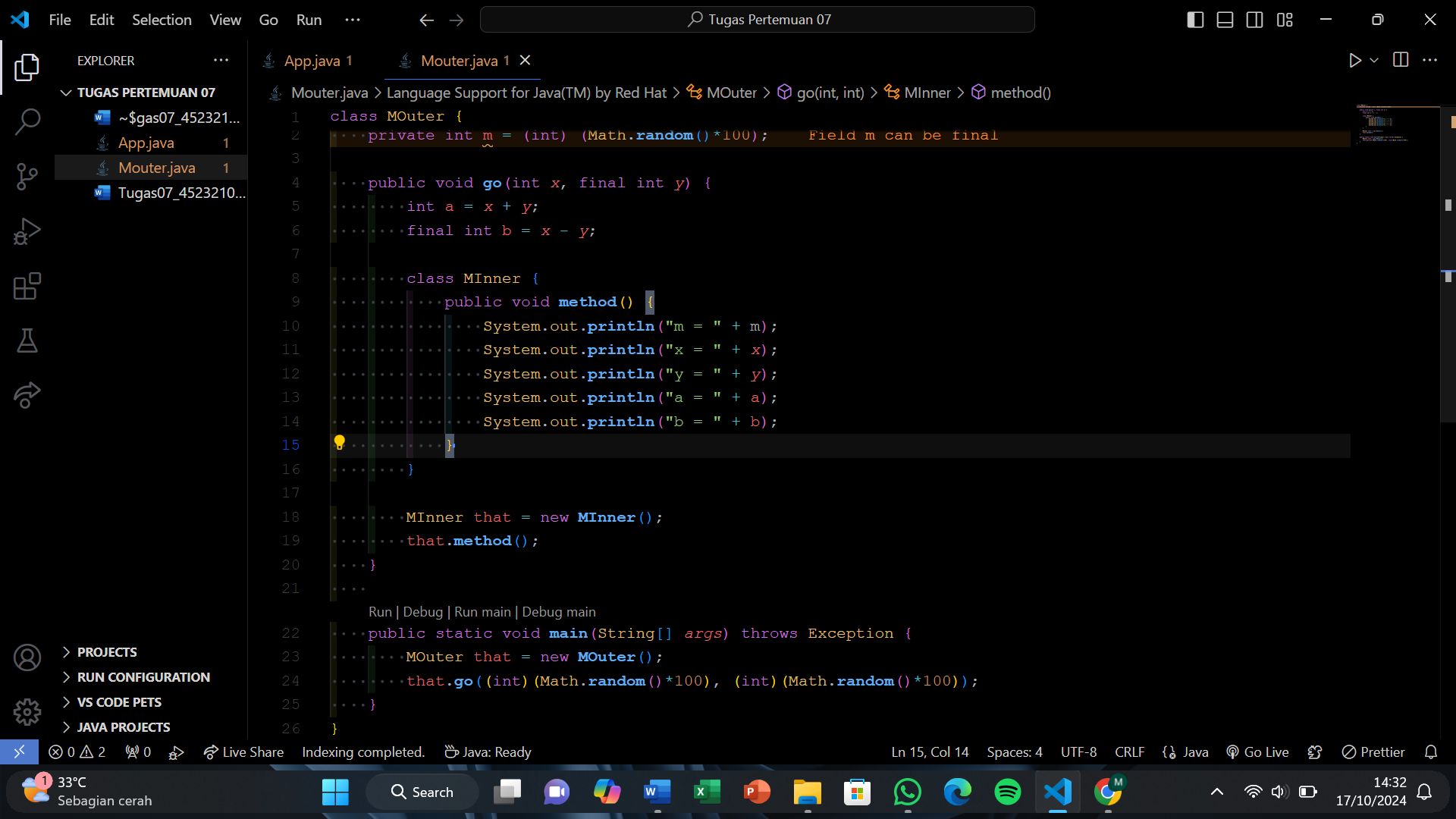


**Penjelasan:**

kelas luar memiliki varibael instance yang bersifat private dengan nilai “variabel luar” didalam kelas luar ada kelas dalam yang memiliki variabel dengan nilai “variabel dalam” dan juga kelas dalam juga memiliki metode bicara yang mencetak nilai innervariable dan outervariable, kemudian di kelas App didalam main method objek luar dan objek dalam dibuat kemudian method bicara dari objek dipanggil.

**LATIHAN 2**

Source Code Mouter.java



**Penjelasan:**

**MOuter** adalah outer class. Di dalam kelas ini memiliki variabel m yang diinisialisasikan dengan nilai acak 0 – 100, **go** adalah metode dari kelas **MOuter** yang menerima dua variabel: x dan y. **y** ditandai sebagai **final**, artinya nilainya tidak dapat diubah di dalam metode. **a**menyimpan hasil penjumlahan x + y. **b** dideklarasikan sebagai **final** dan menyimpan hasil pengurangan x - y. Variabel **b** harus **final** karena digunakan di dalam local inner class. **MInner adalah** local inner class, yaitu kelas yang dideklarasikan di dalam metode go. **MInner** memiliki satu metode: method(), yang mencetak nilai variabel m, x, y, a, dan b Setelah mendeklarasikan kelas **MInner**, kita membuat objek **MInner** bernama **that** dan memanggil metode **method()** untuk mencetak semua nilai yang disebutkan di atas. Di dalam metode main, membuat objek MOuter yang Bernama that. Metode go kemudian dipanggil dengan dua nilai acak sebagai variabel untuk x dan y.

**LATIHAN 3**

* **Kelebihan Inner Class**

1. Organisasi kode lebih baik: memungkinkan untuk mengelompokkan kelas yang terkait secara logis yang membantu dalam menjaga ketertarikan antara dua kelas yang sering digunakan bersama dalam satu file.
2. Modularitas: memungkinkan untuk menyembunyikan kelas yang tidak perlu dilihat atau digunakan oleh kelas lain diluar konteks penggunaannya.
3. Enkapsulation: implementasi detail yang lebih tersembunyi karena hanya dapat digunakan dalam konteks kelas luar.

* **Kekurangan Inner Class**

1. **Kompleksitas kode**: Penggunaan inner class, terutama jika terlalu banyak, dapat membuat kode menjadi lebih sulit dibaca dan dipelihara. Program yang kompleks dengan banyak inner class bisa membingungkan pengembang lain.
2. **Keterikatan kuat (tight coupling):** Inner class memiliki keterikatan kuat dengan kelas luar. Hal ini bisa membuat perubahan pada kelas luar memengaruhi inner class dan sebaliknya, sehingga berpotensi mengurangi fleksibilitas kode.

**LATIHAN 4**

Hasil Running App.java dan Mouter.java

