LAPORAN PRAKTIKUM

STACK



Oleh :

MUHAMMAD GALID AVERO

NIM 2311532008

MATA KULIAH STRUKTUR DATA

DOSEN PENGAMPU : DR. WAHYUDI, S.T, M.T

ASISTEN LABORATORIUM : ZAKY ADIL HAKIM

FAKULTAS TEKNOLOGI INFORMASI

DEPARTEMEN INFORMATIKA

UNIVERSITAS ANDALAS

PADANG, MEI 2024

1. PENDAHULUAN

Stack adalah struktur data yang mengikuti prinsip “Last In, First Out” (LIFO). Artinya, data yang terakhir kali ditambahkan ke dalam stack akan menjadi data pertama yang diambil dari stack.

Berikut adalah beberapa operasi dasar yang biasanya dilakukan pada stack:

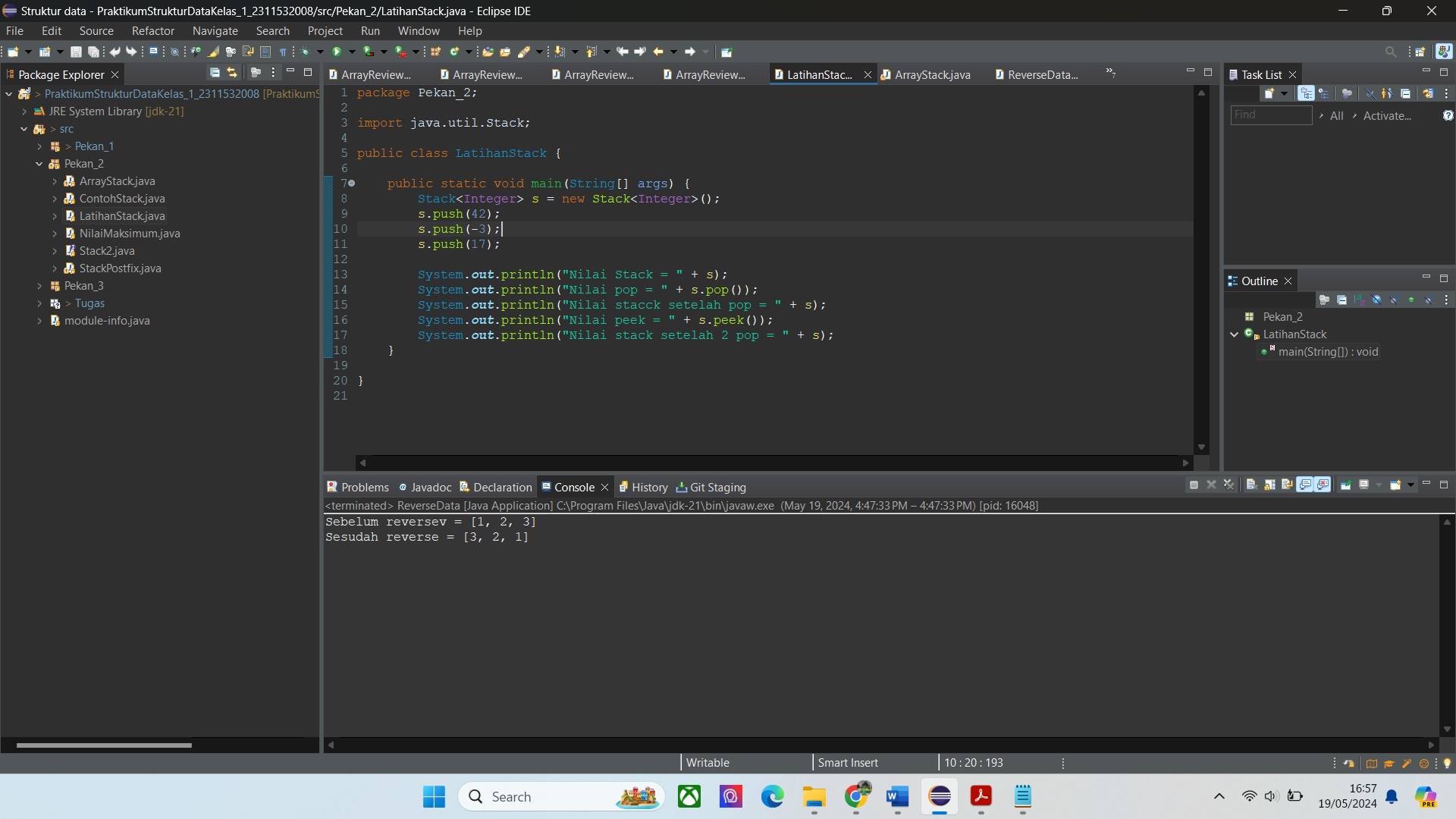
1. Push : Menambahkan elemen baru ke puncak stack.
2. Pop : Menghapus elemen dari puncak stack.
3. Peek/Top : Melihat elemen paling atas dari stack tanpa menghapusnya.
4. isEmpty : Memeriksa apakah stack kosong.

Contoh penggunaan stack dalam kehidupan sehari-hari adalah tumpukan piring. Piring yang terakhir ditambahkan ke tumpukan (push) adalah piring pertama yang akan diambil (pop) dari tumpukan tersebut. Demikian pula dalam pemrograman, stack digunakan dalam berbagai kasus seperti pengecekan sintaks, pemanggilan fungsi, dan lainnya.

