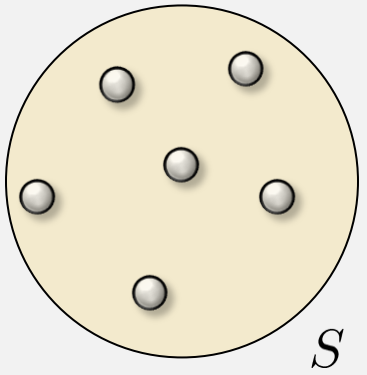
Συλλογές, Στοίβες και Ουρές

Σε πολλές εφαρμογές μας αρκεί η αναπαράσταση ενός δυναμικού συνόλου με μια δομή δεδομένων η οποία δεν υποστηρίζει την αναζήτηση οποιουδήποτε στοιχείου.

Συλλογή (bag) : Επιστρέφει ένα αυθαίρετο στοιχείο του συνόλου.

Στοίβα (stack) : Επιστρέφει το νεότερο στοιχείο του συνόλου.

Ουρά (queue) : Επιστρέφει το παλαιότερο στοιχείο του συνόλου.



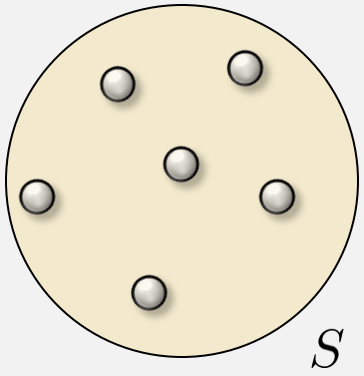
Ουρά προτεραιότητας (priority queue) : Επιστρέφει το στοιχείο με το μέγιστο (ή το ελάχιστο) κλειδί.

Συλλογή

Υποστηρίζει τις λειτουργίες:

εισαγωγή(S,x) : τοποθετεί το στοιχείο x στη συλλογή S

εξαγωγή(S) : αφαιρεί από την συλλογή S ένα αυθαίρετο στοιχείο της x και το επιστρέφει



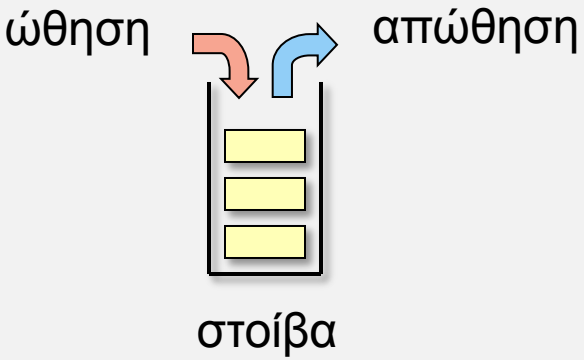
Μπορεί να υλοποιηθεί εύκολα με πίνακα ή συνδεδεμένη λίστα, έτσι ώστε οι λειτουργίες εισαγωγής και εξαγωγής να γίνονται σε O(1) χρόνο.

Στοίβα ώθησης προς τα κάτω

Υποστηρίζει τις λειτουργίες:

ώθηση(S,x) : τοποθετεί το στοιχείο x στην κορυφή της στοίβας S

απώθηση(S) : επιστρέφει το στοιχείο x που βρίσκεται στην κορυφή της στοίβας S και διαγράφει το x από την S



Στοίβα ώθησης προς τα κάτω

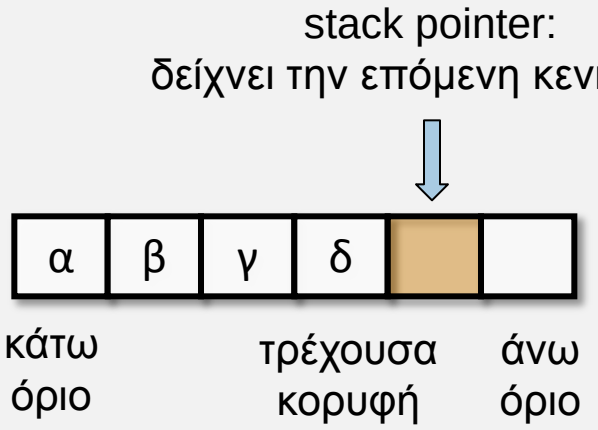
Διασύνδεση

```
class Stack                  // στοίβα αντικειμένων τύπου Item
{
      Stack(int)             // αρχικοποίηση στοίβας
      boolean isEmpty()      // έλεγχος αν η στοίβα είναι άδεια
      void push(Item)        // ώθηση αντικειμένου
      Item pop();            // απώθηση αντικειμένου
}
```

Στοίβα ώθησης προς τα κάτω

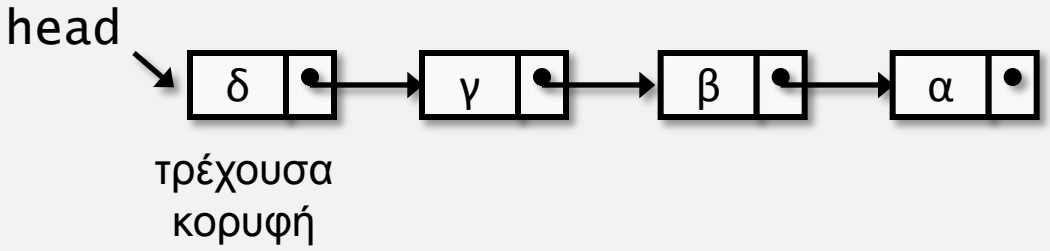
Υλοποίηση

- με πίνακα



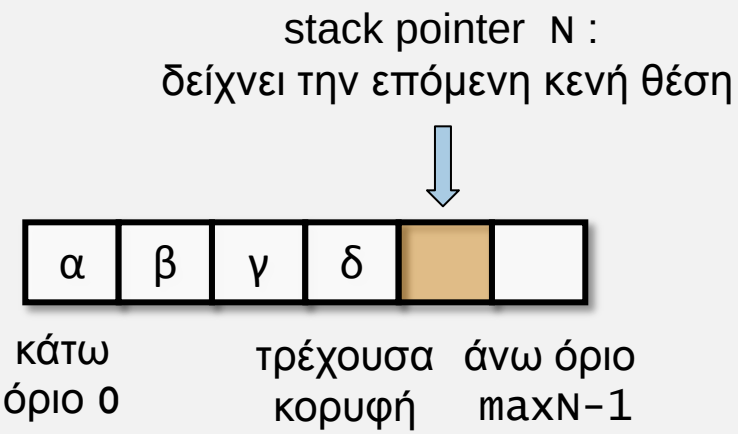

```
push(α)
push(β)
push(γ)
push(δ)
```

- με συνδεδεμένη λίστα



Στοίβα ώθησης προς τα κάτω

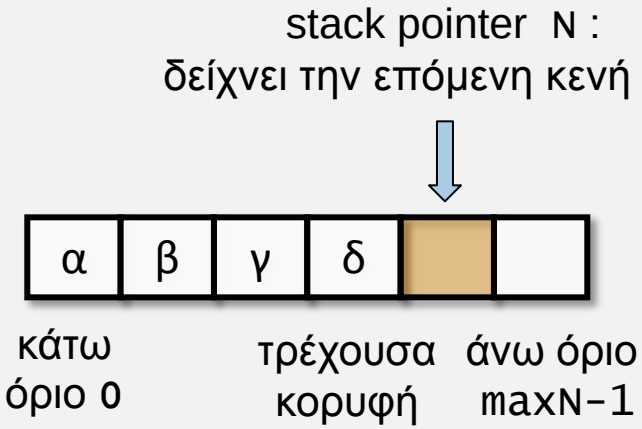
Υλοποίηση με πίνακα




```
class Stack {
      private Item[] S; private int N;

      Stack(int maxN) { S = new Item[maxN]; N=0; }

      boolean isEmpty() { return (N == 0); }

      void push(Item item) { S[N++] = item; }

      Item  pop() { return S[--N]; }
}
```

Στοίβα ώθησης προς τα κάτω

Υλοποίηση με πίνακα



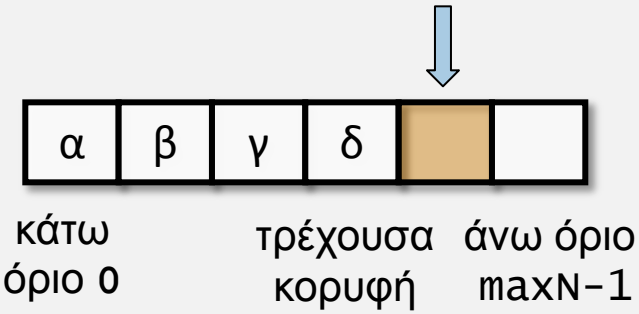

```
class Stack {
      private Item[] S; private int N;

      Stack(int maxN) { S = new Item[maxN]; N=0; }

      boolean isEmpty() { return (N == 0); }

      void push(Item item) { S[N++] = item; }

      Item  pop() { Item t = S[--N]; S[N] = null; return t; }
}
```

Χρειάζεται όταν ο πίνακας S αποθηκεύει αναφορές σε αντικείμενα, έτσι ώστε το σύστημα της Java να γνωρίζει ότι η αντίστοιχη μνήμη μπορεί να απελευθερωθεί.

