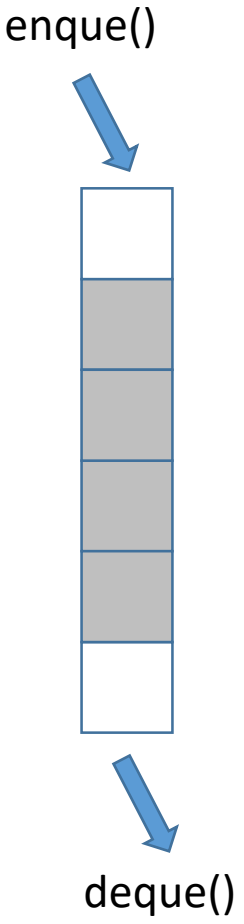
Kolejka

Kolejka (ang. *Queue*)

Struktura danych przechowująca obiekty, funkcjonująca na zasadzie **FIFO** (**F**irst **I**n **F**irst **O**ut).



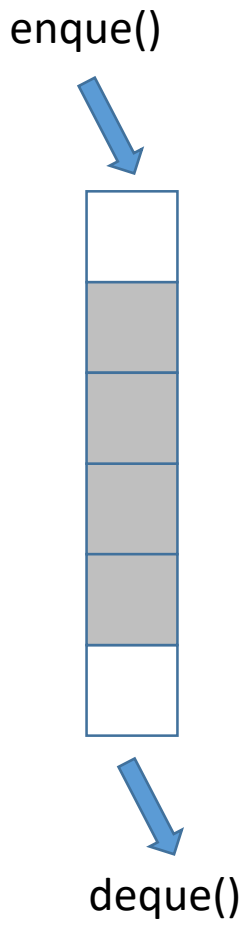
Analogia do „tradycyjnej” kolejki osób przed okienkiem pocztowym, samochodów przed punktem poboru opłat na autostradzie, itp.:

- Nowo przybyły osobnik ustawiany jest na koniec kolejki.
- Do obsługi podchodzi zawsze osobnik znajdujący się na początku kolejki (czyli najdłużej czekający w kolejce), tym samym opuszczając kolejkę.

Kolejka

Interfejs klasy **Kolejka**:

Opis	Metoda
Konstruktor, utworzenie pustej kolejki:	`Kolejka()`, inaczej: `Queue()`
Dodanie elementu do kolejki (na koniec):	`dodajDoKolejki()`, inaczej: `enqueue()`
Pobranie elementu z kolejki (z początku):	`pobierzZKolejki()`, inaczej: `dequeue()`
Liczba elementów w kolejce:	`rozmiar()`, inaczej: `size()`
Sprawdzenie, czy kolejka jest pusta:	`czyPusta()`, inaczej: `isEmpty()`