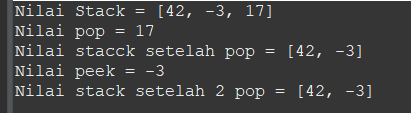
Implementasi Stack

1. Array : Implementasi stack dengan array memerlukan ukuran tetap yang ditentukan sebelumnya. Keuntungan dari metode ini adalah akses elemen yang cepat.
2. Linked List : Implementasi stack dengan linked list lebih fleksibel karena tidak memerlukan ukuran tetap dan dapat dengan mudah menyesuaikan dengan jumlah elemen yang berubah-ubah.

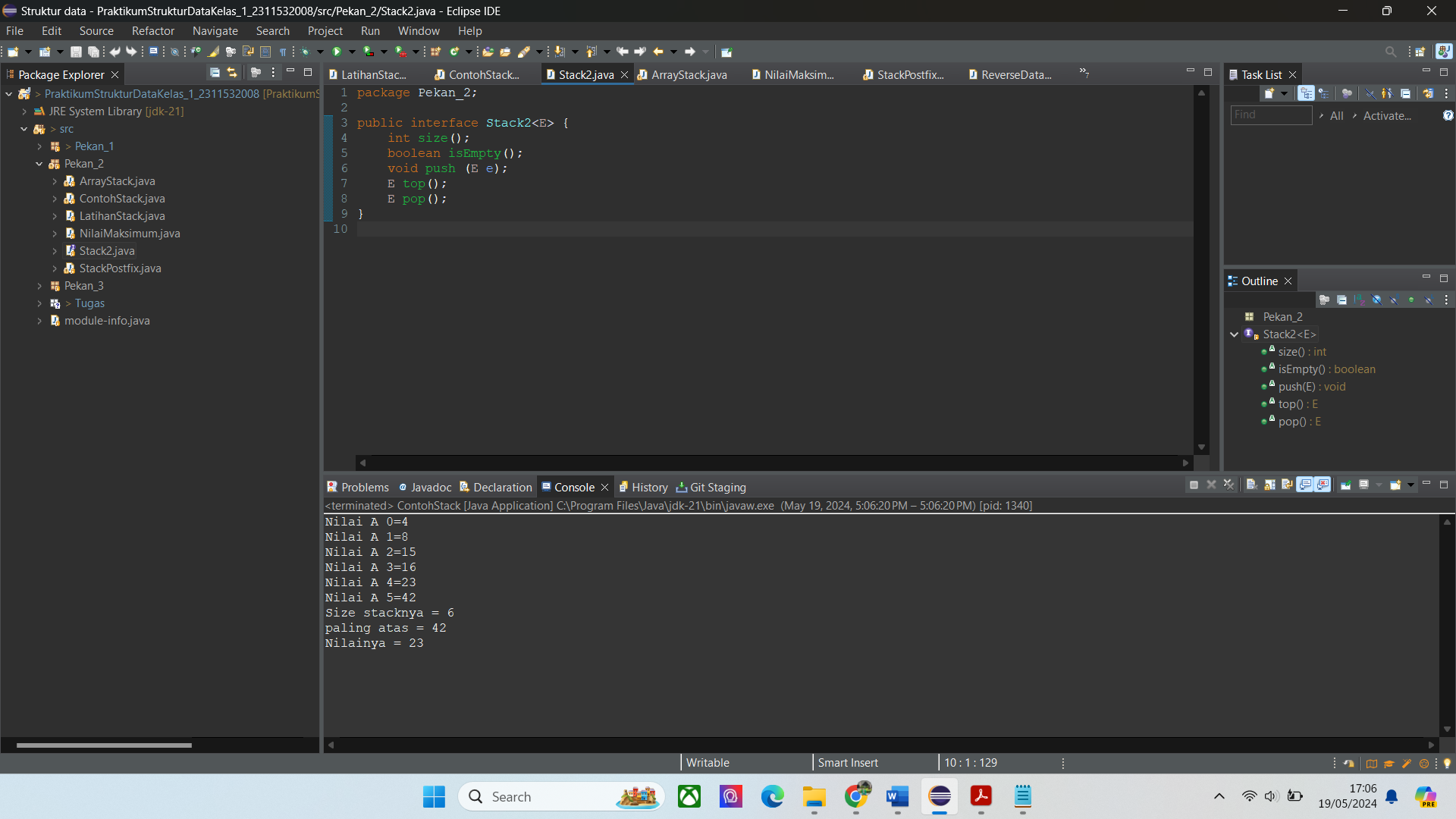
Kekurangan dari Stack :