Στοίβα ώθησης προς τα κάτω

Υλοποίηση με πίνακα

stack pointer N :
δείχνει την επόμενη κενή θέση



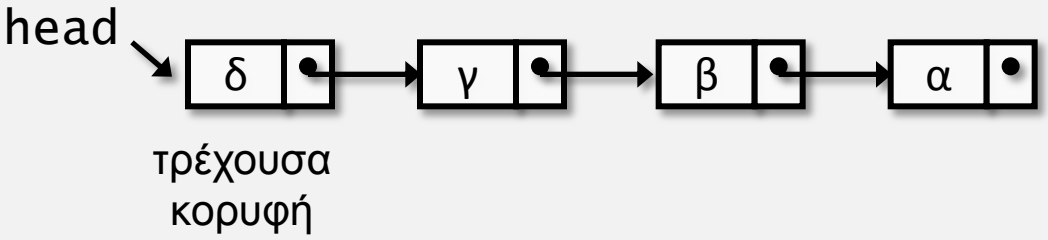
Τι γίνεται όμως αν N == maxN ;

```
class Stack {
      private Item[] S; private int N;

      Stack(int maxN) { S = new Item[maxN]; N=0; }

      boolean isEmpty() { return (N == 0); }

      void push(Item item) { S[N++] = item; }

      Item  pop() { Item t = S[--N]; S[N] = null; return t; }
}
```

Χρειάζεται όταν ο πίνακας S αποθηκεύει αναφορές σε αντικείμενα, έτσι ώστε το σύστημα της Java να γνωρίζει ότι η αντίστοιχη μνήμη μπορεί να απελευθερωθεί.

Στοίβα ώθησης προς τα κάτω

Υλοποίηση με συνδεδεμένη λίστα

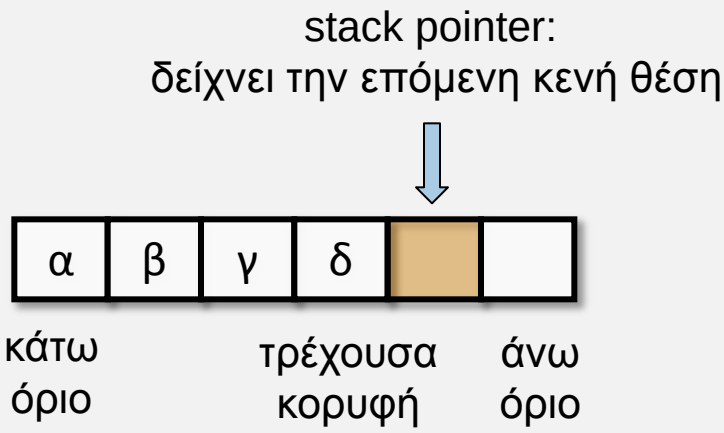



```
class Stack {
        private Node head;
        private class Node {
                Item item; Node next;
                Node(Item item, Node next)
                { this.item = item; this.next = next; }
        }

        Stack(int maxN) { head = null; }

        boolean isEmpty() { return head == null; }

        void push(Item item) { head = new Node(item, head); }

        Item  pop() { Item item = head.item;  Node t = head.next;
                      head = t; return item; }
}
```

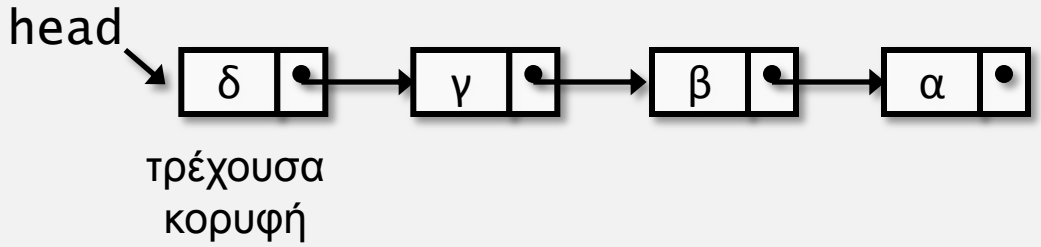
Στοίβα ώθησης προς τα κάτω

Υλοποίηση

- με πίνακα



- με συνδεδεμένη λίστα



- Και οι δύο υλοποιήσεις επιτυγχάνουν O(1) χρόνο ανά λειτουργία.
- Η υλοποίηση με πίνακα είναι πιο γρήγορη στην πράξη.
- Η υλοποίηση με πίνακα έχει το μειονέκτημα ότι πρέπει να οριστεί ο μέγιστος αριθμός στοιχείων (maxN) στη στοίβα. Το αν αυτό αποτελεί σημαντικό πρόβλημα ή όχι εξαρτάται από την εφαρμογή.

Στοίβα ώθησης προς τα κάτω

Παράδειγμα: Υπολογισμός απλών αριθμητικών παραστάσεων

ενθεματική αναπαράσταση $5 * (((9 + 8) * (4 * 6)) + 7)$

μεταθεματική αναπαράσταση $5\ 9\ 8\ +\ 4\ 6\ *\ *\ 7\ +\ *$

Στοίβα ώθησης προς τα κάτω

Παράδειγμα: Υπολογισμός απλών αριθμητικών παραστάσεων

μεταθεματική αναπαράσταση ⇨ ενθεματική αναπαράσταση

Αντικαθιστούμε $a\ b \odot$ με $(a \odot b)$ όπου $\odot$ δυαδικός τελεστής

Στοίβα ώθησης προς τα κάτω

Παράδειγμα: Υπολογισμός απλών αριθμητικών παραστάσεων

μεταθεματική αναπαράσταση ⇒ ενθεματική αναπαράσταση

Αντικαθιστούμε $a\ b\ \odot$ με $(a \odot b)$ όπου $\odot$ δυαδικός τελεστής

$5\ 9\ 8\ +\ 4\ 6\ *\ *\ 7\ +\ *$ ⇒

$5\ (9+8)\ (4*6)\ *\ 7\ +\ *$ ⇒

$5\ ((9+8)*(4*6))\ 7\ +\ *$ ⇒

$5\ (((9+8)*(4*6))+7)\ *$ ⇒

$(5*(((9+8)*(4*6))+7))$

Στοίβα ώθησης προς τα κάτω

Παράδειγμα: Υπολογισμός απλών αριθμητικών παραστάσεων

Υπολογισμός μεταθεματικής αναπαράστασης με στοίβα S

Αν το επόμενο σύμβολο είναι αριθμός x τότε εκτελούμε `S.push(x)`

Διαφορετικά, αν είναι τελεστής $\odot$, εκτελούμε

```
x = S.pop();
y = S.pop();
z = x ⊙ y;
S.push(z);
```

Στοίβα ώθησης προς τα κάτω

Παράδειγμα: Υπολογισμός απλών αριθμητικών παραστάσεων

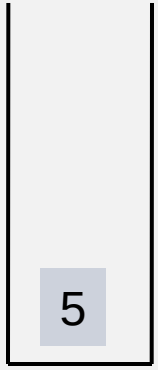
Υπολογισμός μεταθεματικής αναπαράστασης με στοίβα S

Αν το επόμενο σύμβολο είναι αριθμός x τότε εκτελούμε `S.push(x)`

Διαφορετικά, αν είναι τελεστής $\odot$, εκτελούμε

```
x = S.pop();
y = S.pop();
z = x ⊙ y;
S.push(z);
```

$5\ 9\ 8\ +\ 4\ 6\ *\ *\ 7\ +\ *$



στοίβα S

Στοίβα ώθησης προς τα κάτω

Παράδειγμα: Υπολογισμός απλών αριθμητικών παραστάσεων

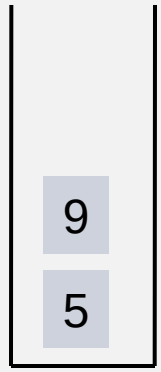
Υπολογισμός μεταθεματικής αναπαράστασης με στοίβα S

Αν το επόμενο σύμβολο είναι αριθμός x τότε εκτελούμε S.push(x)

Διαφορετικά, αν είναι τελεστής $\odot$, εκτελούμε

```
x = S.pop();
y = S.pop();
z = x ⊙ y;
S.push(z);
```

$$5\ 9\ 8\ +\ 4\ 6\ *\ *\ 7\ +\ *$$



στοίβα S

Στοίβα ώθησης προς τα κάτω

Παράδειγμα: Υπολογισμός απλών αριθμητικών παραστάσεων

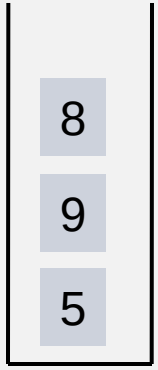
Υπολογισμός μεταθεματικής αναπαράστασης με στοίβα S