Kolejka

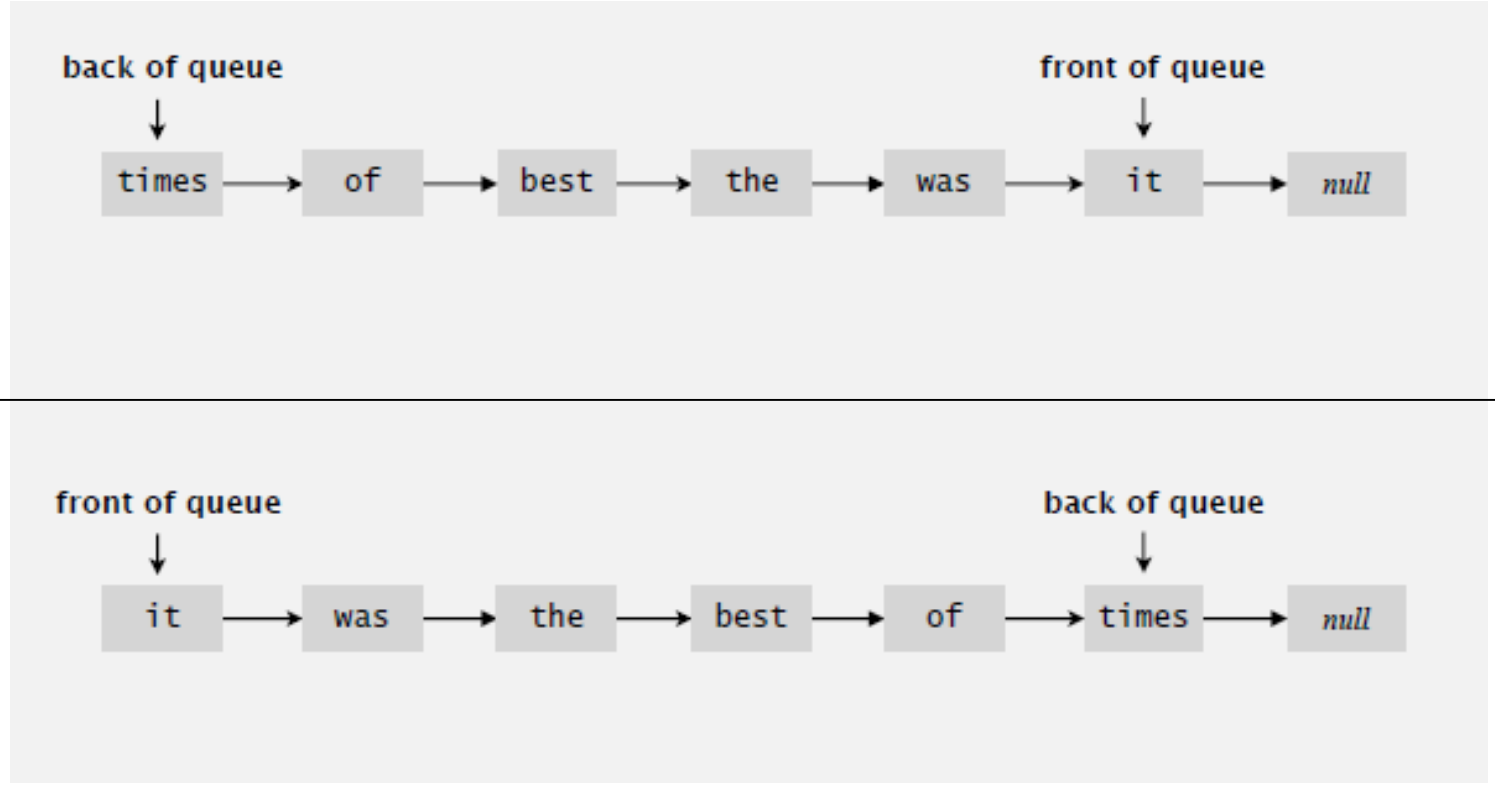
Implementacje:

- Za pomocą listy z dowiązaniami
- Za pomocą tablicy

Kolejka

Implementacja za pomocą listy

?



[SW-2018]

Kolejka

Implementacja za pomocą listy:

- Każdy element umieszczany w kolejce ma postać:

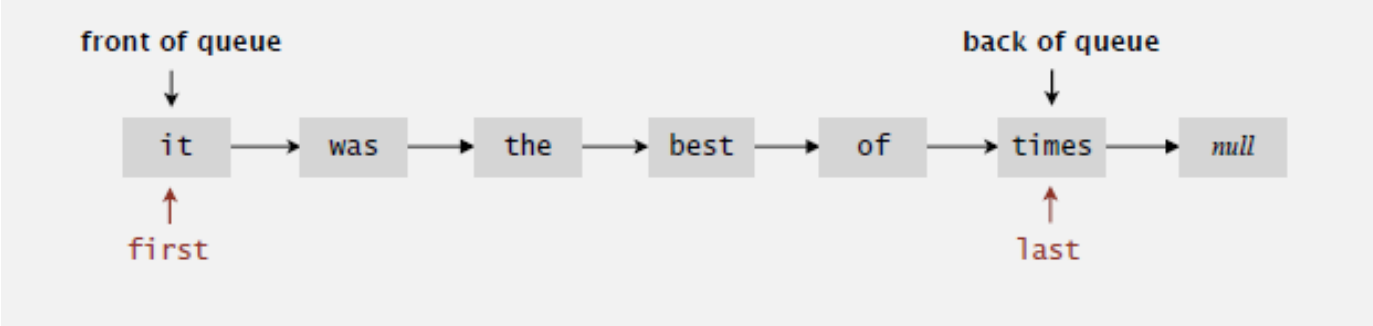
```
private class Node {
    Item item;
    Node next;
}
```

