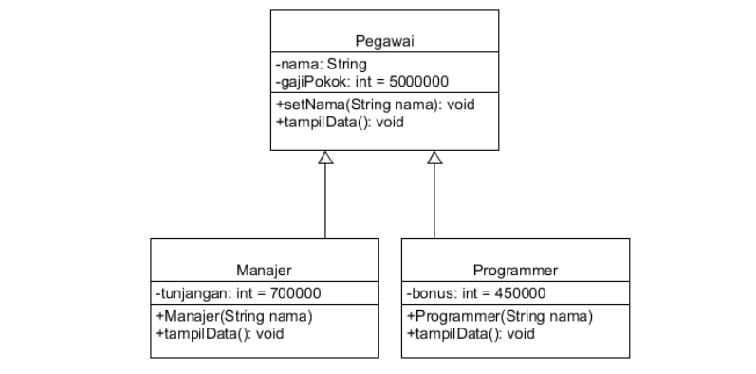
**Praktikum 6**

**Polimorfisme**

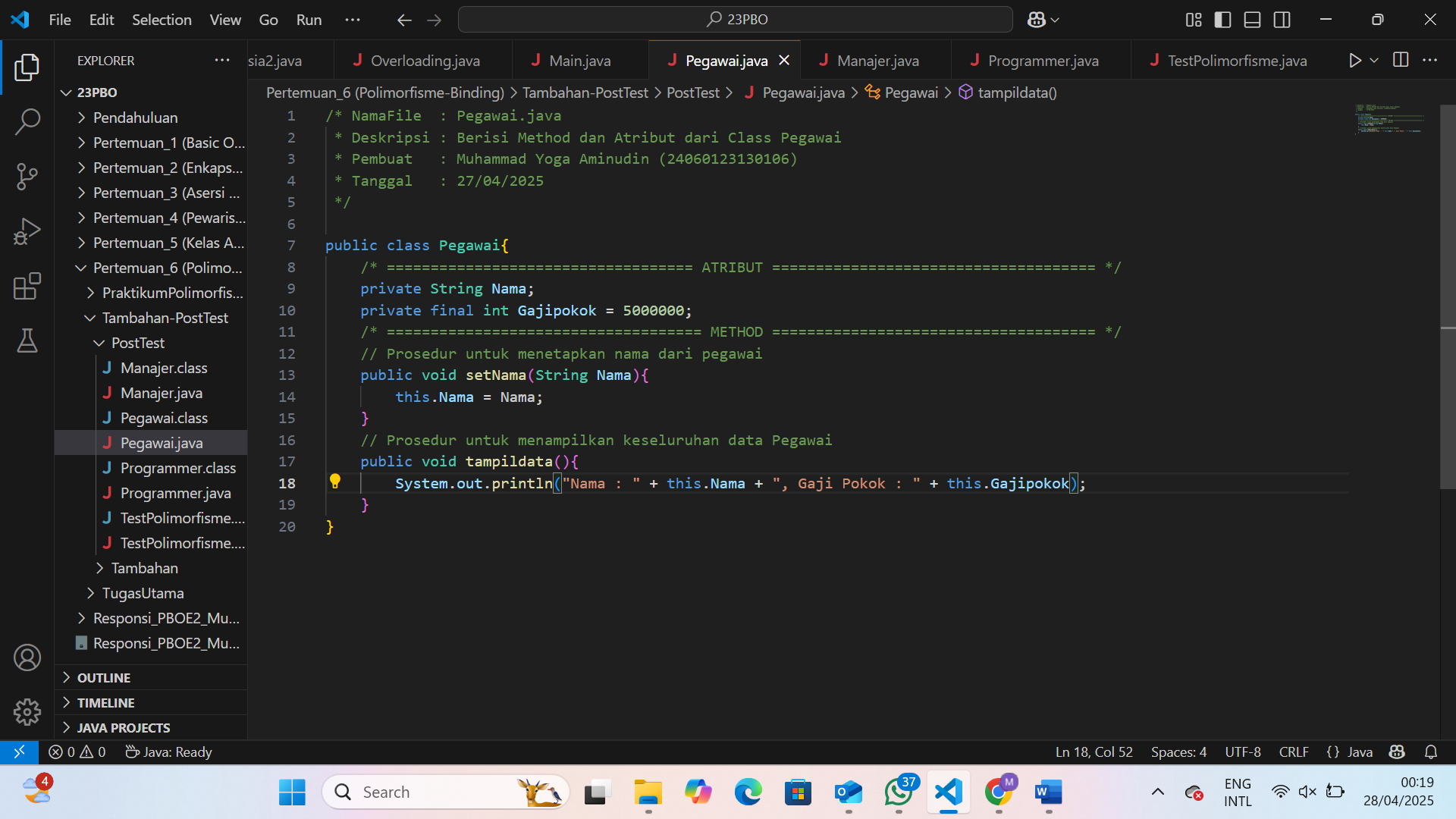


Catat hasil Post Test pada file doc:

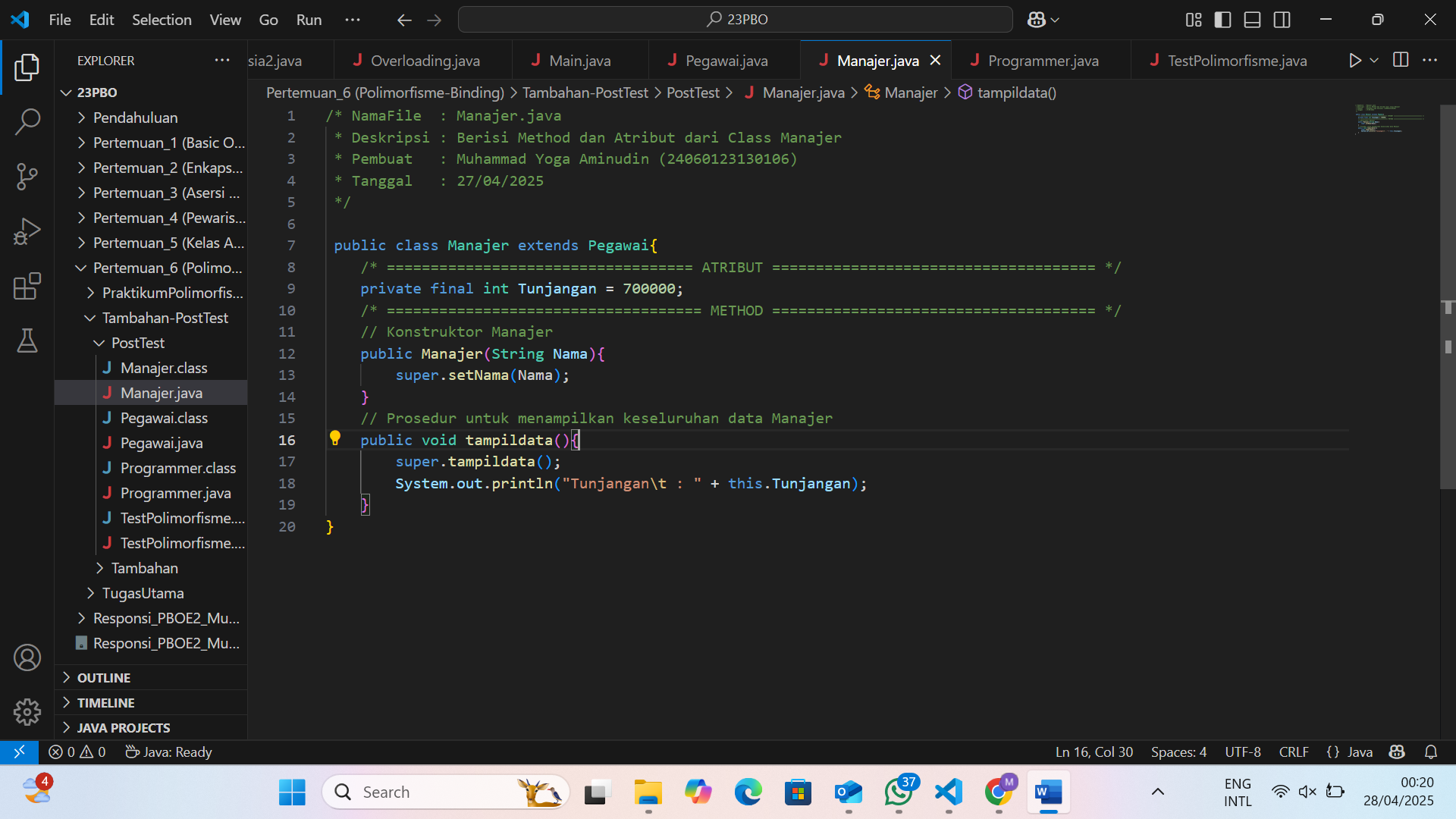
1. Screenshoot kode yang dibuat, dan screenshoot outputnya.

