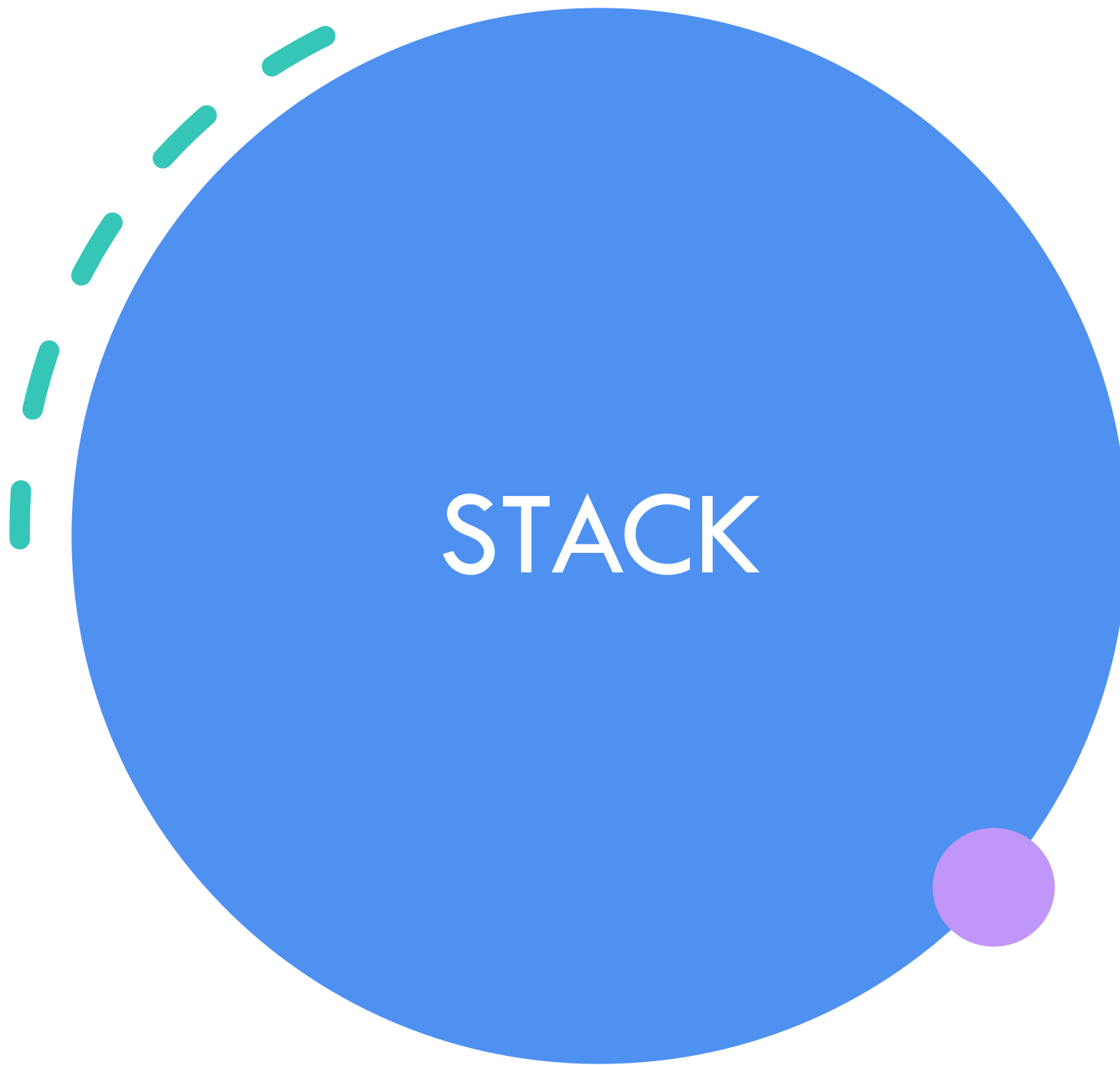




Implementasi Struktur Data

Stack dan Queue

Program Studi Diploma III Teknik Informatika
Fakultas Ilmu Terapan
Universitas Telkom