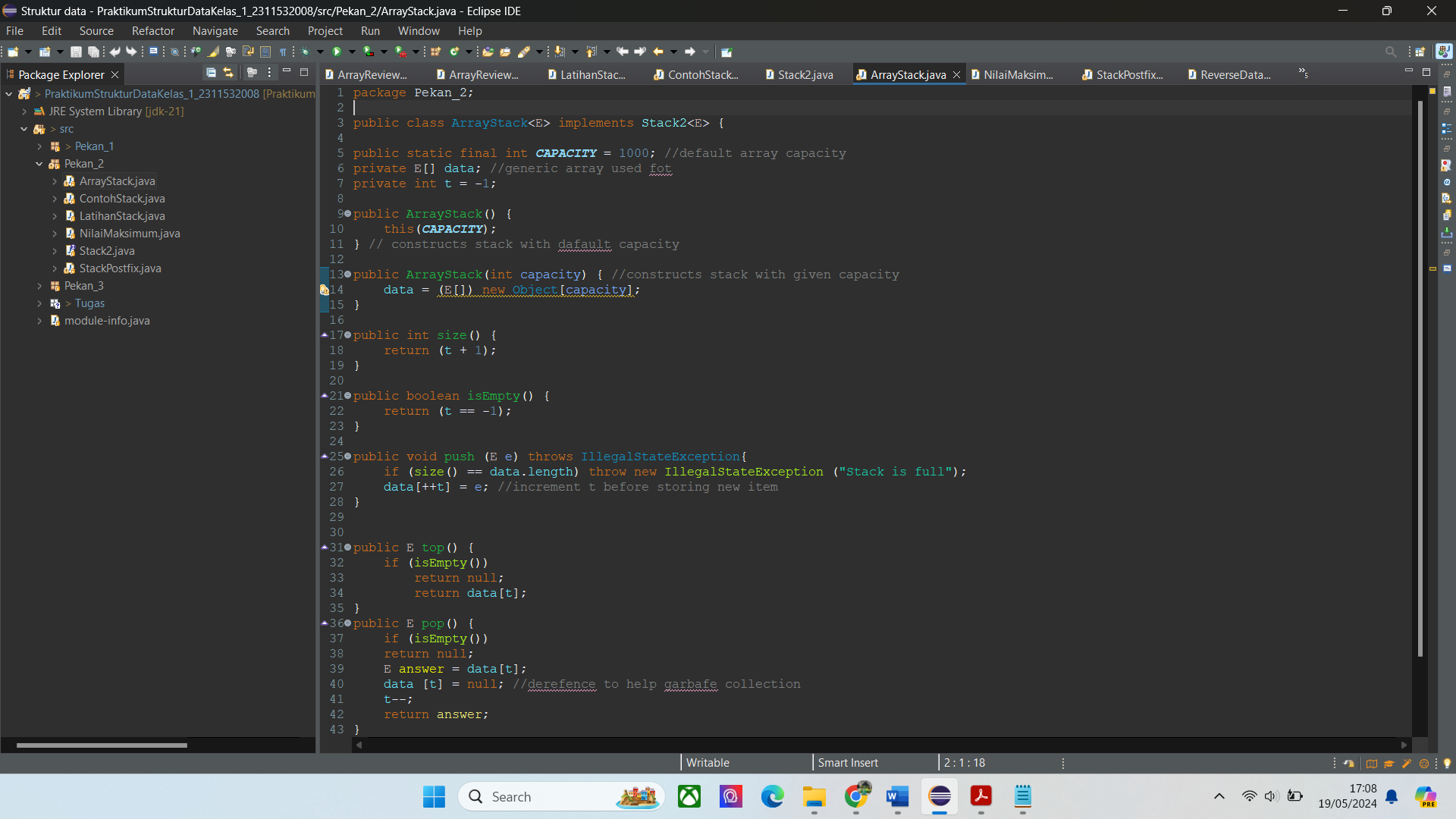
1. Sangat susah untuk stack dibuat banyak objet karena akan meningkatkan resiko Stack Overvlow
2. Memori nya sangat dibatasi
3. Tidak dapat menggunakan random access
4. TUJUAN PRAKTIKUM
5. Memahami cara kerja stack
6. Memahami cara kerja dan membuat ArrayStack.
7. Membuat program untuk menentukan nilai maksimum pada sebuah Stack.
8. Membuat program postfix pada Stack.
9. LANGKAH PRAKTIKUM
10. Cara Kerja Stack

Pada program tersebut, kita dapat menambahkan data ke dalam stack dengan method push(); dan kemudian cukup memanggil variabel stack untuk melihat isi stack secara keseluruhan. Jika kita ingin memanggil nilai stack paling atas dan menghapus data paling atas, kita dapat menggunakan method pop(). Jika kita hanya ingin memanggil nilai stack paling atas saja, kita dapat menggunakan method peek();.   
untuk hasilnya adalah sebagai berikut :



