**LAPORAN PRAKTIKUM 3**

**STRUKTUR DATA**

**A logo of a university

AI-generated content may be incorrect.**

Dosen Pengampu:  
Dr. Wahyudi, M.Kom.

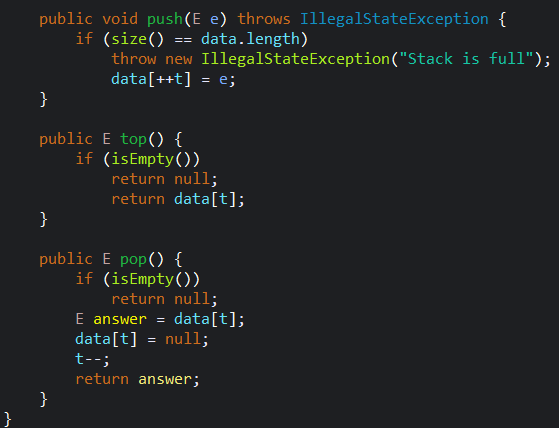
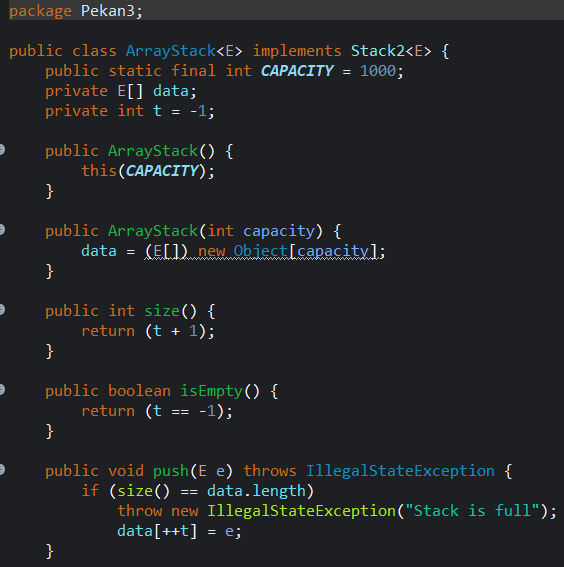
Disusun Oleh:  
Muhammad Luthfi Kautsar Rizata – 2311532020

**DEPARTEMEN INFORMATIKA**

**FAKULTAS TEKNOLOGI INFORMASI**

**UNIVERSITAS ANDALAS**

**2025**

1. **Class ArrayStack**

Kelas ArrayStack<E> merupakan implementasi dari struktur data stack berbasis array dengan pendekatan generic, sehingga dapat menyimpan elemen bertipe data apapun. Stack merupakan struktur data LIFO (*Last In, First Out*), di mana elemen terakhir yang masuk akan menjadi elemen pertama yang keluar.

Secara internal, ArrayStack menyimpan elemen-elemen stack dalam array data, dengan kapasitas yang bisa ditentukan oleh pengguna melalui konstruktor. Jika kapasitas tidak ditentukan, maka digunakan nilai default sebesar 1000, sebagaimana didefinisikan dalam konstanta CAPACITY. Elemen paling atas dari stack dilacak oleh variabel t, yang merupakan indeks dari elemen terakhir yang dimasukkan. Nilai awal t adalah -1, yang menunjukkan bahwa stack dalam keadaan kosong.

Kelas ini menyediakan beberapa operasi dasar stack:

* **size()** mengembalikan jumlah elemen dalam stack, dihitung sebagai t + 1.
* **isEmpty()** memeriksa apakah stack kosong, yaitu jika nilai t adalah -1.
* **push(E e)** menambahkan elemen baru ke atas stack. Jika stack sudah penuh (jumlah elemen sama dengan panjang array), maka akan melempar exception IllegalStateException.
* **top()** mengembalikan elemen paling atas stack tanpa menghapusnya. Jika stack kosong, maka akan mengembalikan null.
* **pop()** menghapus dan mengembalikan elemen paling atas. Jika stack kosong, maka akan mengembalikan null. Setelah elemen dihapus, posisi dalam array di-*null*-kan agar tidak terjadi memory leak, dan indeks t dikurangi satu.