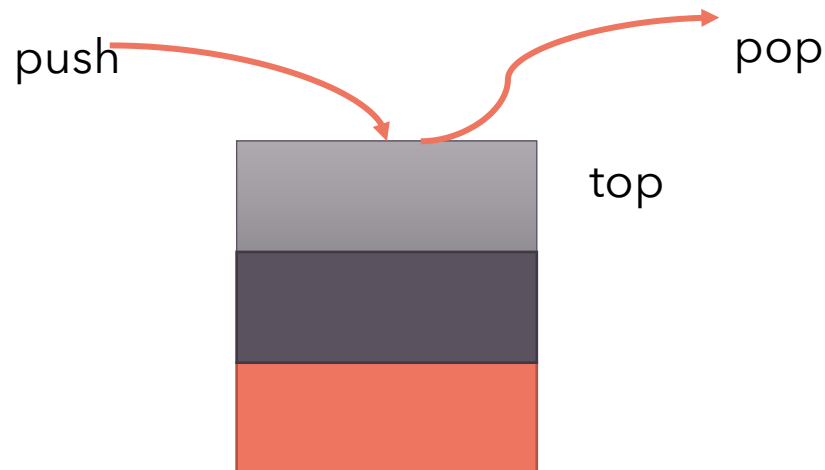
stack

- Stack merupakan kumpulan objek yang cara menambahkan dan menghapus objeknya adalah dengan prinsip Last-In, First-Out (LIFO)



Stack

- Insert objek dilakukan pada satu sisi stack, yang disebut top.
 - Jadi, top entry - masukan yang berada pada posisi top, merupakan objek yang paling baru dari seluruh isi stack
- Operasi dasar yang dilakukan pada stack adalah push dan pop



Computer Usage

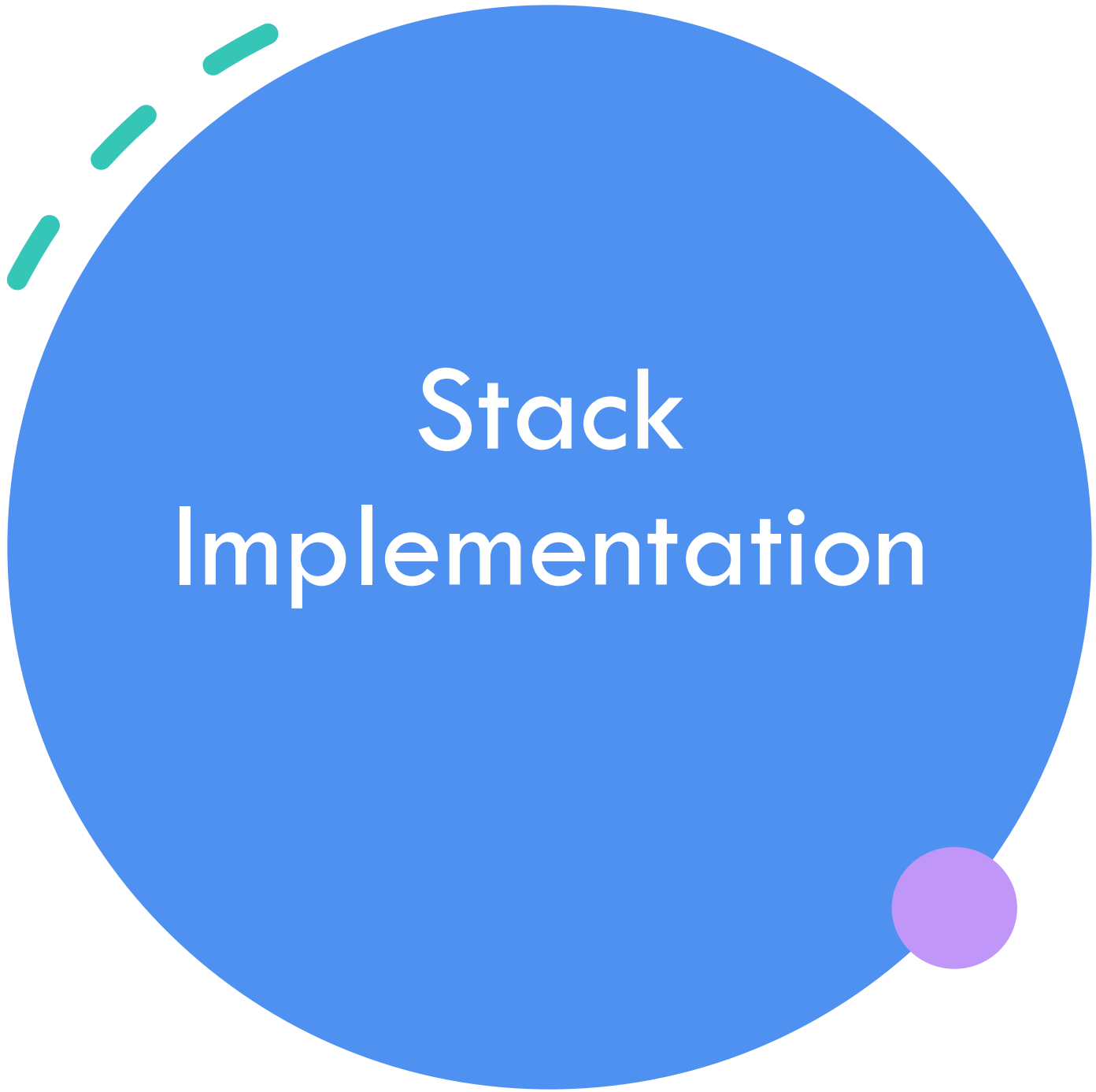
- Rekursif
- Notasi Aljabar (Notasi Postfix/ Polish Notation)
- Browser internet menyimpan alamat dari situs yang paling akhir dikunjungi dengan cara stack.
 - Setiap kali user mengunjungi suatu situs baru, situs tersebut akan di-push ke stack dari alamat. Operasi pop dilakukan ketika tombol back ditekan
- Mekanisme "undo" → perubahan pada text disimpan dalam suatu stack