Αν το επόμενο σύμβολο είναι αριθμός x τότε εκτελούμε S.push(x)

Διαφορετικά, αν είναι τελεστής $\odot$, εκτελούμε

```
x = S.pop();
y = S.pop();
z = x ⊙ y;
S.push(z);
```

$5\ 9\ 8\ +\ 4\ 6\ *\ *\ 7\ +\ *$



στοίβα S

Στοίβα ώθησης προς τα κάτω

Παράδειγμα: Υπολογισμός απλών αριθμητικών παραστάσεων

Υπολογισμός μεταθεματικής αναπαράστασης με στοίβα S

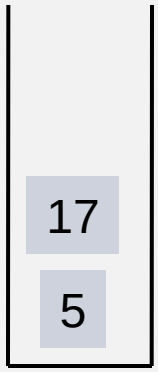
Αν το επόμενο σύμβολο είναι αριθμός x τότε εκτελούμε S.push(x)

Διαφορετικά, αν είναι τελεστής $\odot$, εκτελούμε

```
x = S.pop();
y = S.pop();
z = x ⊙ y;
S.push(z);
```

$$5\ 9\ 8\ +\ 4\ 6\ *\ *\ 7\ +\ *$$

8 + 9 = 17



στοίβα S

Στοίβα ώθησης προς τα κάτω

Παράδειγμα: Υπολογισμός απλών αριθμητικών παραστάσεων

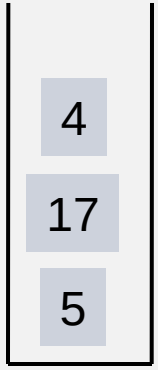
Υπολογισμός μεταθεματικής αναπαράστασης με στοίβα S

Αν το επόμενο σύμβολο είναι αριθμός x τότε εκτελούμε S.push(x)

Διαφορετικά, αν είναι τελεστής $\odot$, εκτελούμε

```
x = S.pop();
y = S.pop();
z = x ⊙ y;
S.push(z);
```

$$5\ 9\ 8\ +\ 4\ 6\ *\ *\ 7\ +\ *$$



στοίβα S

Στοίβα ώθησης προς τα κάτω

Παράδειγμα: Υπολογισμός απλών αριθμητικών παραστάσεων

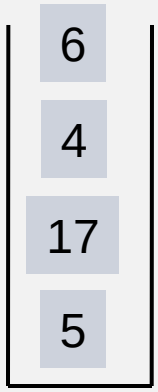
Υπολογισμός μεταθεματικής αναπαράστασης με στοίβα S

Αν το επόμενο σύμβολο είναι αριθμός x τότε εκτελούμε `S.push(x)`

Διαφορετικά, αν είναι τελεστής $\odot$, εκτελούμε

```
x = S.pop();
y = S.pop();
z = x ⊙ y;
S.push(z);
```

$$5\ 9\ 8\ +\ 4\ 6\ *\ *\ 7\ +\ *$$



στοίβα S

Στοίβα ώθησης προς τα κάτω

Παράδειγμα: Υπολογισμός απλών αριθμητικών παραστάσεων

Υπολογισμός μεταθεματικής αναπαράστασης με στοίβα S

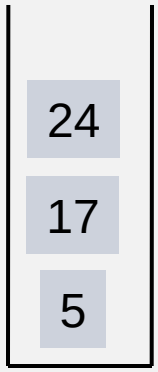
Αν το επόμενο σύμβολο είναι αριθμός x τότε εκτελούμε `S.push(x)`

Διαφορετικά, αν είναι τελεστής $\odot$, εκτελούμε

```
x = S.pop();
y = S.pop();
z = x ⊙ y;
S.push(z);
```

$$5\ 9\ 8\ +\ 4\ 6\ *\ *\ 7\ +\ *$$

6*4 =24



στοίβα S

Στοίβα ώθησης προς τα κάτω

Παράδειγμα: Υπολογισμός απλών αριθμητικών παραστάσεων

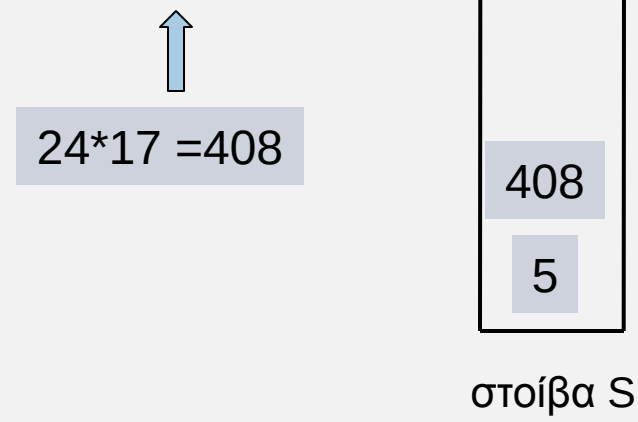
Υπολογισμός μεταθεματικής αναπαράστασης με στοίβα S

Αν το επόμενο σύμβολο είναι αριθμός x τότε εκτελούμε `S.push(x)`

Διαφορετικά, αν είναι τελεστής $\odot$, εκτελούμε

```
x = S.pop();
y = S.pop();
z = x ⊙ y;
S.push(z);
```

$$5\ 9\ 8\ +\ 4\ 6\ *\ *\ 7\ +\ *$$



Στοίβα ώθησης προς τα κάτω

Παράδειγμα: Υπολογισμός απλών αριθμητικών παραστάσεων

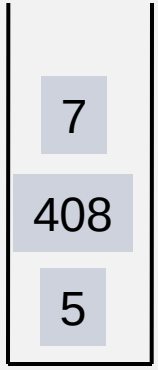
Υπολογισμός μεταθεματικής αναπαράστασης με στοίβα S

Αν το επόμενο σύμβολο είναι αριθμός x τότε εκτελούμε S.push(x)

Διαφορετικά, αν είναι τελεστής $\odot$, εκτελούμε

```
x = S.pop();
y = S.pop();
z = x ⊙ y;
S.push(z);
```

$$5\ 9\ 8\ +\ 4\ 6\ *\ *\ 7\ +\ *$$



στοίβα S

Στοίβα ώθησης προς τα κάτω

Παράδειγμα: Υπολογισμός απλών αριθμητικών παραστάσεων

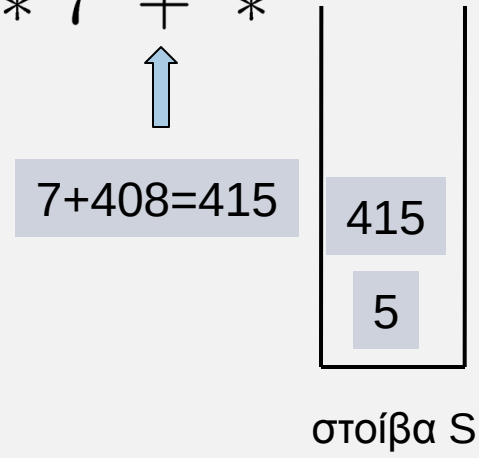
Υπολογισμός μεταθεματικής αναπαράστασης με στοίβα S

Αν το επόμενο σύμβολο είναι αριθμός x τότε εκτελούμε `S.push(x)`

Διαφορετικά, αν είναι τελεστής $\odot$, εκτελούμε

```
x = S.pop();
y = S.pop();
z = x ⊙ y;
S.push(z);
```

$$5\ 9\ 8\ +\ 4\ 6\ *\ *\ 7\ +\ *$$



Στοίβα ώθησης προς τα κάτω

Παράδειγμα: Υπολογισμός απλών αριθμητικών παραστάσεων

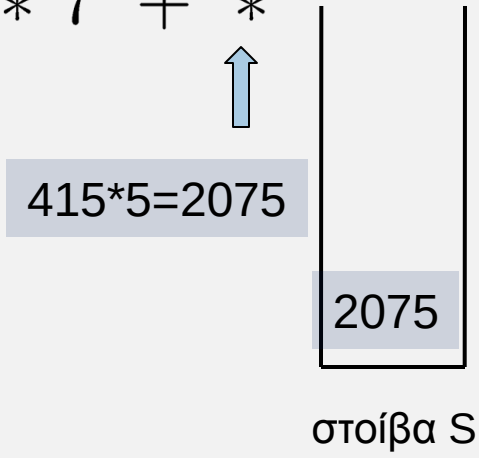
Υπολογισμός μεταθεματικής αναπαράστασης με στοίβα S