- Kolejka posiada:
  - głowę – **head** lub **first** (wskaźnik do pierwszego elementu) – początek kolejki
  - ogon – **tail** lub **last** (wskaźnik do elementu ostatniego) – koniec kolejki

Kolejka

Implementacja za pomocą listy:

- `enqueue()` – nowy element wstawiamy na koniec listy (za dotychczasowy **last**)
- `dequeue()` – zwracamy element z początku (wskazywanego dotychczas przez **first**, po wykonaniu operacji wskaźnik **first** będzie wskazywał na obecny drugi element)



[SW-2018]

Kolejka

Implementacja za pomocą listy:

```
public class Kolejka <Item> {
   private Node first;
   private Node last;
   private int N;

   private class Node {
      Item item;
      Node next;
   }
   …
}
```

Kolejka

Implementacja za pomocą listy:

- Interfejs:

```
Kolejka()
void enqueue(Item item)
Item dequeue()
int size()
boolean isEmpty()
```

```
public class Kolejka
{
   private Node first, last;

   private class Node {
      String item;
      Node next;
   }

   public boolean isEmpty() {
      return first == null;
   }

   public void enqueue(String item) {
      Node oldlast = last;
      last = new Node();
      last.item = item;
      last.next = null;
      if (isEmpty()) first = last;
      else oldlast.next = last;
   }

   public String dequeue() {
      String item = first.item;
      first = first.next;
      if (isEmpty()) last = null;
      return item;
   }
}
```

Kolejka

Implementacja za pomocą listy:

- Interfejs:

```
Kolejka()
void enqueue(Item item)
Item dequeue()
int size()
boolean isEmpty()
```