ADT Stack

- Stack membatasi akses pada objek yang berada didalamnya. Isi dari stack hanya dapat dilihat atau dihapus dari top saja.
 - Jika isi yang diinginkan tidak berada pada top? → Hapus satu per satu sampai menemukan isi tersebut
- Jika seluruh isi stack dikeluarkan satu per satu, maka akan terbentuk urutan yang terbalik dari masukan stack, diawali dari yang paling akhir dimasukkan, dan diakhiri dengan objek yang pertama kali dimasukkan
- Pencarian data pada stack biasanya tidak dilakukan

ADT Stack

- ❖ `push(e)`: Adds element `e` to the top of the stack.
- ❖ `pop()`: Removes and returns the top element from the stack (or null if the stack is empty).
- ❖ `peek()`: Returns the top element of the stack, without removing it (or null if the stack is empty).
- ❖ `size()`: Returns the number of elements in the stack.
- ❖ `isEmpty()`: Returns a boolean indicating whether the stack is empty.



Stack Implementation

Array based

- Elemen stack disimpan di dalam array (misal, data) yang memiliki kapasitas N
- Elemen paling bawah disimpan pada indeks pertama array ($\text{data}[0]$)
- Elemen paling atas disimpan pada $\text{data}[\text{top}]$, dengan $\text{top} \leq \text{ukuran stack saat ini}$



Array based

- Ukuran stack adalah $\text{top} + 1$
- Ketika stack kosong, $\text{top} = -1$
 - Dengan ukuran awal stack = nol ($\text{top} = -1 + 1$)