Secara umum, implementasi ini menggambarkan bagaimana stack dapat direpresentasikan secara efisien menggunakan array dengan batas kapasitas tertentu.

1. **Interface Stack2**

A screen shot of a computer program

AI-generated content may be incorrect.Kode di samping mendefinisikan antarmuka (interface) Stack2<E> yang merepresentasikan struktur data **stack** dalam bentuk abstrak menggunakan generic parameter <E>, sehingga dapat digunakan untuk berbagai tipe data. Antarmuka ini berisi deklarasi metode-metode dasar yang umumnya terdapat pada struktur stack, tanpa memberikan implementasi langsung.

Metode size() berfungsi untuk mengembalikan jumlah elemen yang saat ini terdapat di dalam stack. isEmpty() mengembalikan nilai boolean yang menunjukkan apakah stack kosong atau tidak. Metode push(E e) digunakan untuk menambahkan elemen e ke atas stack, sedangkan pop() digunakan untuk menghapus dan mengembalikan elemen paling atas dari stack. Terakhir, metode top() berfungsi untuk melihat elemen paling atas dari stack tanpa menghapusnya. Dengan adanya interface ini, berbagai kelas dapat mengimplementasikan stack dengan cara yang berbeda (misalnya berbasis array, linked list, atau struktur lainnya), namun tetap menjamin bahwa semua kelas tersebut menyediakan operasi-operasi dasar yang konsisten seperti yang didefinisikan dalam Stack2.

1. A screen shot of a computer program

   AI-generated content may be incorrect.**Class NilaiMaksimum**

Kode di atas merupakan implementasi program Java yang bertujuan untuk mencari nilai maksimum dari elemen-elemen yang terdapat dalam suatu stack bertipe Integer, tanpa mengubah urutan asli elemen dalam stack tersebut setelah proses pencarian selesai.

Fungsi max(Stack<Integer> s) menerima sebuah objek stack sebagai parameter dan bertugas untuk mencari nilai maksimum dari semua elemen yang ada di dalamnya. Proses diawali dengan membuat stack bantu bernama backup yang akan menyimpan elemen-elemen sementara selama proses pencarian berlangsung. Nilai maksimum awal diambil dari elemen pertama yang di-*pop* dari stack s, kemudian nilai ini disimpan ke dalam backup.

Selanjutnya, selama stack s belum kosong, elemen-elemen berikutnya di-*pop* satu per satu dan dibandingkan dengan maxValue menggunakan fungsi Math.max(). Nilai maksimum yang ditemukan selama iterasi akan terus diperbarui. Setiap elemen yang diambil dari s juga disimpan sementara di stack backup agar bisa dikembalikan lagi ke s nanti.

Setelah seluruh elemen diperiksa, dilakukan proses pemulihan isi stack ke bentuk semula dengan mem-*pop* elemen dari backup dan mendorongnya kembali (push) ke dalam s. Dengan cara ini, urutan elemen dalam stack tetap terjaga seperti awal, dan fungsi mengembalikan nilai maksimum yang telah ditemukan.

Pada metode main, dibuat sebuah stack dan dimasukkan tiga buah elemen: 70, 12, dan 20. Kemudian, isi stack ditampilkan, diikuti oleh nilai teratas stack (menggunakan peek()), dan terakhir nilai maksimum yang ditemukan dengan memanggil fungsi max().

1. A computer screen shot of a code

   AI-generated content may be incorrect.**Class StackPostfix**

Kode di atas merupakan implementasi evaluasi ekspresi matematika dalam notasi postfix menggunakan struktur data stack. Notasi postfix menyatakan operasi matematika di mana operator ditempatkan setelah operand-operandnya, sehingga tidak membutuhkan tanda kurung dan mengandalkan urutan saja.