Αν το επόμενο σύμβολο είναι αριθμός x τότε εκτελούμε `S.push(x)`

Διαφορετικά, αν είναι τελεστής $\odot$, εκτελούμε

```
x = S.pop();
y = S.pop();
z = x ⊙ y;
S.push(z);
```

$$5\ 9\ 8\ +\ 4\ 6\ *\ *\ 7\ +\ *$$



Στοίβα ώθησης προς τα κάτω

Παράδειγμα: Υπολογισμός απλών αριθμητικών παραστάσεων

Υπολογισμός μεταθεματικής αναπαράστασης με στοίβα S (τύπου int)

```
public static void main(String[] args){
        char[] a = args[0].toCharArray();
        int N = a.length;
        Stack S = new Stack(N);
        for (i = 0; i < N; i++)
        {
                if (a[i] == '+') S.push(S.pop()+S.pop());
                if (a[i] == '*') S.push(S.pop()*S.pop());
                if ((a[i] >= '0') && (a[i] <= '9') S.push(0);
                while ((a[i] >= '0') && (a[i] <= '9'))
                        S.push(10*S.pop() + (a[i++]-'0'));
        }
        System.out.println(S.pop() + "");
}
```

Στοίβα ώθησης προς τα κάτω

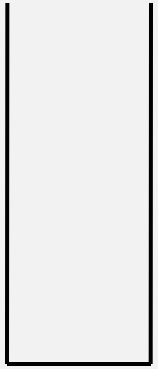
Παράδειγμα: Υπολογισμός απλών αριθμητικών παραστάσεων

ενθεματική αναπαράσταση ⇒ μεταθεματική αναπαράσταση

Χρησιμοποιούμε στοίβα χαρακτήρων S. Αν το επόμενο σύμβολο είναι:

- αριθμός : τον τυπώνουμε
- '(' : δεν κάνουμε τίποτα
- ')' : εκτελούμε `S.pop()` και τυπώνουμε το σύμβολο που λάβαμε
- τελεστής : εκτελούμε `S.push()`

$$(5 * (((9 + 8) * (4 * 6)) + 7))$$



στοίβα S

Στοίβα ώθησης προς τα κάτω

Παράδειγμα: Υπολογισμός απλών αριθμητικών παραστάσεων

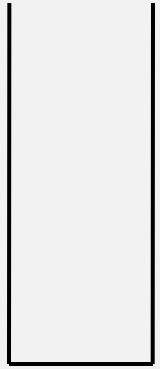
ενθεματική αναπαράσταση ⇨ μεταθεματική αναπαράσταση

Χρησιμοποιούμε στοίβα χαρακτήρων S. Αν το επόμενο σύμβολο είναι:

- αριθμός : τον τυπώνουμε
- ‘(’ : δεν κάνουμε τίποτα
- ‘)’ : εκτελούμε `S.pop()` και τυπώνουμε το σύμβολο που λάβαμε
- τελεστής : εκτελούμε `S.push()`

$$(5 * (((9 + 8) * (4 * 6)) + 7))$$

5



στοίβα S

Στοίβα ώθησης προς τα κάτω

Παράδειγμα: Υπολογισμός απλών αριθμητικών παραστάσεων

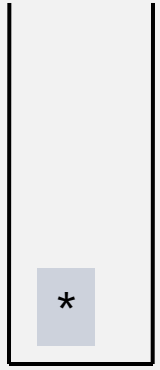
ενθεματική αναπαράσταση ⇒ μεταθεματική αναπαράσταση

Χρησιμοποιούμε στοίβα χαρακτήρων S. Αν το επόμενο σύμβολο είναι:

- αριθμός : τον τυπώνουμε
- '(' : δεν κάνουμε τίποτα
- ')' : εκτελούμε `S.pop()` και τυπώνουμε το σύμβολο που λάβαμε
- τελεστής : εκτελούμε `S.push()`

$$(5 * (((9 + 8) * (4 * 6)) + 7))$$

5



στοίβα S

Στοίβα ώθησης προς τα κάτω

Παράδειγμα: Υπολογισμός απλών αριθμητικών παραστάσεων

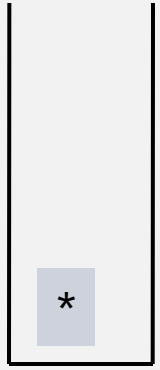
ενθεματική αναπαράσταση ⇒ μεταθεματική αναπαράσταση

Χρησιμοποιούμε στοίβα χαρακτήρων S. Αν το επόμενο σύμβολο είναι:

- αριθμός : τον τυπώνουμε
- '(' : δεν κάνουμε τίποτα
- ')' : εκτελούμε `S.pop()` και τυπώνουμε το σύμβολο που λάβαμε
- τελεστής : εκτελούμε `S.push()`

$$(5 * (((9 + 8) * (4 * 6)) + 7))$$

5



στοίβα S

Στοίβα ώθησης προς τα κάτω

Παράδειγμα: Υπολογισμός απλών αριθμητικών παραστάσεων

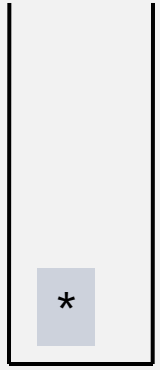
ενθεματική αναπαράσταση ⇒ μεταθεματική αναπαράσταση

Χρησιμοποιούμε στοίβα χαρακτήρων S. Αν το επόμενο σύμβολο είναι:

- αριθμός : τον τυπώνουμε
- '(' : δεν κάνουμε τίποτα
- ')' : εκτελούμε `S.pop()` και τυπώνουμε το σύμβολο που λάβαμε
- τελεστής : εκτελούμε `S.push()`

$$(5 * (((9 + 8) * (4 * 6)) + 7))$$

5



στοίβα S

Στοίβα ώθησης προς τα κάτω

Παράδειγμα: Υπολογισμός απλών αριθμητικών παραστάσεων

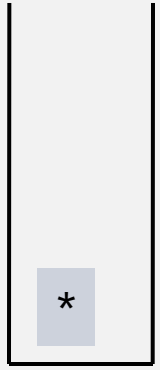
ενθεματική αναπαράσταση ⇨ μεταθεματική αναπαράσταση

Χρησιμοποιούμε στοίβα χαρακτήρων S. Αν το επόμενο σύμβολο είναι:

- αριθμός : τον τυπώνουμε
- '(' : δεν κάνουμε τίποτα
- ')' : εκτελούμε `S.pop()` και τυπώνουμε το σύμβολο που λάβαμε
- τελεστής : εκτελούμε `S.push()`

$$(5 * (((9 + 8) * (4 * 6)) + 7))$$

5



στοίβα S

Στοίβα ώθησης προς τα κάτω

Παράδειγμα: Υπολογισμός απλών αριθμητικών παραστάσεων

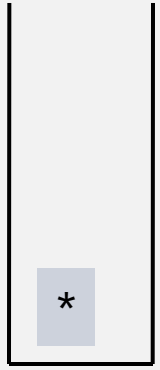
ενθεματική αναπαράσταση ⇒ μεταθεματική αναπαράσταση

Χρησιμοποιούμε στοίβα χαρακτήρων S. Αν το επόμενο σύμβολο είναι:

- αριθμός : τον τυπώνουμε
- '(' : δεν κάνουμε τίποτα
- ')' : εκτελούμε `S.pop()` και τυπώνουμε το σύμβολο που λάβαμε
- τελεστής : εκτελούμε `S.push()`

$$(5 * (((9 + 8) * (4 * 6)) + 7))$$

5 9



στοίβα S

Στοίβα ώθησης προς τα κάτω

Παράδειγμα: Υπολογισμός απλών αριθμητικών παραστάσεων

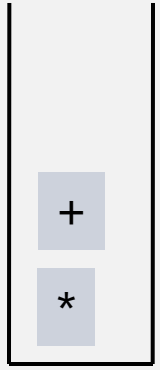
ενθεματική αναπαράσταση ⇒ μεταθεματική αναπαράσταση

Χρησιμοποιούμε στοίβα χαρακτήρων S. Αν το επόμενο σύμβολο είναι:

- αριθμός : τον τυπώνουμε
- '(' : δεν κάνουμε τίποτα
- ')' : εκτελούμε `S.pop()` και τυπώνουμε το σύμβολο που λάβαμε
- τελεστής : εκτελούμε `S.push()`

$$(5 * (((9 + 8) * (4 * 6)) + 7))$$

5 9



στοίβα S

Στοίβα ώθησης προς τα κάτω

Παράδειγμα: Υπολογισμός απλών αριθμητικών παραστάσεων

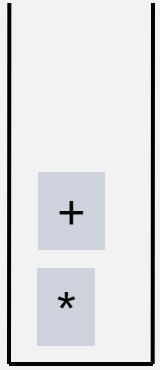
ενθεματική αναπαράσταση ⇒ μεταθεματική αναπαράσταση

Χρησιμοποιούμε στοίβα χαρακτήρων S. Αν το επόμενο σύμβολο είναι:

- αριθμός : τον τυπώνουμε
- '(' : δεν κάνουμε τίποτα
- ')' : εκτελούμε `S.pop()` και τυπώνουμε το σύμβολο που λάβαμε
- τελεστής : εκτελούμε `S.push()`

$$(5 * (((9 + 8) * (4 * 6)) + 7))$$

5 9 8



στοίβα S

Στοίβα ώθησης προς τα κάτω

Παράδειγμα: Υπολογισμός απλών αριθμητικών παραστάσεων

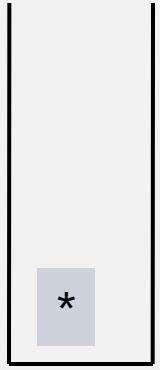
ενθεματική αναπαράσταση ⇒ μεταθεματική αναπαράσταση

Χρησιμοποιούμε στοίβα χαρακτήρων S. Αν το επόμενο σύμβολο είναι:

- αριθμός : τον τυπώνουμε
- '(' : δεν κάνουμε τίποτα
- ')' : εκτελούμε `S.pop()` και τυπώνουμε το σύμβολο που λάβαμε
- τελεστής : εκτελούμε `S.push()`

$$(5 * (((9 + 8) * (4 * 6)) + 7))$$

5 9 8 +



στοίβα S

Στοίβα ώθησης προς τα κάτω

Παράδειγμα: Υπολογισμός απλών αριθμητικών παραστάσεων

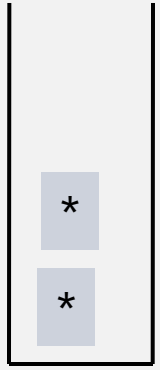
ενθεματική αναπαράσταση ⇒ μεταθεματική αναπαράσταση

Χρησιμοποιούμε στοίβα χαρακτήρων S. Αν το επόμενο σύμβολο είναι:

- αριθμός : τον τυπώνουμε
- '(' : δεν κάνουμε τίποτα
- ')' : εκτελούμε `S.pop()` και τυπώνουμε το σύμβολο που λάβαμε
- τελεστής : εκτελούμε `S.push()`

$$(5 * (((9 + 8) * (4 * 6)) + 7))$$

5 9 8 +



στοίβα S

Στοίβα ώθησης προς τα κάτω

Παράδειγμα: Υπολογισμός απλών αριθμητικών παραστάσεων

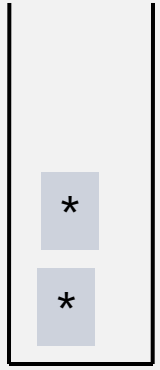
ενθεματική αναπαράσταση ⇒ μεταθεματική αναπαράσταση

Χρησιμοποιούμε στοίβα χαρακτήρων S. Αν το επόμενο σύμβολο είναι:

- αριθμός : τον τυπώνουμε
- '(' : δεν κάνουμε τίποτα
- ')' : εκτελούμε `S.pop()` και τυπώνουμε το σύμβολο που λάβαμε
- τελεστής : εκτελούμε `S.push()`

$$(5 * (((9 + 8) * (4 * 6)) + 7))$$

5 9 8 +



στοίβα S

Στοίβα ώθησης προς τα κάτω

Παράδειγμα: Υπολογισμός απλών αριθμητικών παραστάσεων

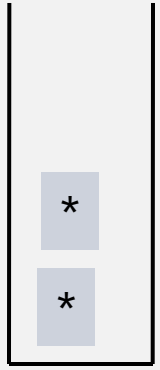
ενθεματική αναπαράσταση ⇒ μεταθεματική αναπαράσταση

Χρησιμοποιούμε στοίβα χαρακτήρων S. Αν το επόμενο σύμβολο είναι:

- αριθμός : τον τυπώνουμε
- '(' : δεν κάνουμε τίποτα
- ')' : εκτελούμε `S.pop()` και τυπώνουμε το σύμβολο που λάβαμε
- τελεστής : εκτελούμε `S.push()`

$$(5 * (((9 + 8) * (4 * 6)) + 7))$$

5 9 8 + 4



στοίβα S

Στοίβα ώθησης προς τα κάτω

Παράδειγμα: Υπολογισμός απλών αριθμητικών παραστάσεων

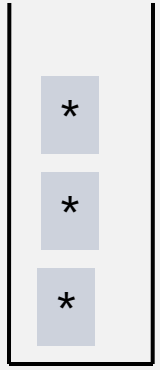
ενθεματική αναπαράσταση ⇒ μεταθεματική αναπαράσταση

Χρησιμοποιούμε στοίβα χαρακτήρων S. Αν το επόμενο σύμβολο είναι:

- αριθμός : τον τυπώνουμε
- '(' : δεν κάνουμε τίποτα
- ')' : εκτελούμε `S.pop()` και τυπώνουμε το σύμβολο που λάβαμε
- τελεστής : εκτελούμε `S.push()`

$$(5 * (((9 + 8) * (4 * 6)) + 7))$$

5 9 8 + 4



στοίβα S

Στοίβα ώθησης προς τα κάτω

Παράδειγμα: Υπολογισμός απλών αριθμητικών παραστάσεων

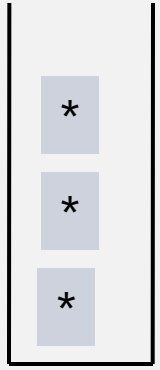
ενθεματική αναπαράσταση ⇒ μεταθεματική αναπαράσταση

Χρησιμοποιούμε στοίβα χαρακτήρων S. Αν το επόμενο σύμβολο είναι:

- αριθμός : τον τυπώνουμε
- '(' : δεν κάνουμε τίποτα
- ')' : εκτελούμε `S.pop()` και τυπώνουμε το σύμβολο που λάβαμε
- τελεστής : εκτελούμε `S.push()`

$$(5 * (((9 + 8) * (4 * 6)) + 7))$$

5 9 8 + 4 6



στοίβα S

Στοίβα ώθησης προς τα κάτω

Παράδειγμα: Υπολογισμός απλών αριθμητικών παραστάσεων

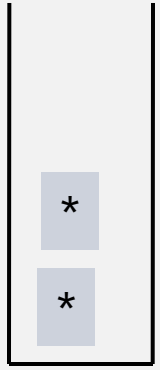
ενθεματική αναπαράσταση ⇒ μεταθεματική αναπαράσταση

Χρησιμοποιούμε στοίβα χαρακτήρων S. Αν το επόμενο σύμβολο είναι:

- αριθμός : τον τυπώνουμε
- '(' : δεν κάνουμε τίποτα
- ')' : εκτελούμε `S.pop()` και τυπώνουμε το σύμβολο που λάβαμε
- τελεστής : εκτελούμε `S.push()`

$$(5 * (((9 + 8) * (4 * 6)) + 7))$$

5 9 8 + 4 6 *



στοίβα S

Στοίβα ώθησης προς τα κάτω

Παράδειγμα: Υπολογισμός απλών αριθμητικών παραστάσεων

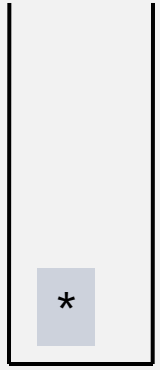
ενθεματική αναπαράσταση ⇨ μεταθεματική αναπαράσταση

Χρησιμοποιούμε στοίβα χαρακτήρων S. Αν το επόμενο σύμβολο είναι:

- αριθμός : τον τυπώνουμε
- '(' : δεν κάνουμε τίποτα
- ')' : εκτελούμε `S.pop()` και τυπώνουμε το σύμβολο που λάβαμε
- τελεστής : εκτελούμε `S.push()`

$$(5 * (((9 + 8) * (4 * 6)) + 7))$$

5 9 8 + 4 6 * *



στοίβα S

Στοίβα ώθησης προς τα κάτω

Παράδειγμα: Υπολογισμός απλών αριθμητικών παραστάσεων

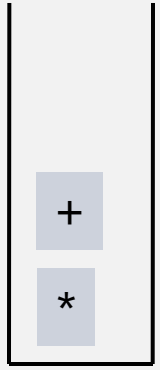
ενθεματική αναπαράσταση ⇒ μεταθεματική αναπαράσταση

Χρησιμοποιούμε στοίβα χαρακτήρων S. Αν το επόμενο σύμβολο είναι:

- αριθμός : τον τυπώνουμε
- '(' : δεν κάνουμε τίποτα
- ')' : εκτελούμε `S.pop()` και τυπώνουμε το σύμβολο που λάβαμε
- τελεστής : εκτελούμε `S.push()`

$$(5 * (((9 + 8) * (4 * 6)) + 7))$$

5 9 8 + 4 6 * *



στοίβα S

Στοίβα ώθησης προς τα κάτω

Παράδειγμα: Υπολογισμός απλών αριθμητικών παραστάσεων

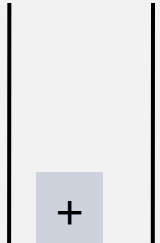
ενθεματική αναπαράσταση ⇒ μεταθεματική αναπαράσταση