Hasil Kode Program

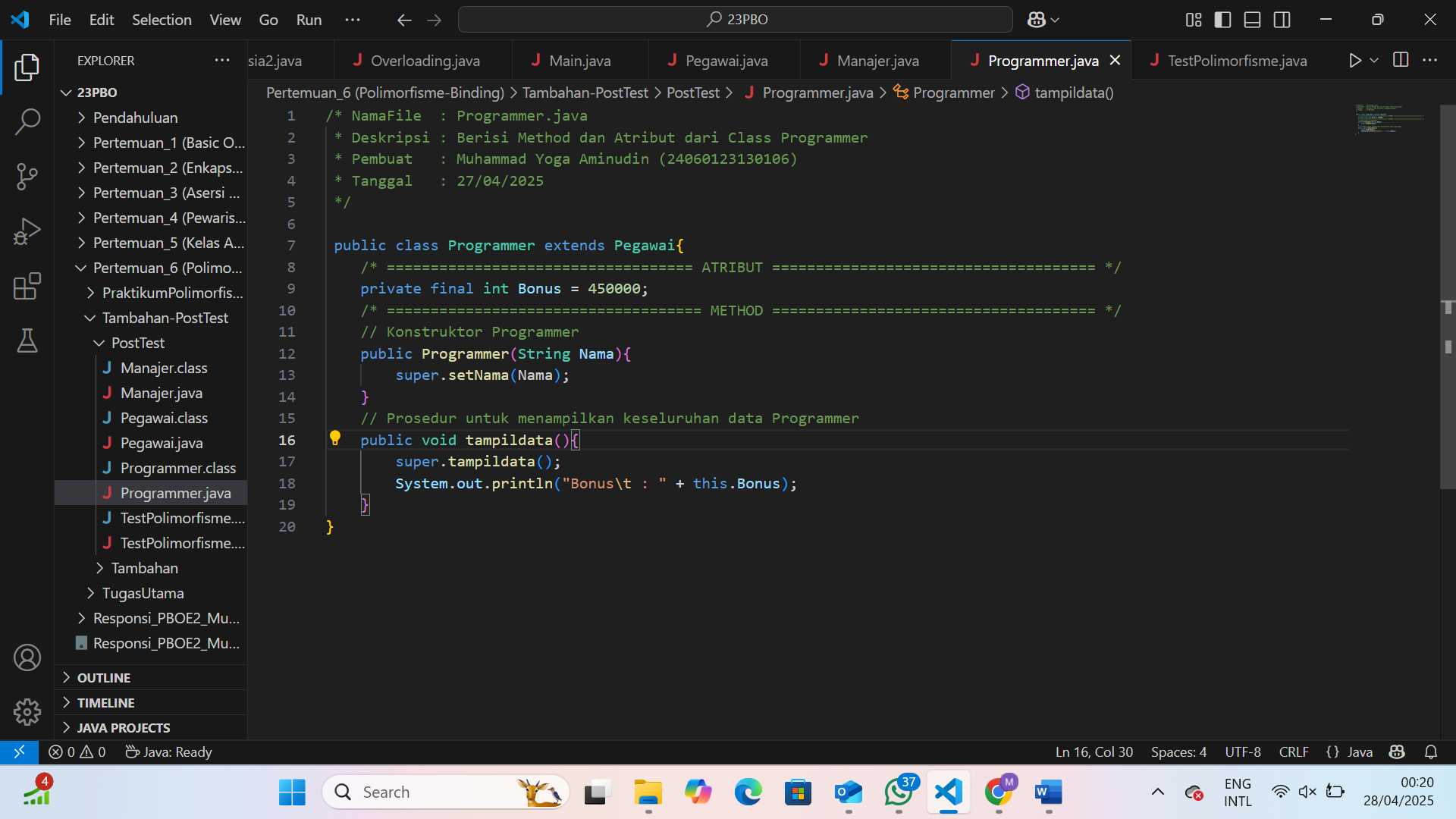
Pegawai.java :