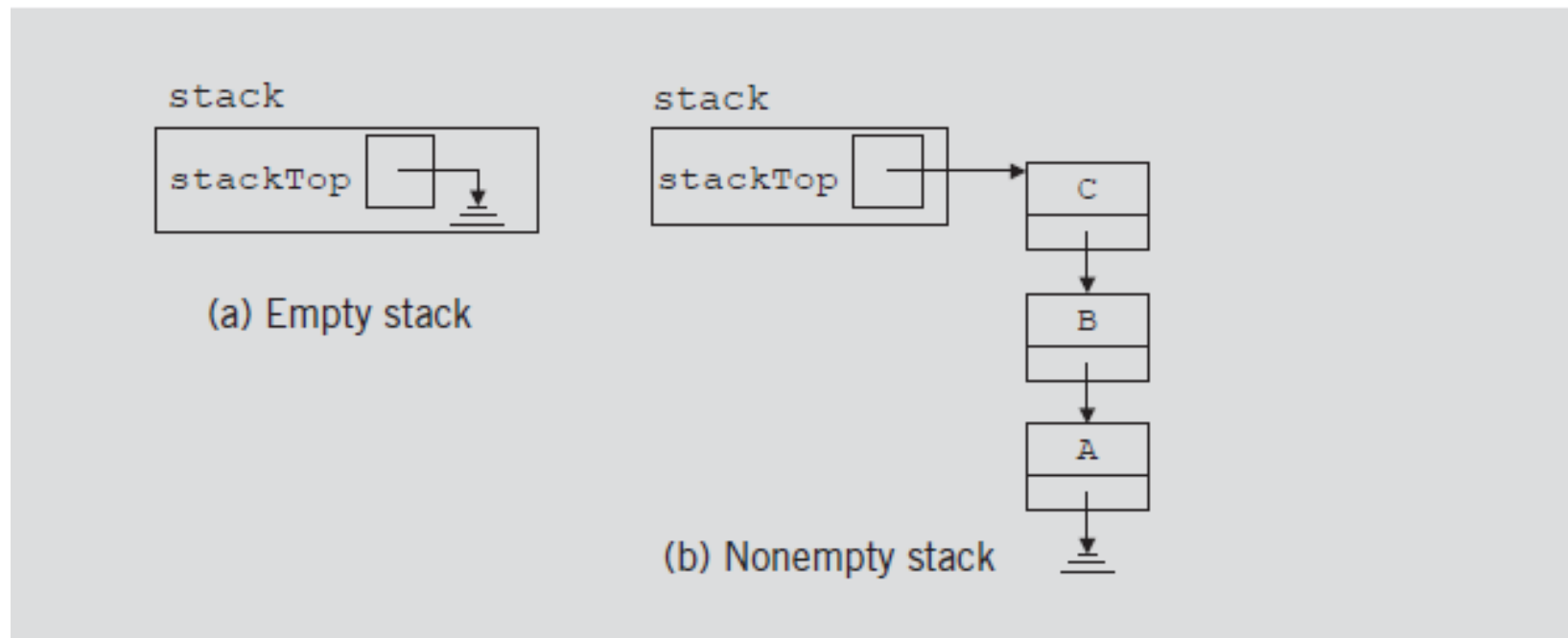
Array based Stack

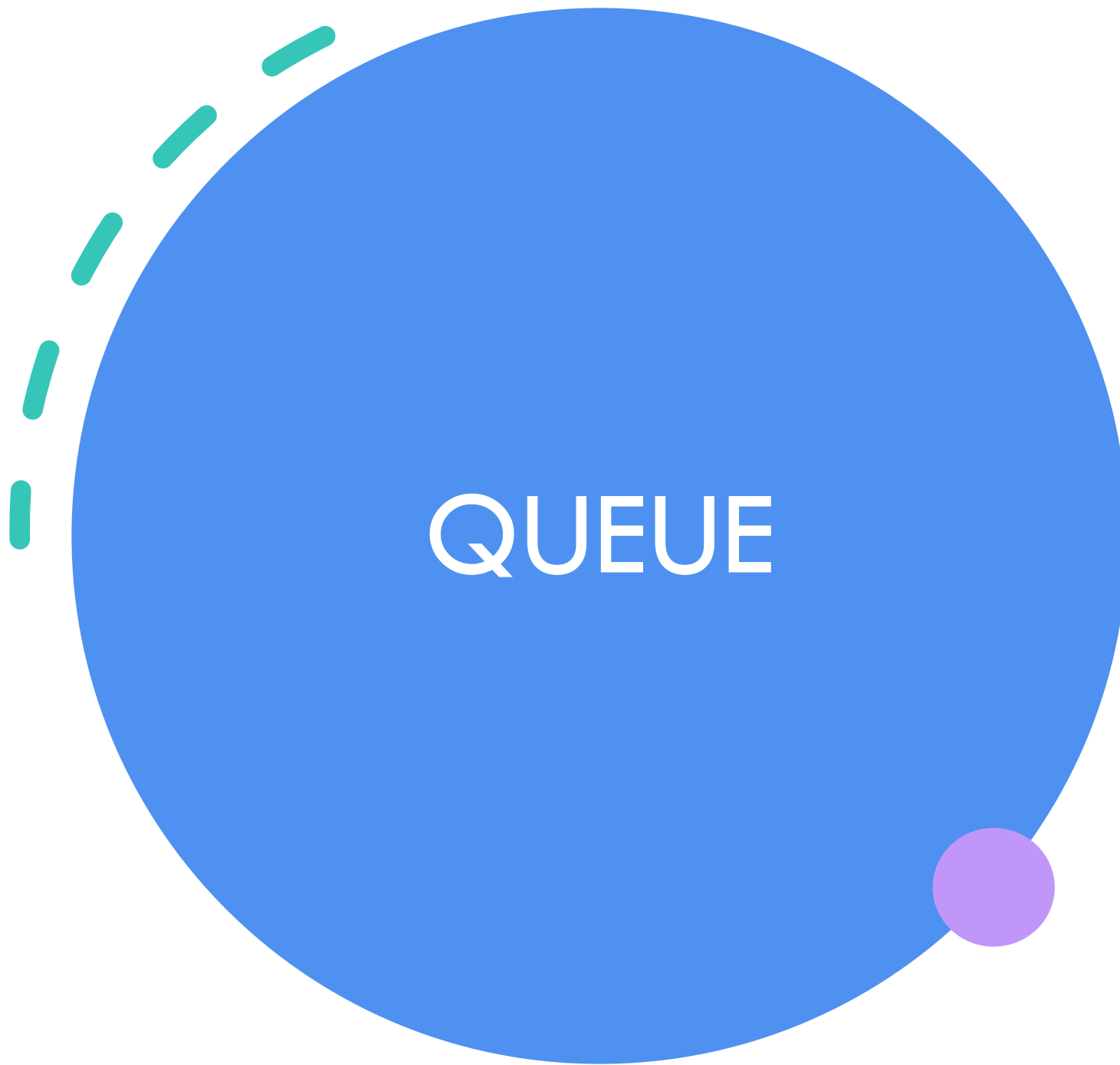
- Push : $\text{Top} = \text{Top} + 1$
- Pop : $\text{Top} = \text{Top} - 1$

Linked-list based

- Representasi Stack secara logik digambarkan sebagai list linier (singly linked list)
- Referensi pertama dari linked list (head) akan menjadi top
- Push dilakukan dengan sisip depan
- Pop dilakukan dengan hapus depan

Linked List based





Queue


- Merupakan kumpulan objek yang menggunakan prinsip First-In, First-Out (FIFO) pada proses memasukkan dan menghapus objek
- Jadi, elemen yang paling awal dimasukkan yang akan pertama kali dikeluarkan
- Elemen dimasukkan di sisi belakang, dan dikeluarkan dari sisi depan

Queue - EXAMPLE

- Contoh queue pada kehidupan nyata adalah saat kita mengantri: antrian bank, bioskop, atau membayar di kantin
- Pada komputer, antrian digunakan pada printer, juga pada Web server ketika merespon request dari client.

ADT Queue

- `enqueue(e)`: Adds element `e` to the back of queue.
- `dequeue()`: Removes and returns the first element from the queue
- (or null if the queue is empty).
- `first()`: Returns the first element of the queue, without removing it
- (or null if the queue is empty).
- `size()`: Returns the number of elements in the queue.
- `isEmpty()`: Returns a boolean indicating whether the queue is empty.