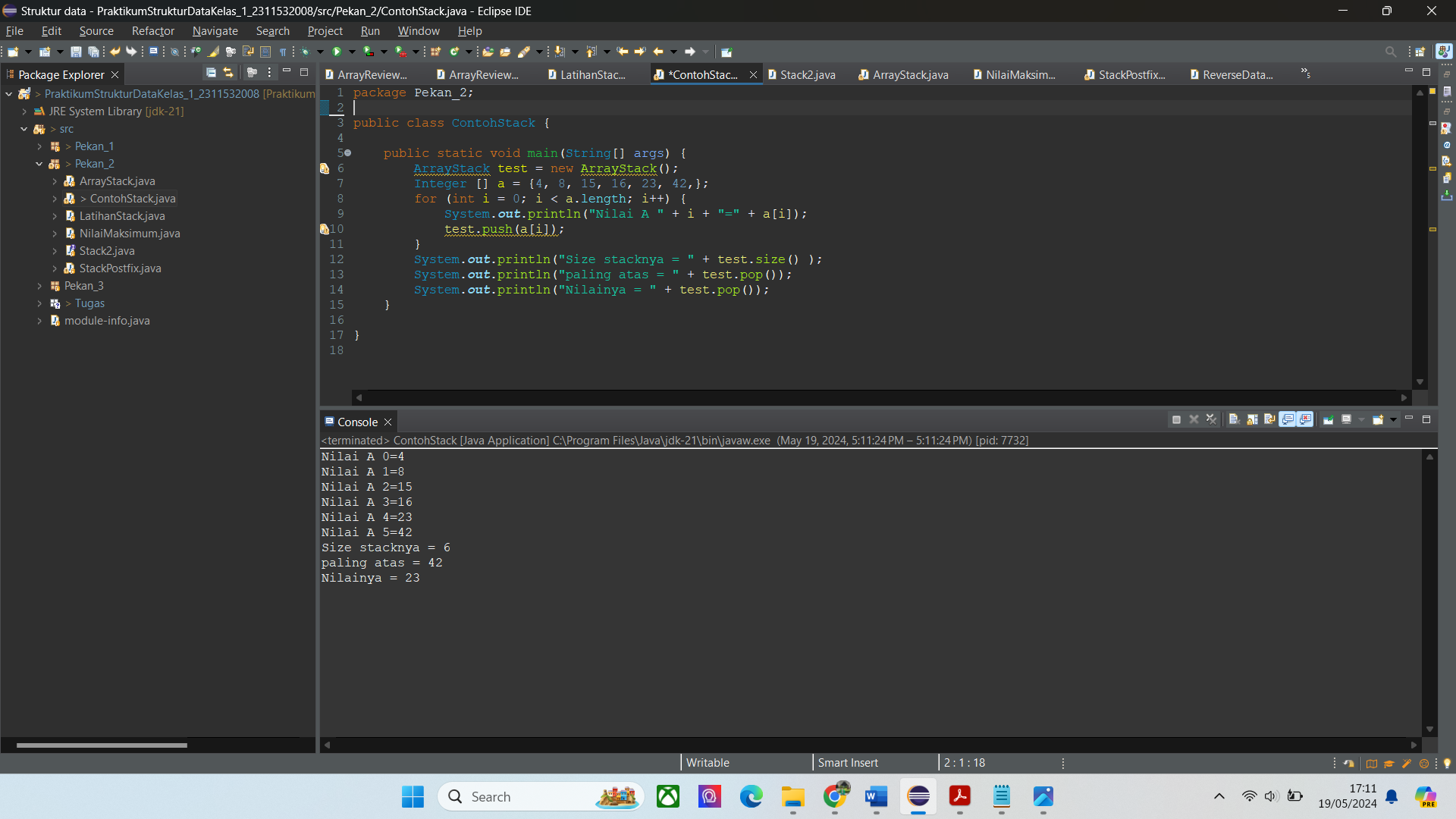
1. Interface dan Class ArrayStack



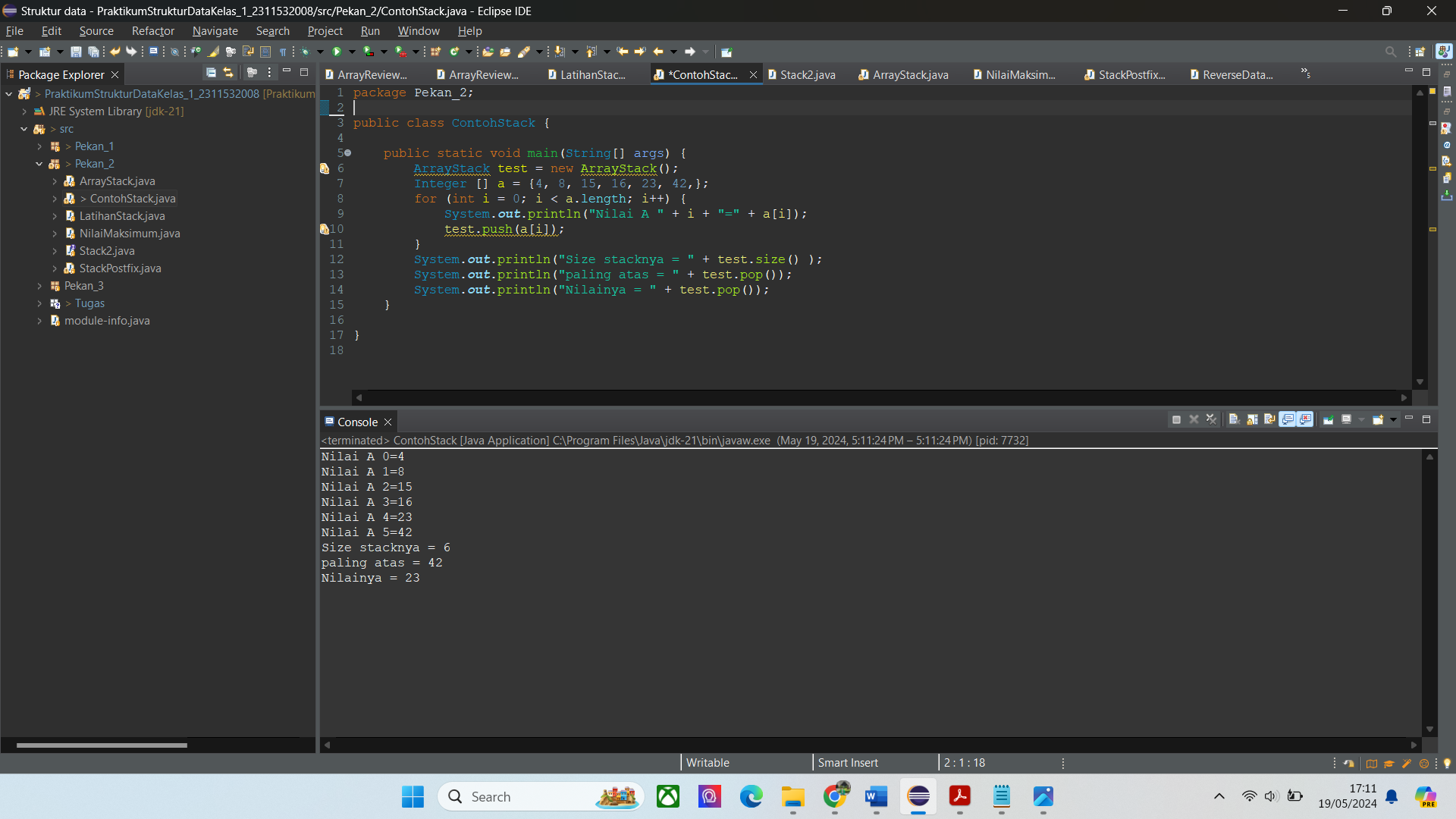


Program ini menjelaskan cara kerja stack. Kita dapat membuat sistem stack kita sendiri tanpa mengimport java.util.Stack: Program bernama ArrayStack menggunakan interface Stack2, seperti Stak biasa, yang menggunakan Array.   
Setiap metode diberikan rinciannya di bawah ini:

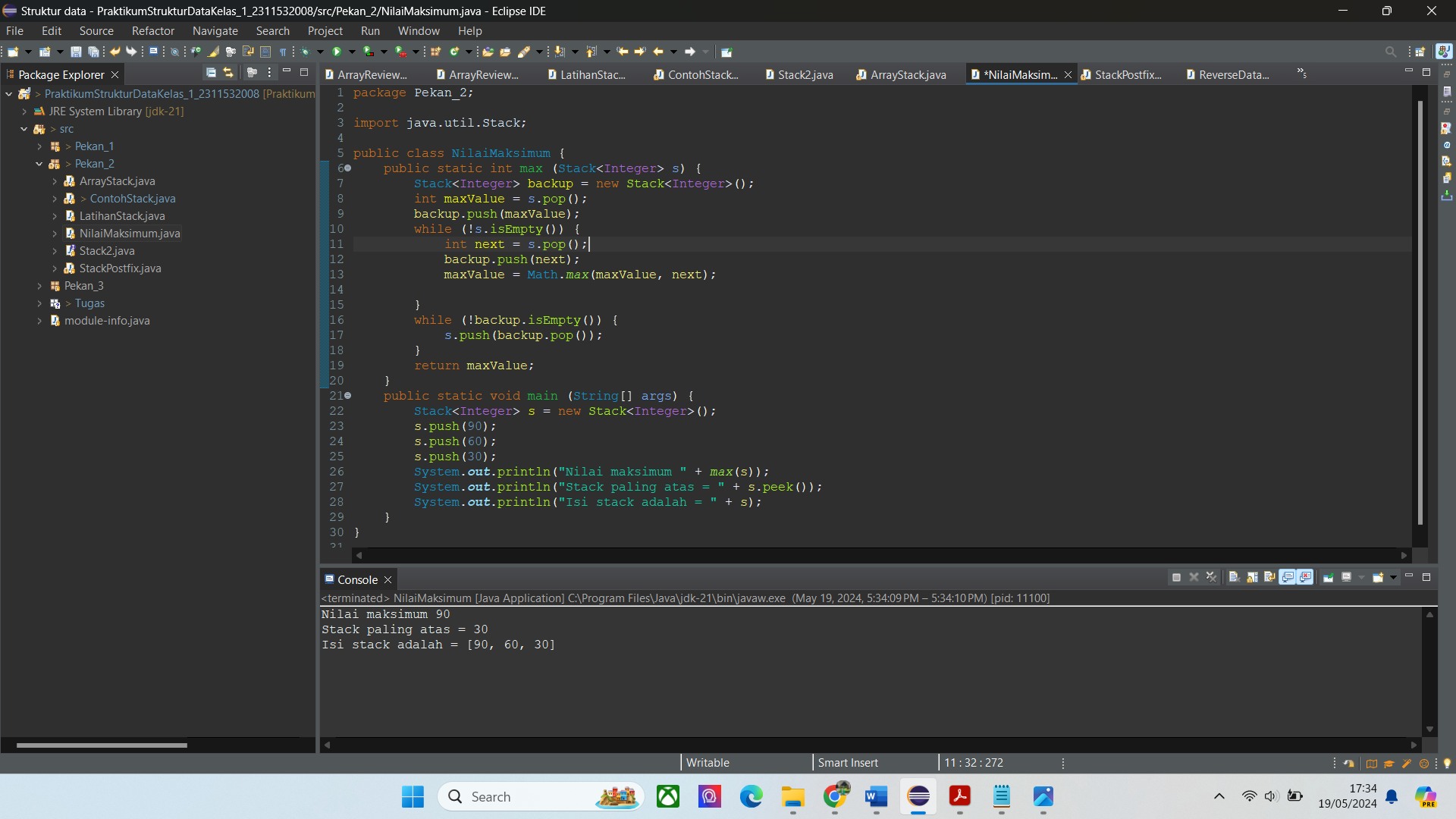
1. "public static final int CAPACITY = 1000;" menunjukkan jumlah maksimum elemen yang dapat disimpan di stack, yaitu 1000 buah.
2. "private E[] data;": dalam program ini, dasar stack adalah array yang diwakili oleh variabel data, sehingga data masuk akan tersimpan pada variabel ini.
3. "private int t = -1;": metode ini memberikan informasi tentang jumlah index pada stack, dengan -1 menunjukkan bahwa stack kosong.
4. public ArrayStack(): Constructor default untuk class ArrayStak, dengan kapasitas default 1000
5. - Public ArrayStack(int capacity): constructor overloaded yang akan menggunakan kapasitas sesuai input yang diberikan jika dipanggil.
6. Public int size(): metode untuk mengeluarkan ukuran stack. Metode ini melihat ukuran stack dengan mereturn t+1
7. public boolean isEmpty(): Metode ini mengecek apakah t = -1 yang menunjukkan stack kosong; jika stack kosong, maka bernilai benar, dan jika stack tidak t=0 maka bernilai false.
8. Dengan public void push(E e), metode ini memasukkan data ke dalam stack dan melihat kapasitas yang diberikan saat pendeklarasian. Jika metode dipanggil lebih dari kapasitasnya, akan mengeluarkan "Stack is full"
9. public E top(): Metode ini akan mengeluarkan data teratas dari stack.
10. Metode publik E pop(): Mengeluarkan data teratas dan menghapusnya dari stack. Jika stack kosong, return null. Selain itu, metode ini akan mengurangi nilai t.
11. Implementasi Stack