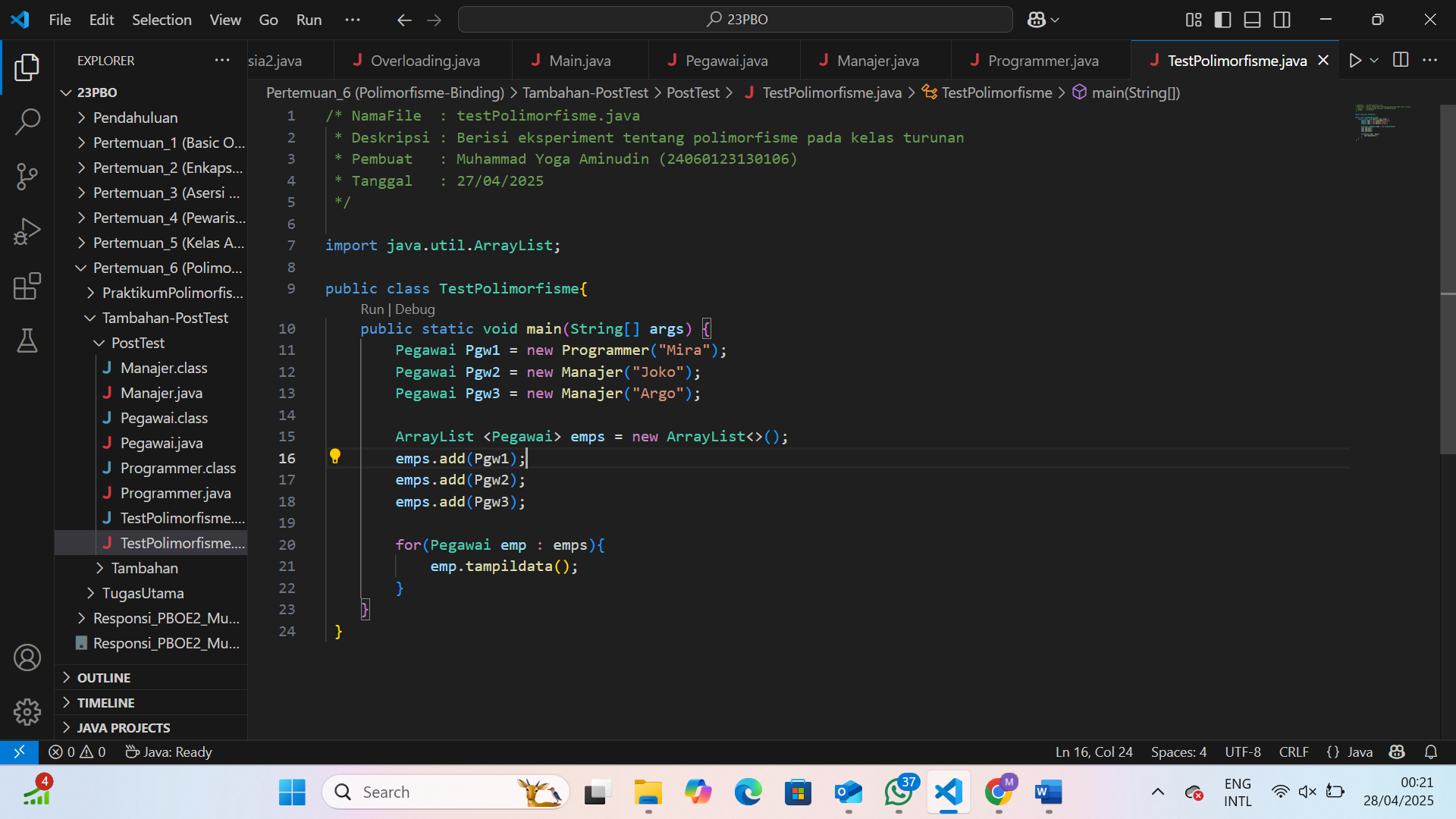
Manajer.java



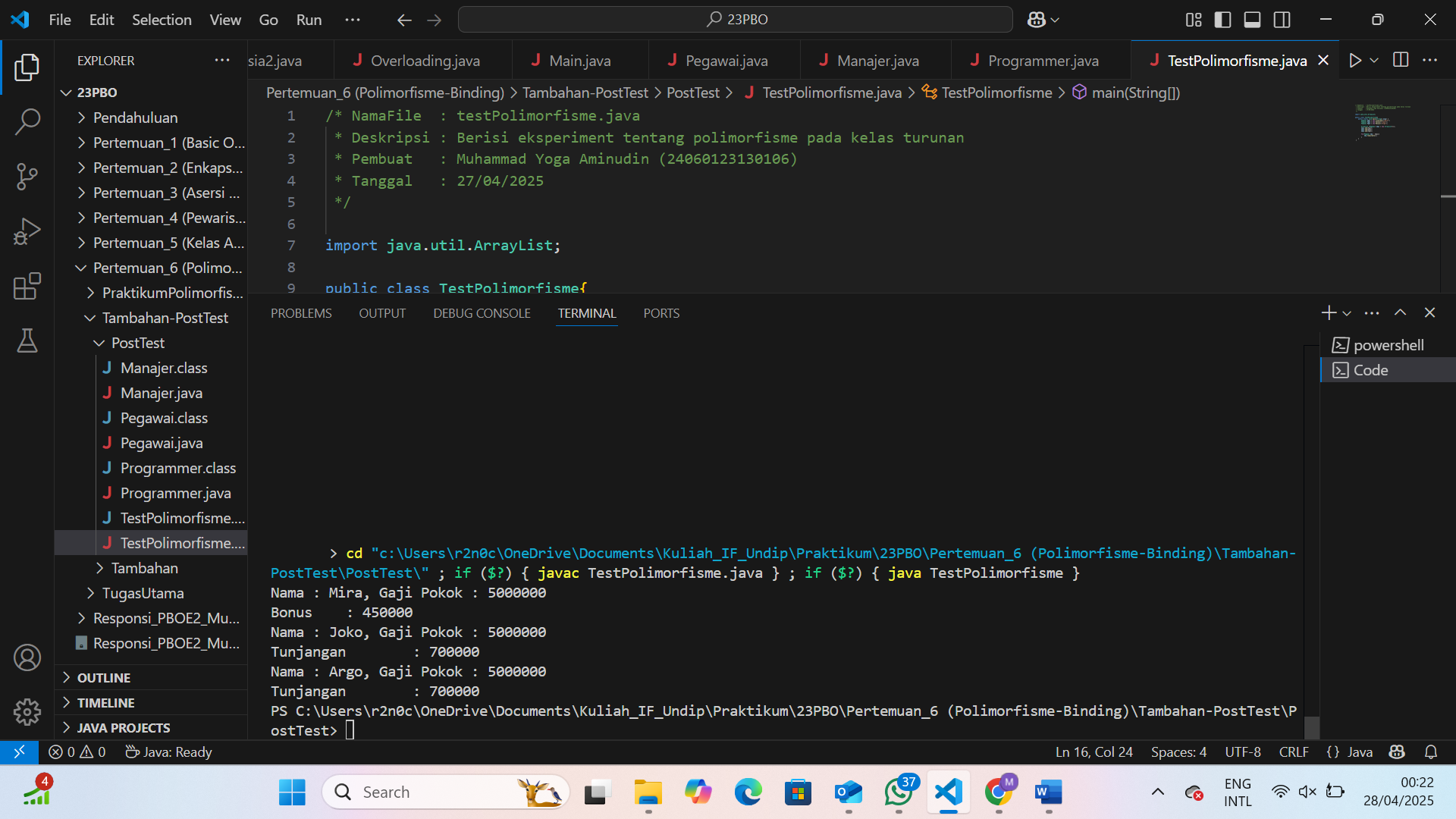
Programmer.java :



TestPolimorfisme.java :



Hasil Kode Program :



1. Jelaskan manfaat oclimorfisme pada kasus ini.

Polimorfisme dalam program ini memberikan beberapa keuntungan penting yang membuat kode menjadi lebih efisien dan mudah dikelola. Pertama, dengan menggunakan polimorfisme, kita dapat menyimpan berbagai jenis objek (seperti Manajer dan Programmer) dalam satu koleksi yang sama (ArrayList<Pegawai>), meskipun mereka memiliki perilaku yang berbeda. Hal ini memungkinkan kita untuk memproses semua pegawai dengan cara yang seragam melalui loop yang sama, tanpa perlu mengetahui secara spesifik jenis pegawai masing-masing. Ketika metode tampilData() dipanggil, Java secara otomatis akan memanggil implementasi yang sesuai untuk setiap jenis pegawai

Kedua, polimorfisme memudahkan pengembangan program di masa depan. Jika kita ingin menambahkan jenis pegawai baru (misalnya Desainer), kita cukup membuat class baru yang extends Pegawai dan mengoverride metode tampilData(), tanpa perlu mengubah kode yang sudah ada di kelas TestPolimorfisme.

1. Apabila pada main program perlu menambahkan pegawai4 dan pegawai5! Apa permasalahan yang muncul jika diterapkan tanpa polimorfisme (inclusion)?

Tanpa menggunakan polimorfisme, program akan menjadi jauh lebih kompleks dan sulit dikelola. Kita harus membuat koleksi terpisah untuk setiap jenis pegawai (satu ArrayList untuk Programmer dan satu lagi untuk Manajer), serta menulis logika pemrosesan yang terpisah untuk setiap koleksi. Setiap kali kita menambahkan jenis pegawai baru, kita harus menambah koleksi baru dan menulis kode pemrosesan baru. Pendekatan ini tidak hanya membuat kode menjadi lebih panjang dan berulang, tetapi juga meningkatkan risiko kesalahan dan membuat maintenance menjadi lebih sulit. Selain itu, tanpa polimorfisme, kita kehilangan fleksibilitas untuk menulis kode yang generik yang dapat bekerja dengan berbagai jenis pegawai, sehingga perubahan kecil pun bisa membutuhkan modifikasi di banyak tempat