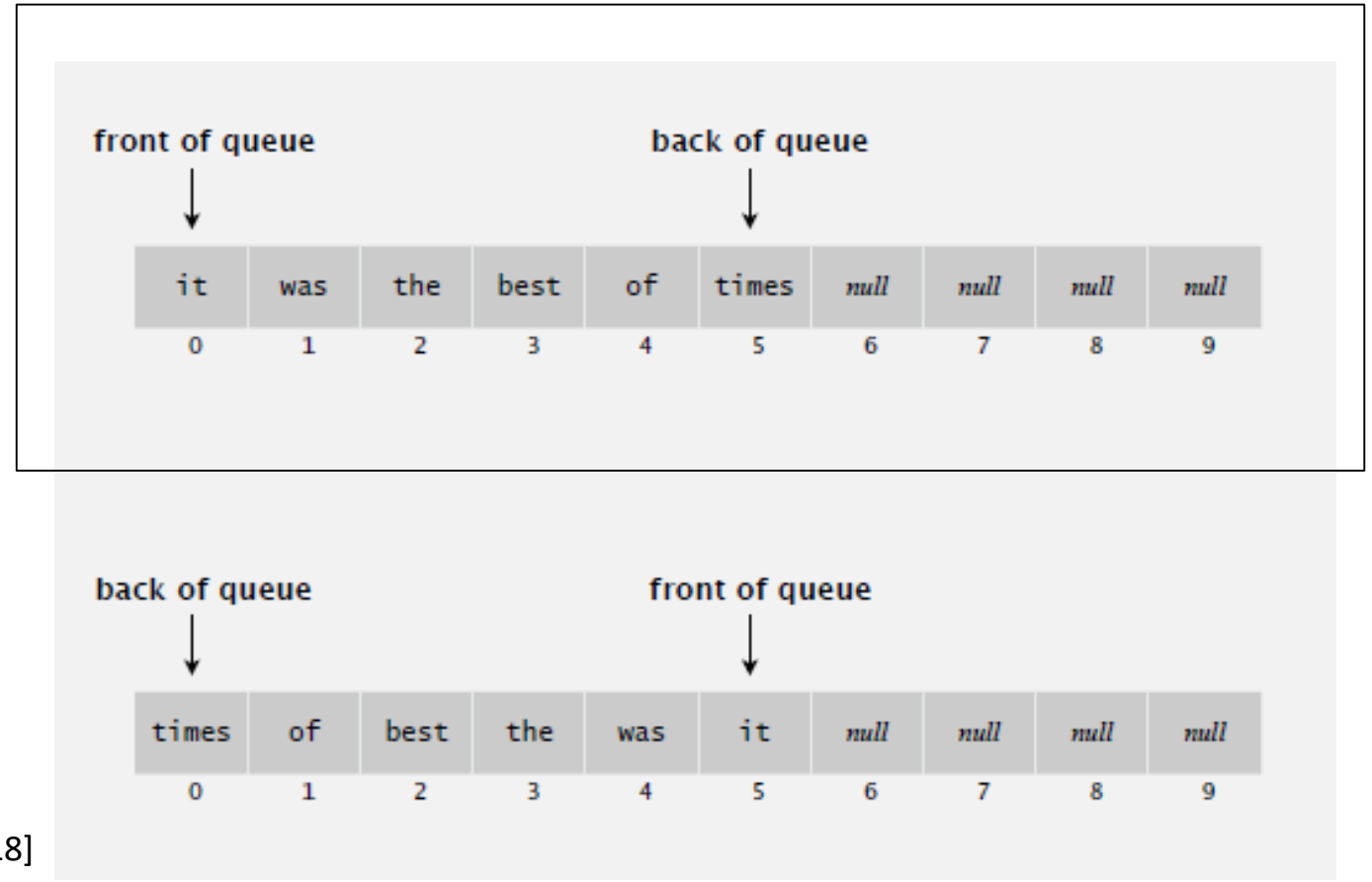
-> laboratorium

Kolejka

Implementacja za pomocą tablicy

?

- enqueue()
- dequeue()

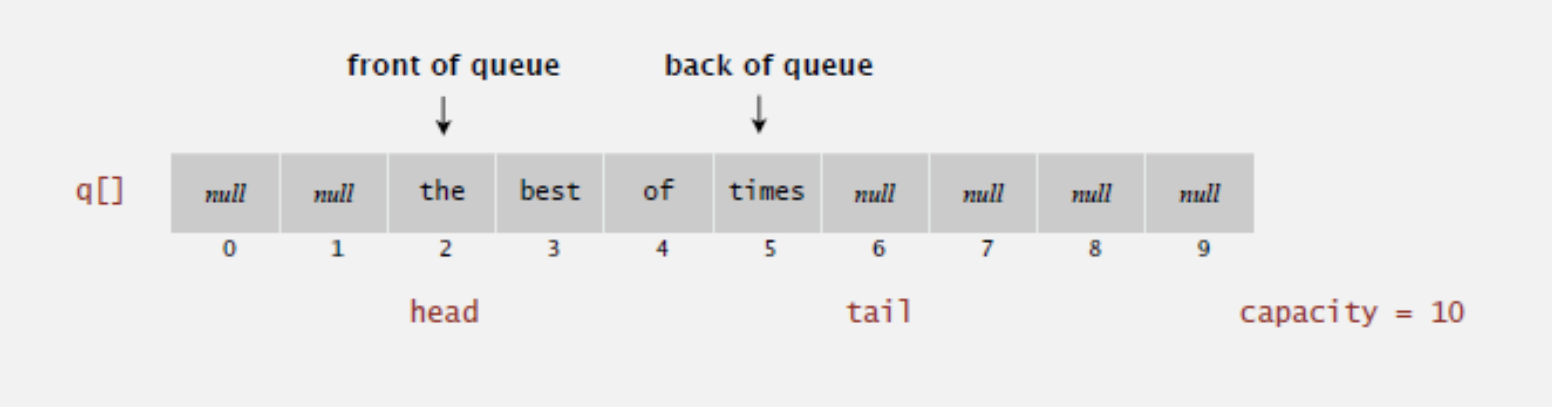


[SW-2018]

Kolejka

Implementacja za pomocą tablicy:

- Mamy tablicę Q[] do przechowywana elementów *N* elementów.
- enqueue(): dodaj nowy element na pozycję Q[tail] (Q[last]) (ew. tail+1).
- dequeue(): pobierz/usuń element z pozycji Q [head] (Q[first]).
- Zmodyfikuj head and tail.