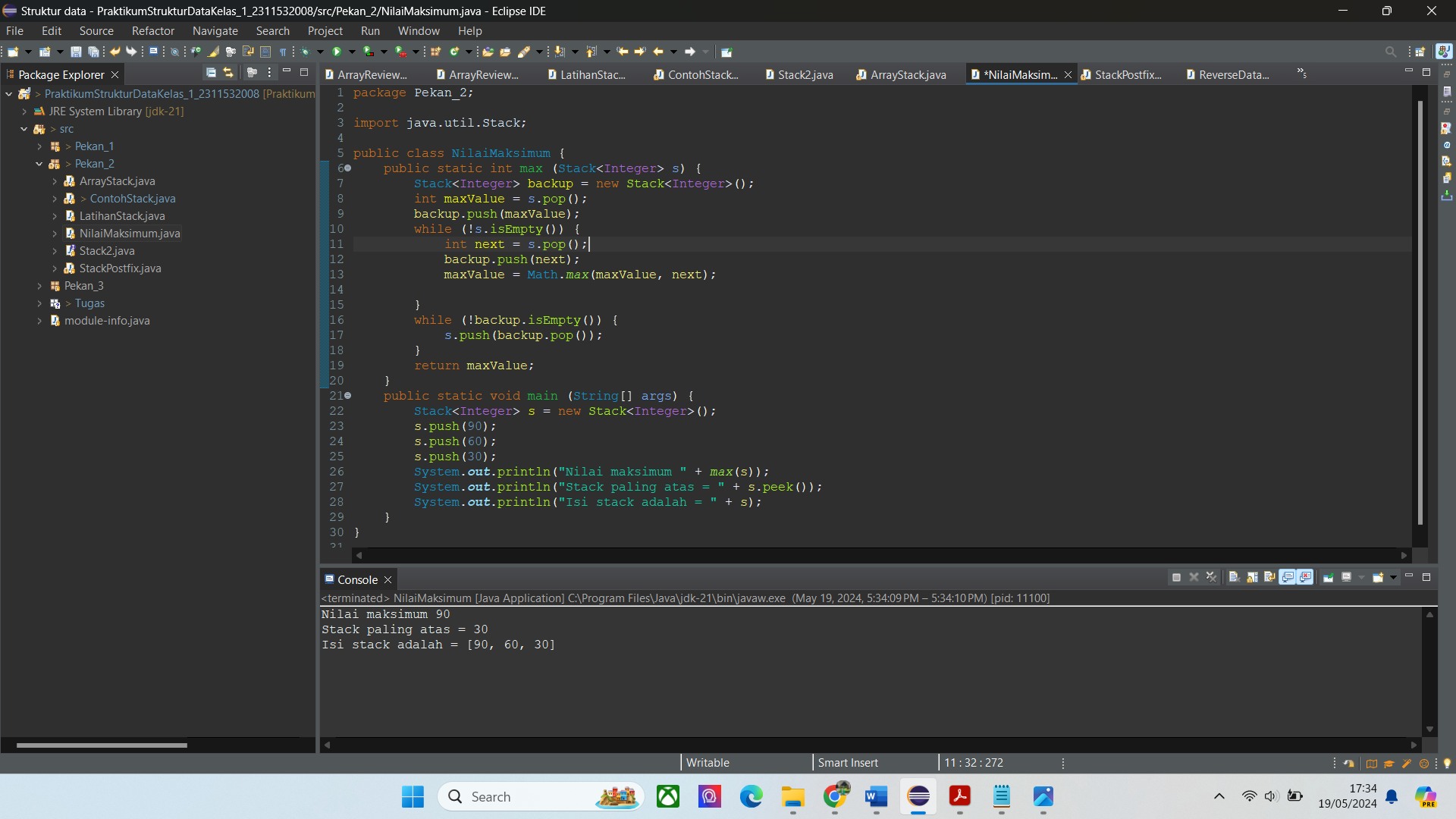
Pada kode di atas menggunakan implementasi dari kode class dan juga interface yang telah dibuat sebelumnya. Untuk hasil dari kode ini akan terlihat seperti berikut :