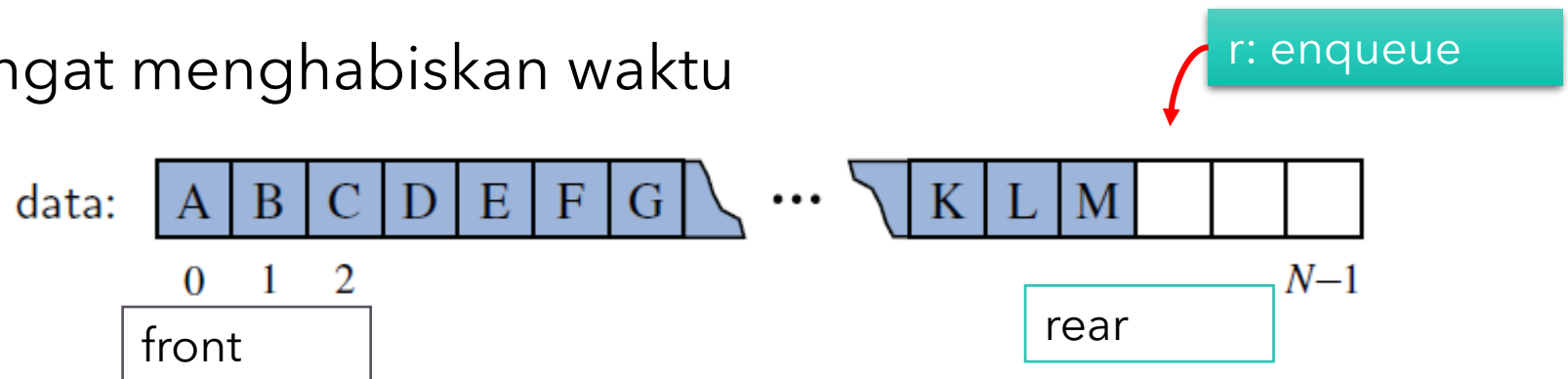
Queue Implementation

Implementasi Struktur Data



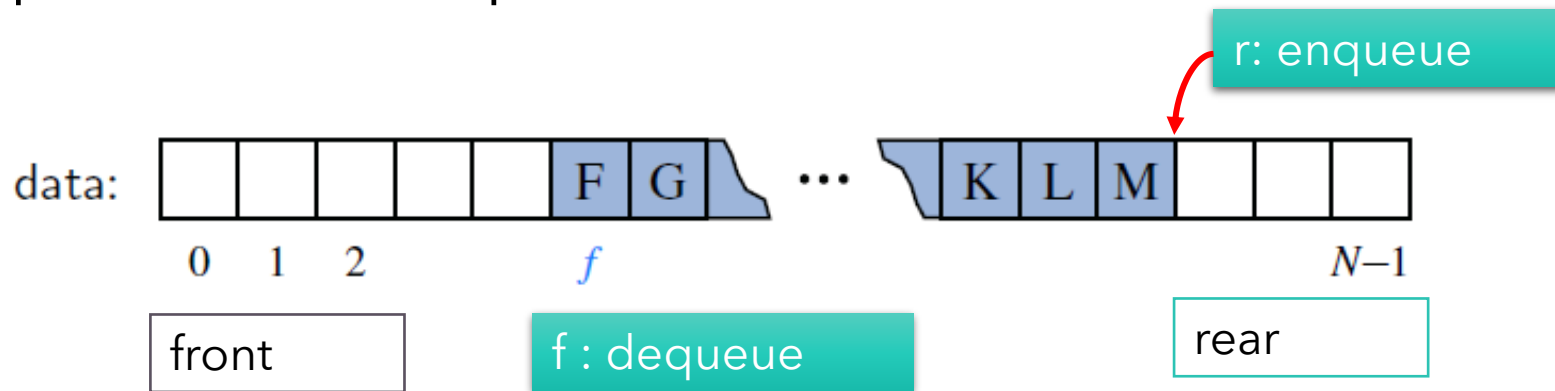
Array based

- Elemen disimpan dalam array, dengan elemen pertama disimpan pada indeks 0, elemen kedua pada indeks 1, dan seterusnya.
- Dengan demikian, enqueue dilakukan pada sisi rear
- Bagaimana cara melakukan dequeue?
 - Salah satu cara dapat dengan menggeser keseluruhan array saat dequeue
 - Tapi akan sangat menghabiskan waktu