Χρησιμοποιούμε στοίβα χαρακτήρων S. Αν το επόμενο σύμβολο είναι:

- αριθμός : τον τυπώνουμε
- '(' : δεν κάνουμε τίποτα
- ')' : εκτελούμε `S.pop()` και τυπώνουμε το σύμβολο που λάβαμε
- τελεστής : εκτελούμε `S.push()`

$$(5 * (((9 + 8) * (4 * 6)) + 7))$$

5 9 8 + 4 6 * * 7



στοίβα S

Στοίβα ώθησης προς τα κάτω

Παράδειγμα: Υπολογισμός απλών αριθμητικών παραστάσεων

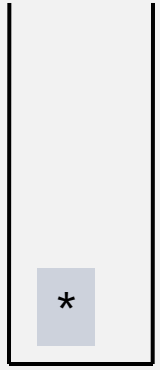
ενθεματική αναπαράσταση ⇒ μεταθεματική αναπαράσταση

Χρησιμοποιούμε στοίβα χαρακτήρων S. Αν το επόμενο σύμβολο είναι:

- αριθμός : τον τυπώνουμε
- '(' : δεν κάνουμε τίποτα
- ')' : εκτελούμε `S.pop()` και τυπώνουμε το σύμβολο που λάβαμε
- τελεστής : εκτελούμε `S.push()`

$$(5 * (((9 + 8) * (4 * 6)) + 7))$$

5 9 8 + 4 6 * * 7 +



στοίβα S

Στοίβα ώθησης προς τα κάτω

Παράδειγμα: Υπολογισμός απλών αριθμητικών παραστάσεων

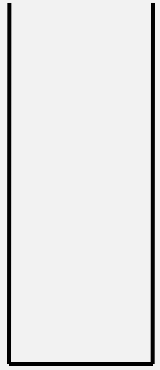
ενθεματική αναπαράσταση ⇒ μεταθεματική αναπαράσταση

Χρησιμοποιούμε στοίβα χαρακτήρων S. Αν το επόμενο σύμβολο είναι:

- αριθμός : τον τυπώνουμε
- ‘(’ : δεν κάνουμε τίποτα
- ‘)’ : εκτελούμε `S.pop()` και τυπώνουμε το σύμβολο που λάβαμε
- τελεστής : εκτελούμε `S.push()`

$$(5 * (((9 + 8) * (4 * 6)) + 7))$$

5 9 8 + 4 6 * * 7 + *



στοίβα S

Στοίβα ώθησης προς τα κάτω

Παράδειγμα: Υπολογισμός απλών αριθμητικών παραστάσεων

ενθεματική αναπαράσταση ⇒ μεταθεματική αναπαράσταση

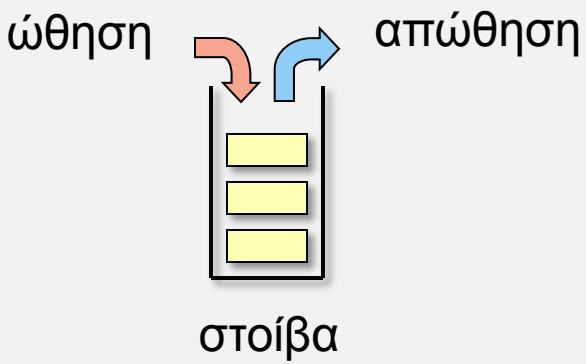
```
public static main(String[] args) {
        char[] a = args[0].toCharArray();
        int N = a.length;
        Stack S = new Stack(N);
        for (i = 0; i < N; i++)
        {
                if (a[i] == ‘)’)
                        System.out.print(S.pop() + “ ”);
                if ((a[i] == ‘*’) || (a[i] == ‘+’))
                        S.push(a[i]);
                if ((a[i] >= ‘0’) && (a[i] <= ‘9’))
                        System.out.print(a[i] + “ “);
        }
        System.out.println(“”);
}
```

Στοίβα ώθησης προς τα κάτω

Η στοίβα υποστηρίζει τις λειτουργίες:

ώθηση(S,x) : τοποθετεί το στοιχείο x στην κορυφή της στοίβας S

απώθηση(S) : επιστρέφει το στοιχείο x που βρίσκεται στην κορυφή της στοίβας S και διαγράφει το x από την S



Η απώθηση απομακρύνει το στοιχείο που προστέθηκε πιο πρόσφατα

⇨ LIFO (Last In First Out) ουρά

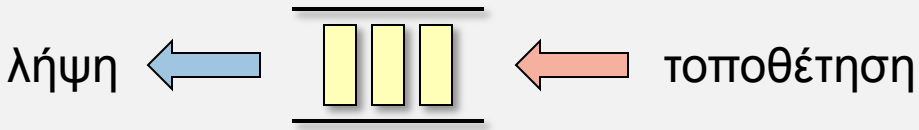
FIFO Ουρά

FIFO (First In First Out) ουρά :

η διαγραφή απομακρύνει το παλαιότερο στοιχείο της ουράς

τοποθέτηση(Q,x) : τοποθετεί το στοιχείο x στο τέλος της ουράς Q

λήψη(Q) : επιστρέφει το στοιχείο x που βρίσκεται στην αρχή της ουράς Q και διαγράφει το x από την Q



FIFO ουρά

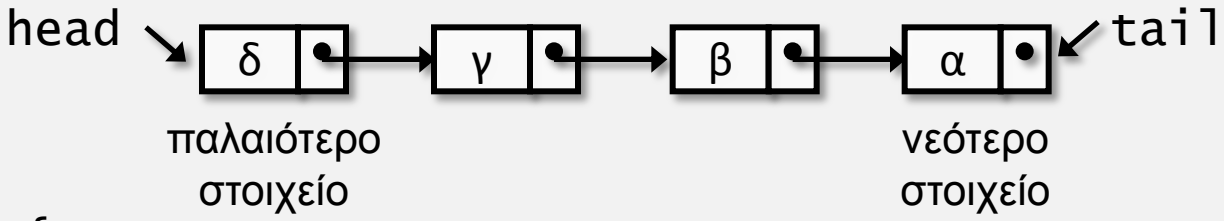
FIFO Ουρά

Διασύνδεση

```
class Queue                  // ουρά αντικειμένων τύπου Item
{
      Queue(int)             // αρχικοποίηση στοίβας
      boolean isEmpty()      // έλεγχος αν η στοίβα είναι άδεια
      void put(Item)         // τοποθέτηση αντικειμένου
      Item get();            // λήψη αντικειμένου
}
```

FIFO Ουρά

Υλοποίηση με συνδεδεμένη λίστα



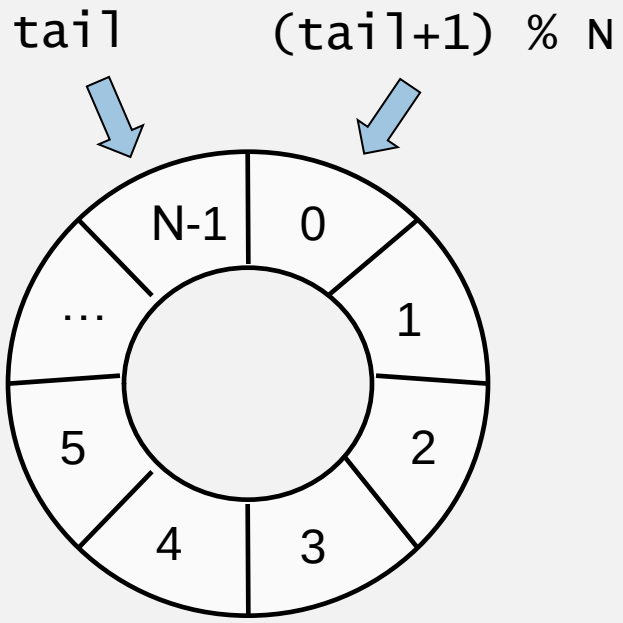

```
class Queue {
        private class Node {
                Item item; Node next;
                Node(Item item) { this.item = item; next = null; }
        }
        private Node head, tail;
        Queue(int maxN) { head = null; tail = null; }

        boolean isEmpty() { return (head == null); }

        void put(Item item) { Node t = tail; tail = new Node(item);
                if (isEmpty()) head = tail; else t.next=tail;  }

        Item get() {  item = head.item; Node t = head.next;
                head = t; return item;  }
```