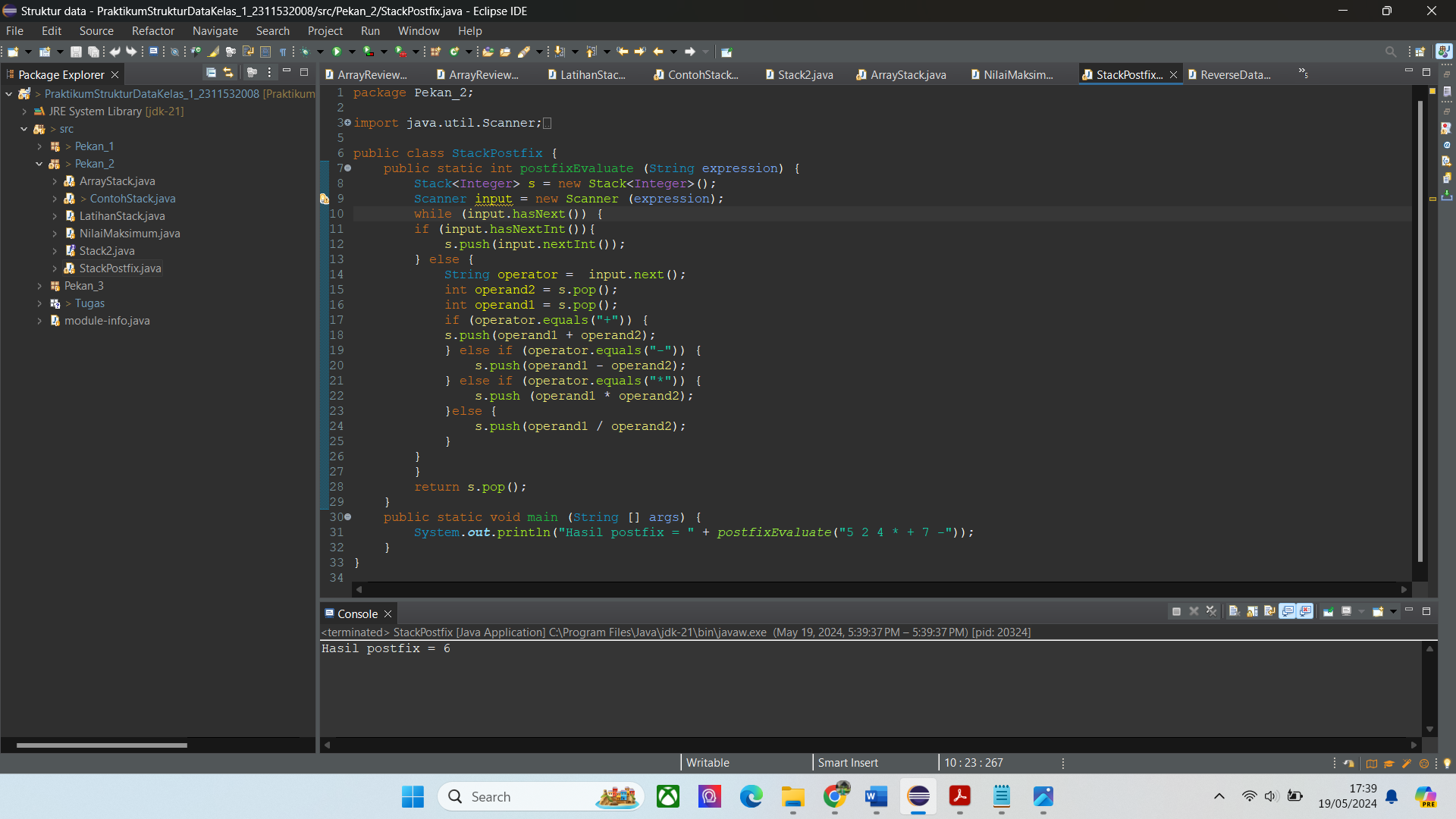
Contoh lainnya



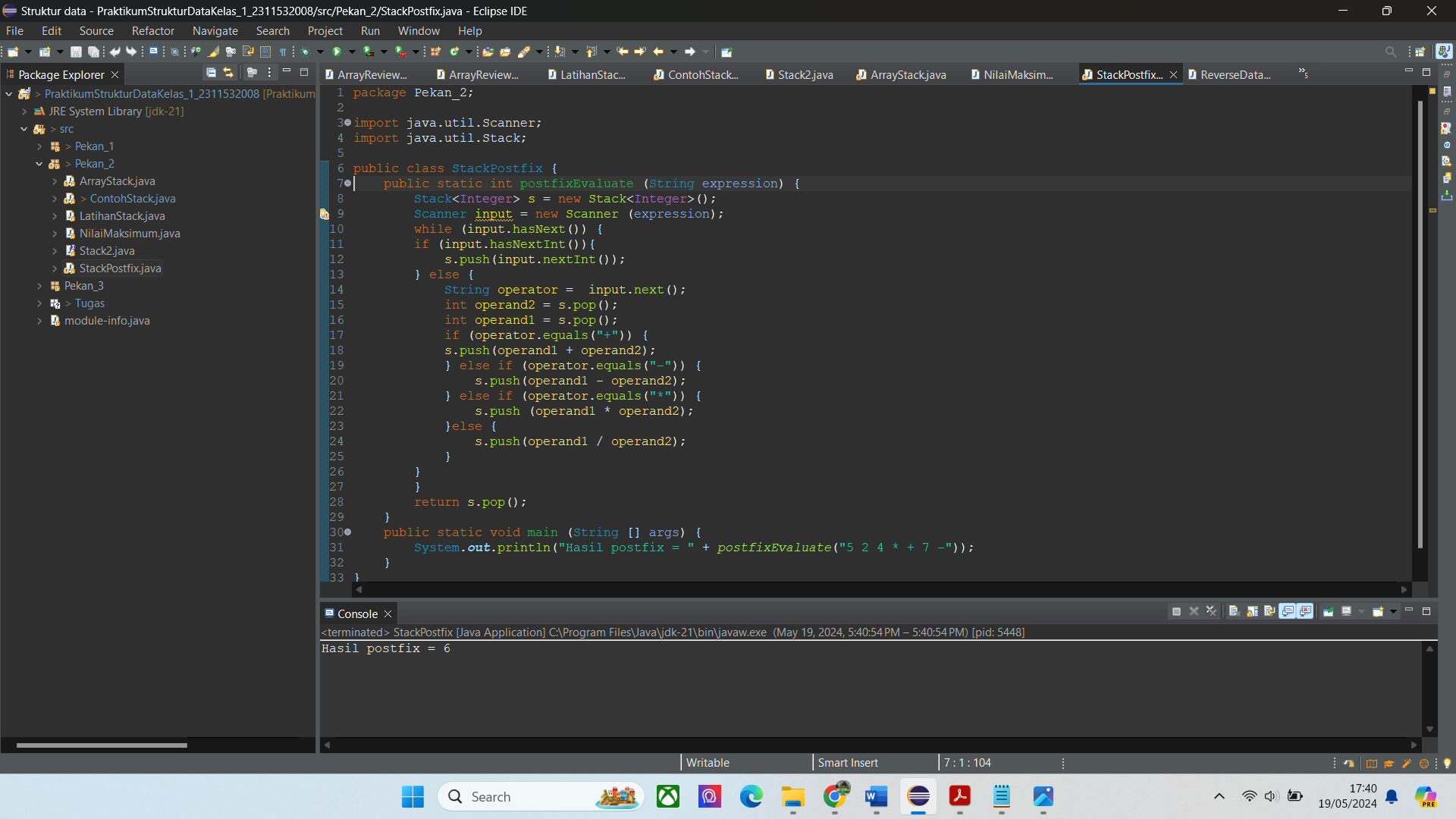
Kita membuat dua stack dalam program tersebut: stack utama pada metode main dan stack backup pada metode max. Program ini berfungsi untuk menemukan nilai maksimum dari sebuah stack dengan memasukkan stack ke metode max(), lalu stack ini dimasukkan ke pop() dan dimasukkan ke variabel maxValue. Kami juga membuat variabel kedua, variabel next, untuk memasukkan data kedua sebagai perbandingan dari nilai maksimal. Untuk hasilnya adalah sebagai berikut :



1. Penggunaan Stack pada postfix



Pada kode diatas dilakukan operasi postfix yang mana pada operasi tersebut operator dan operand ditulis dari kiri ke kanan tanpa adanya urutan. Cara pengkodean nya adalah terlebih dahulu diimport library Scanner dan juga library Stack lalu digunakan logika if, else if, dan juga else untuk menjalankan program tersebut. Berikut untuk hasil dari kode diatas



1. KESIMPULAN

Sebagai hasil dari praktikum yang telah dilakukan, dapat disimpulkan bahwa "stack" adalah kumpulan elemen yang disusun secara linear yang memiliki dua fungsi utama: push (menambahkan elemen ke dalam kumpulan) dan push (menambahkan elemen ke dalam kumpulan). pop, yang berarti menghapus elemen tertinggi dari stack. Stack biasanya disusun sesuai dengan prinsip LIFO, atau Last In, First Out, yang berarti elemen pertama yang dihapus adalah yang terakhir yang dimasukkan ke dalam stack. Penulis akan membuat program stack sederhana yang menggunakan operasi push, pop, dan peek pada praktikum kali ini. Selanjutnya, kami membuat class Interface dan class tambahan yang akan mengimplementasikan program ArrayStack, juga membuat program untuk menentukan nilai maksimum stack dan program untuk membuat postfix pada stack.