Array based

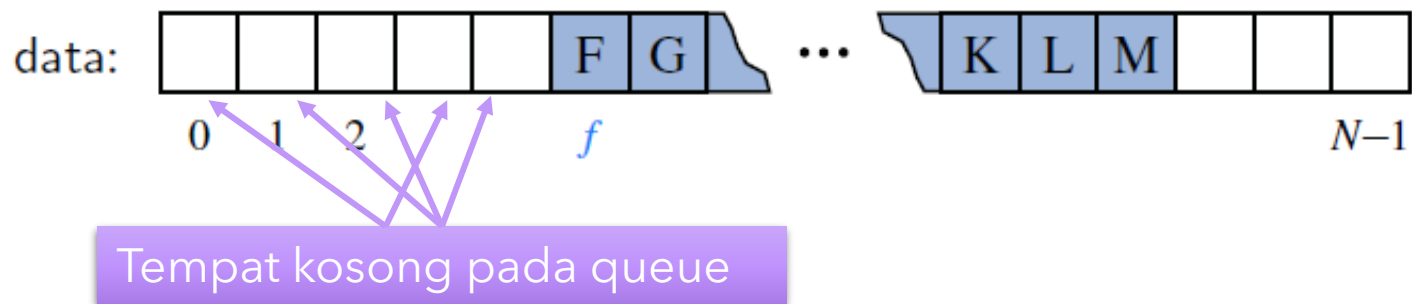
- Cara lain yang lebih efisien adalah dengan menggeser indeks front
 - Front tidak selalu berada di indeks 0
- Jadi, proses enqueue dilakukan pada sisi rear, dan proses dequeue dilakukan pada sisi front



Metode ini tetap menimbulkan masalah. Apakah itu?

Array based

- Bagaimana cara mengisi queue ketika rear telah mencapai indeks terakhir array, tetapi proses dequeue telah dilakukan?
- Dengan kata lain, sebenarnya masih ada tempat untuk mengisi antrian?

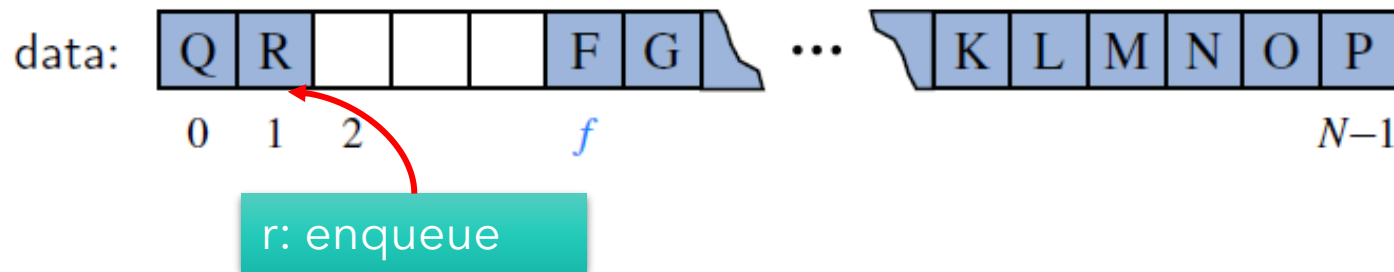


Circular Array - Queue

Pembuatan queue dengan menganggap ujung array yang satu "bertemu" dengan ujung array lainnya.

Enqueue dilakukan pada rear, dequeue pada front

Ketika telah mencapai ujung array, baik rear maupun front akan kembali lagi ke indeks 0



Circular array - Queue

Bagaimana implementasi dalam program?

- Menggunakan linear array berukuran N.
- Menggunakan modular arithmetic: ekspresi $X \% N$ menghasilkan nilai X tetap dalam range $0..N-1$.
 - Set Rear: tambahkan nilai rear lama dengan 1; jika $\text{rear} = N$, set dengan 0.
 - $\text{Rear} = (\text{Rear} + 1) \% N$;
 - Hal yang sama untuk Front: $\text{Front} = (\text{Front} + 1) \% N$;

Queue – Linked List

- Implementasi queue pada Linked List menggunakan Single Linked List
 1. deQueue → delete depan
 2. enQueue → insert belakang

Java collection - queue

- Queue merupakan interface dari Linked List
- Contoh implementasi:
- `Queue<String> stringQueue = new LinkedList<>();`
- **Method:**
 - `deQueue` - remove
 - `enqueue` - add
 - `first` - first

References

- Carrano, F., M., *Data Structures and Abstraction with Java*, 3rd Ed, Prentice Hall. (2012)
- Hortsman, C., *Big Java*, 4th Ed., John Wiley & Sons, Inc. (2010)
- Antonius Rachmat C, S.Kom, *STRUKTUR DATA (7) - single linked list circular*, Lecture slide