[SW-2018]

Kolejka

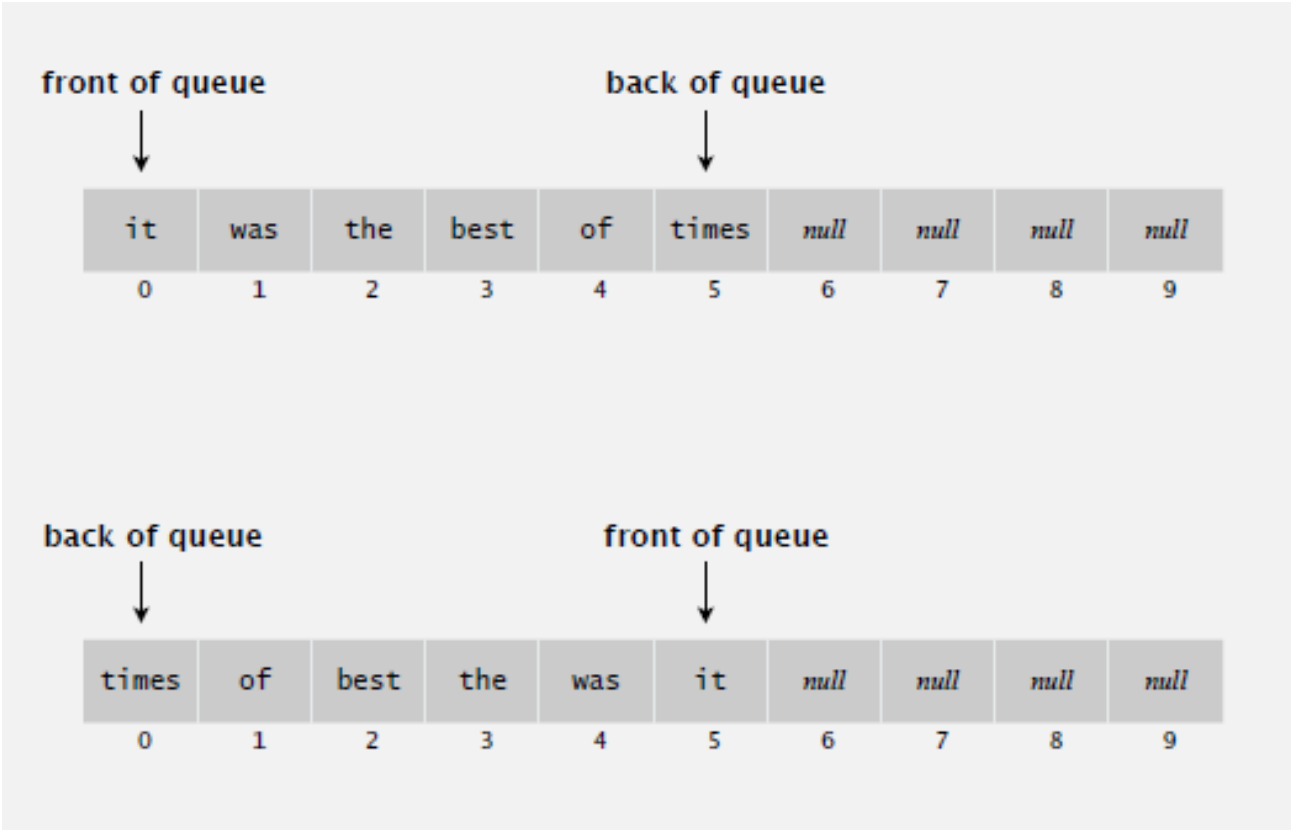
Implementacja za pomocą tablicy

Konieczność znajomości z góry rozmiaru kolejki,

Problemy:

- *overflow* – przy dodawaniu kolejnego elementu na listę
- *underflow* – przy pobieraniu elementów z kolejki

- Zmiana rozmiaru tablicy



[SW-2018]