Fungsi postfixEvaluate(String expression) menerima sebuah ekspresi dalam bentuk string yang terdiri dari angka dan operator, dipisahkan oleh spasi. Ekspresi ini kemudian di-*parse* menggunakan objek Scanner, dan tiap token akan diproses satu per satu dalam sebuah loop.

Jika token yang dibaca adalah angka (dicek dengan hasNextInt()), maka angka tersebut akan dimasukkan ke dalam stack. Namun jika token berupa operator (+, -, \*, atau /), maka dua elemen teratas dari stack akan diambil sebagai operand. Pengambilan dimulai dari operand kedua (operand2) lalu operand pertama (operand1), mengikuti urutan operasi postfix. Setelah itu, dilakukan operasi aritmatika sesuai dengan operator yang ditemukan, dan hasilnya dimasukkan kembali ke dalam stack.

Setelah seluruh token dalam ekspresi selesai diproses, hasil akhir dari evaluasi ekspresi berada di puncak stack dan di-*pop* untuk dikembalikan sebagai output.

Pada metode main, dilakukan pengujian terhadap ekspresi "5 2 5 \* + 7 -". Ekspresi ini dapat diuraikan sebagai:

* 2 \* 5 = 10
* 5 + 10 = 15
* 15 - 7 = 8

Sehingga, hasil akhirnya adalah 8, yang ditampilkan lewat pernyataan System.out.println().

1. **Class contohStack**

A computer screen shot of text

AI-generated content may be incorrect.Kode di atas merupakan program sederhana yang mendemonstrasikan penggunaan dasar struktur data stack di Java dengan menggunakan kelas Stack dari library java.util. Program ini menampilkan bagaimana data dimasukkan (*push*), dilihat (*peek*), dan diambil (*pop*) dari stack.

Pertama, dibuat sebuah objek stack bernama test yang menyimpan elemen bertipe Integer. Kemudian dideklarasikan sebuah array a berisi enam angka: 4, 8, 15, 16, 23, dan 42. Melalui perulangan for, setiap elemen array dicetak ke layar, lalu dimasukkan ke dalam stack satu per satu menggunakan metode push(). Karena stack bersifat *Last In, First Out (LIFO)*, maka elemen terakhir yang dimasukkan (dalam hal ini angka 42) akan berada di posisi paling atas.

Setelah seluruh elemen dimasukkan, program mencetak ukuran stack dengan test.size(), yang menunjukkan jumlah elemen saat ini. Kemudian, test.peek() digunakan untuk melihat elemen paling atas tanpa menghapusnya dari stack. Terakhir, test.pop() digunakan untuk mengambil elemen paling atas dari stack (yaitu 42) dan sekaligus menghapusnya dari struktur stack.

1. A computer screen shot of a program code

   AI-generated content may be incorrect.**Class latihanStack**

Kode di atas merupakan program sederhana yang menunjukkan perilaku dasar dari struktur data stack di Java melalui contoh penggunaan kelas Stack dari package java.util. Tujuan utamanya adalah memberikan ilustrasi mengenai bagaimana elemen-elemen dimasukkan ke dalam stack (*push*), diambil dan dihapus (*pop*), serta dilihat tanpa dihapus (*peek*).

Pertama, dibuat objek stack bertipe Integer dengan nama s, kemudian tiga elemen ditambahkan secara berurutan: 42, -3, dan 17. Karena stack bersifat *Last In, First Out (LIFO)*, maka elemen terakhir yang masuk (yaitu 17) akan berada di posisi paling atas. Cetakan pertama menggunakan System.out.println("nilai stack = " + s) menunjukkan urutan elemen di stack dari bawah ke atas.

Kemudian, s.pop() digunakan untuk menghapus dan mengembalikan elemen paling atas (dalam hal ini 17), sehingga hasil pop akan ditampilkan dan stack akan berisi 42 dan -3. Setelah itu, s.peek() dipanggil untuk melihat nilai yang sekarang berada di puncak stack tanpa menghapusnya (yaitu -3), dan stack akan tetap berisi elemen yang sama.