FIFO Ουρά

Υλοποίηση με πίνακα

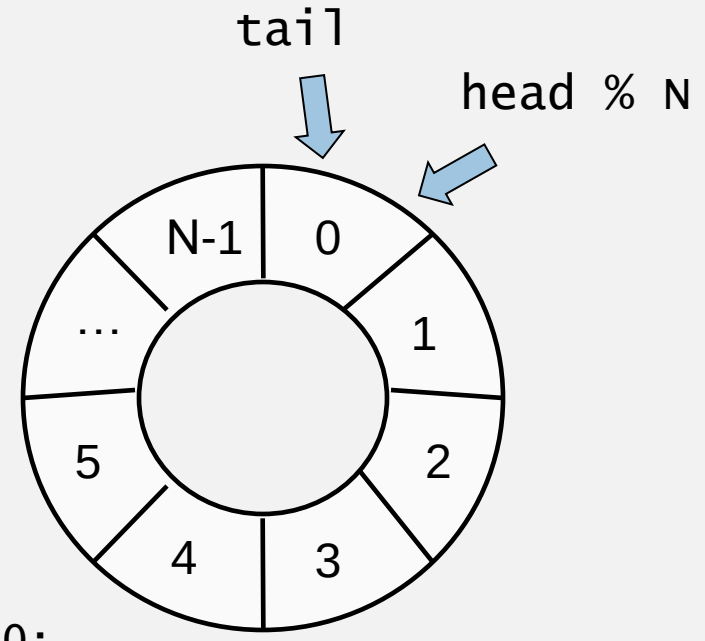
Θεωρούμε την ουρά ως ένα **αναδιπλωμένο πίνακα** `q[]` :

αν η τρέχουσα θέση του πιο πρόσφατου στοιχείου είναι η `tail`
τότε το επόμενο στοιχείο θα εισαχθεί στη θέση `(tail+1) % N`



FIFO Ουρά

Υλοποίηση με πίνακα

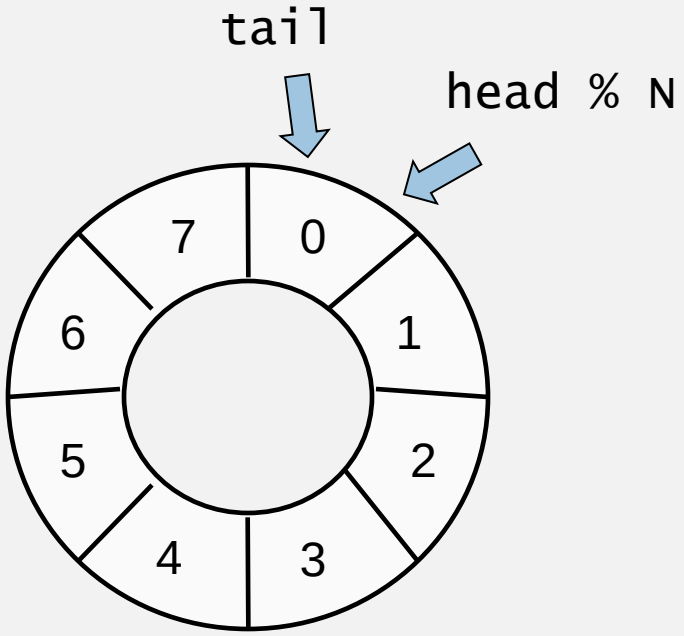



```
class Queue {
        private Item[] q;
        private int N, head, tail;
        Queue(int maxN) {
                q = new Item[maxN + 1];
                N = maxN+1; head = N; tail = 0;
        }

        boolean isEmpty() { return (head % N == tail); }

        void put(Item item) { q[tail++] = item; tail = tail % N; }

        Item get() { head = head % N; return q[head++]; }
}
```

FIFO Ουρά

Υλοποίηση με πίνακα

αρχική κατάσταση:



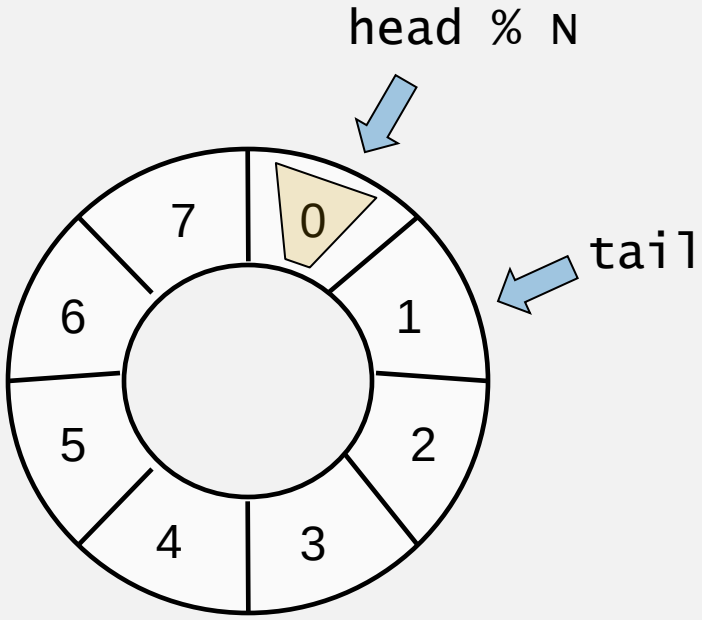
`tail` : δείχνει την επόμενη κενή θέση του πίνακα στην οποία θα τοποθετηθεί νέο στοιχείο

`head` : δείχνει τη θέση του παλαιότερου στοιχείου της ουράς

FIFO Ουρά

Υλοποίηση με πίνακα

τοποθέτηση:



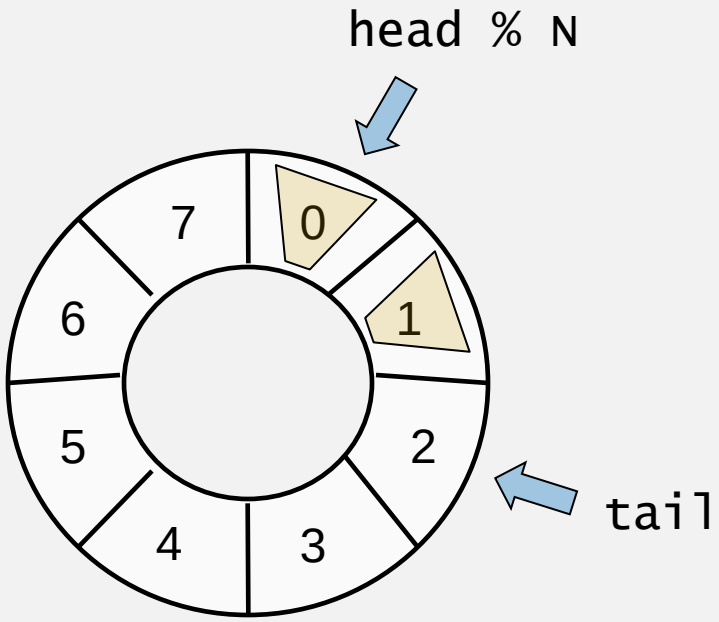
`tail` : δείχνει την επόμενη κενή θέση του πίνακα στην οποία θα τοποθετηθεί νέο στοιχείο

`head` : δείχνει τη θέση του παλαιότερου στοιχείου της ουράς

FIFO Ουρά

Υλοποίηση με πίνακα

τοποθέτηση:



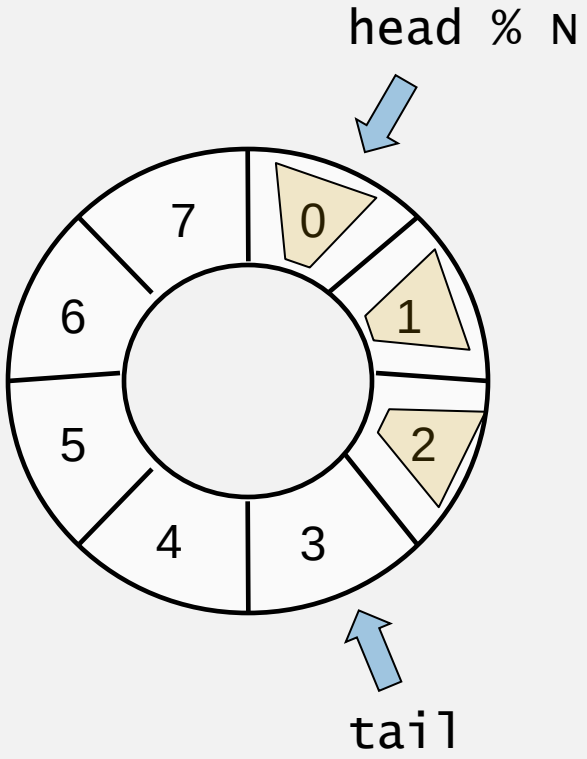
`tail` : δείχνει την επόμενη κενή θέση του πίνακα στην οποία θα τοποθετηθεί νέο στοιχείο

`head` : δείχνει τη θέση του παλαιότερου στοιχείου της ουράς

FIFO Ουρά

Υλοποίηση με πίνακα

τοποθέτηση:



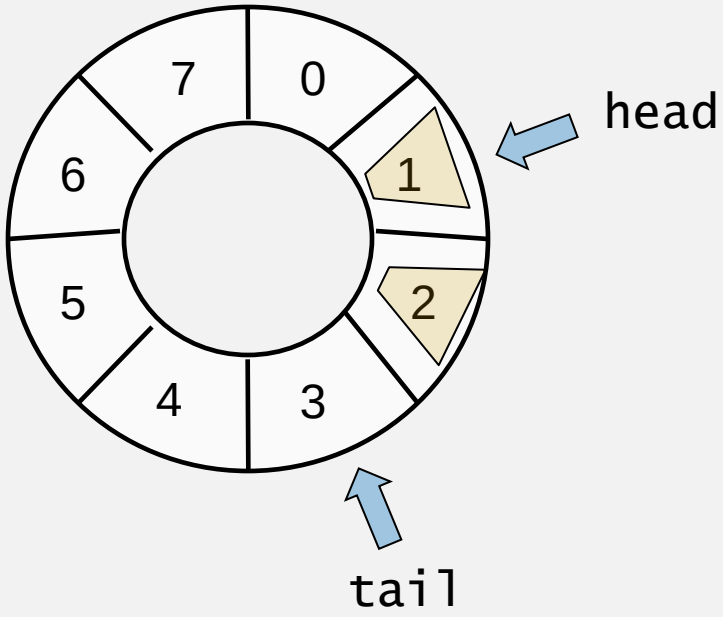
`tail` : δείχνει την επόμενη κενή θέση του πίνακα στην οποία θα τοποθετηθεί νέο στοιχείο

`head` : δείχνει τη θέση του παλαιότερου στοιχείου της ουράς

FIFO Ουρά

Υλοποίηση με πίνακα

λήψη:



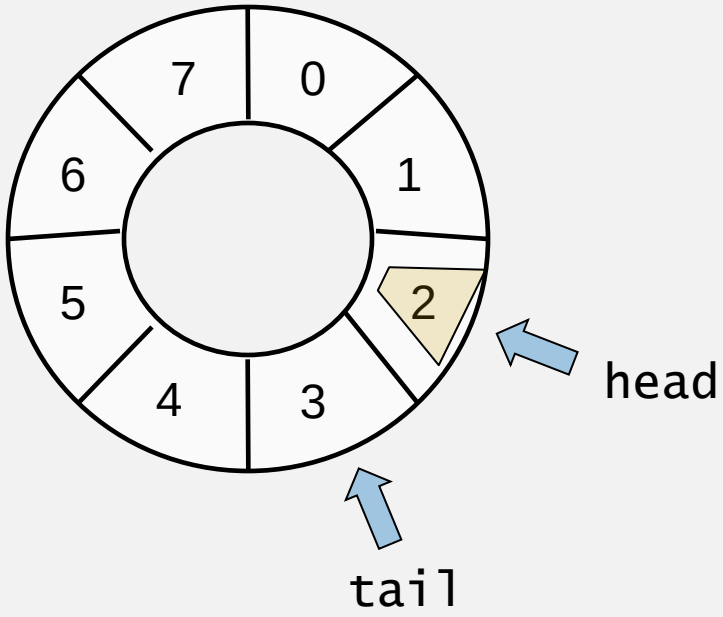
`tail` : δείχνει την επόμενη κενή θέση του πίνακα στην οποία θα τοποθετηθεί νέο στοιχείο

`head` : δείχνει τη θέση του παλαιότερου στοιχείου της ουράς

FIFO Ουρά

Υλοποίηση με πίνακα

λήψη:



`tail` : δείχνει την επόμενη κενή θέση του πίνακα στην οποία θα τοποθετηθεί νέο στοιχείο

`head` : δείχνει τη θέση του παλαιότερου στοιχείου της ουράς

FIFO Ουρά

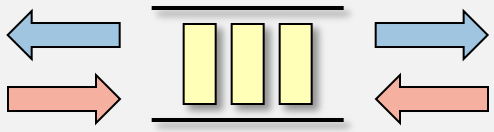
Όπως και στην περίπτωση της στοίβας :

- Και οι δύο υλοποιήσεις επιτυγχάνουν O(1) χρόνο ανά λειτουργία.
- Η υλοποίηση με πίνακα είναι πιο γρήγορη στην πράξη.
- Η υλοποίηση με πίνακα έχει το μειονέκτημα ότι πρέπει να οριστεί ο μέγιστος αριθμός στοιχείων (`maxN`) στην ουρά. Το αν αυτό αποτελεί σημαντικό πρόβλημα ή όχι εξαρτάται από την εφαρμογή.

Ουρά δύο άκρων

Deque (Double Ended Queue) :

επιτρέπει εισαγωγή/διαγραφή και στα δύο άκρα της ουράς



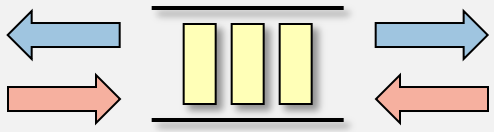
deque

Μπορεί να υλοποιηθεί με παρόμοιο τρόπο έτσι ώστε να επιτυγχάνει Ο(1) χρόνο ανά λειτουργία

Ουρά δύο άκρων

Deque (Double Ended Queue) :

επιτρέπει εισαγωγή/διαγραφή και στα δύο άκρα της ουράς



deque

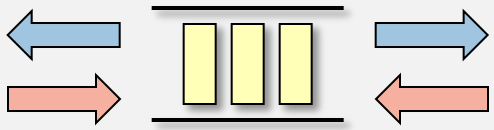
Μπορεί να υλοποιηθεί με παρόμοιο τρόπο έτσι ώστε να επιτυγχάνει Ο(1) χρόνο ανά λειτουργία

Τι συμβαίνει όταν μπορούμε να έχουμε εισαγωγή του ίδιου στοιχείου πάνω από 1 φορά;

Ουρά δύο άκρων

Deque (Double Ended Queue) :

επιτρέπει εισαγωγή/διαγραφή και στα δύο άκρα της ουράς



deque

Μπορεί να υλοποιηθεί με παρόμοιο τρόπο έτσι ώστε να επιτυγχάνει Ο(1) χρόνο ανά λειτουργία

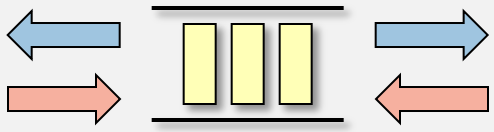
Τι συμβαίνει όταν μπορούμε να έχουμε εισαγωγή του ίδιου στοιχείου πάνω από 1 φορά;

Για να διατηρήσουμε Ο(1) χρόνο ανά λειτουργία σε αυτή την περίπτωση μπορούμε να χρησιμοποιήσουμε την τεχνική του **κατακερματισμού** που θα δούμε αργότερα…

Ουρά δύο άκρων

Deque (Double Ended Queue) :

επιτρέπει εισαγωγή/διαγραφή και στα δύο άκρα της ουράς



deque

Μπορεί να υλοποιηθεί με παρόμοιο τρόπο έτσι ώστε να επιτυγχάνει Ο(1) χρόνο ανά λειτουργία

Τι συμβαίνει όταν μπορούμε να έχουμε εισαγωγή του ίδιου στοιχείου πάνω από 1 φορά;

Απλή λύση αν τα στοιχεία που εισάγουμε είναι ακέραιοι στο διάστημα [0,M-1] :

Διατηρούμε ένα πίνακα `M` θέσεων `t[]` (τύπου `boolean`) :

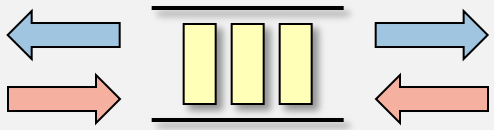
`t[i]=true` αν και μόνο αν το στοιχείο `i` υπάρχει στην ουρά

`t[i]=false` αν και μόνο αν το στοιχείο `i` δεν υπάρχει στην ουρά

Ουρά δύο άκρων

Deque (Double Ended Queue) :

επιτρέπει εισαγωγή/διαγραφή και στα δύο άκρα της ουράς



deque

Μπορεί να υλοποιηθεί με παρόμοιο τρόπο έτσι ώστε να επιτυγχάνει
O(1) χρόνο ανά λειτουργία

Τι συμβαίνει όταν μπορούμε να έχουμε εισαγωγή του ίδιου στοιχείου πάνω από 1 φορά;

Ακόμα και αν μπορούμε να ελέγξουμε γρήγορα αν ένα στοιχείο υπάρχει στην ουρά σε περίπτωση εισαγωγής του ίδιου στοιχείου ποιο θα κρατήσουμε;

Το παλαιότερο ή το νεότερο;
(Έχει σημασία όταν το κάθε στοιχείο συνοδεύεται και από άλλα δεδομένα)