Δομές Δεδομένων & Ανάλυση Αλγορίθμων

3ο Εξάμηνο

Στοίβα (Stack) και Ουρά (Queue)

Υλοποίηση τους με τη βοήθεια πίνακα

Δημοσθένης Σταμάτης
http:/www.iee.ihu.gr/~demos
Τμήμα Μηχανικών Πληροφορικής & Ηλεκτρονικών Συστημάτων

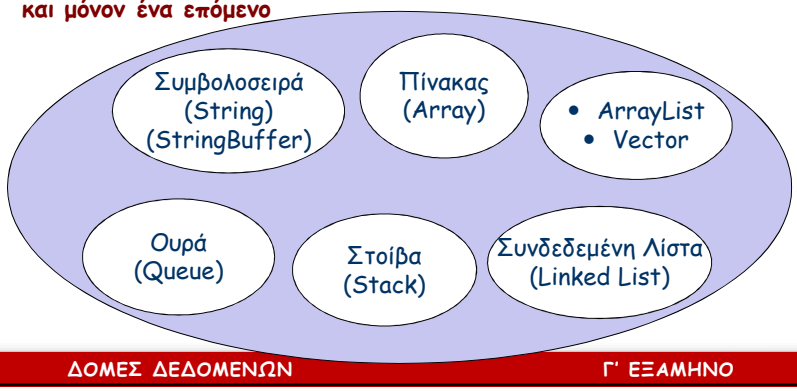
Γραμμική Δομή Δεδομένων

Ονομάζεται η δομή δεδομένων το σύνολο των στοιχείων της οποίας είναι διατεταγμένο με τέτοιο τρόπο ώστε να ισχύουν τα εξής:

(α) υπάρχει ένα στοιχείο το οποίο ονομάζεται αρχή και έχει ένα και μόνον ένα επόμενο στοιχείο

(β) υπάρχει ένα στοιχείο το οποίο ονομάζεται τέλος και έχει ένα και μόνον ένα προηγούμενο

(γ) κάθε άλλο στοιχείο έχει ένα και μόνον ένα προηγούμενο και ένα και μόνον ένα επόμενο



Στοίβα (Stack)

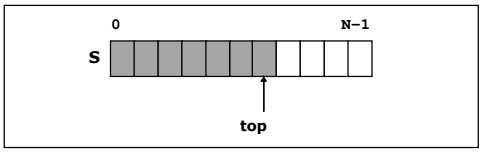
☞ Στοίβα (stack) είναι μία λίστα στην οποία νέα στοιχεία μπορούν να προστεθούν και να αφαιρεθούν μόνο από τη μία άκρη της (κορυφή της στοίβας). Μία στοίβα αναφέρεται και σαν μία λίστα τύπου LIFO (Last-In-First-Out)

☞ Η Στοίβα σαν Αφηρημένος Τύπος Δεδομένων στη Java:

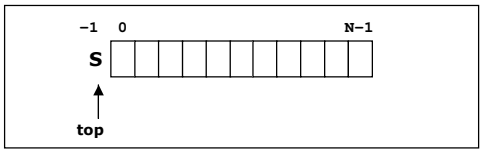
```
public interface Stack
{
public int size( );
  // επιστρέφει το μέγεθος της στοίβας
public boolean isEmpty( );
  // αληθεύει εάν η στοίβα είναι κενή
public Object top( ) throws StackEmptyException;
  // επιστρέφει το στοιχείο που βρίσκεται στην κορυφή της στοίβας
public void push(Object item) throws StackFullException;
  // εισάγει ένα νέο στοιχείο στην κορυφή της στοίβας
public Object pop( ) throws StackEmptyException;
  // εξάγει και επιστρέφει το στοιχείο που βρίσκεται στην κορυφή της στοίβας
}
```


Υλοποίηση Στοίβας με τη βοήθεια πίνακα

☞ Υλοποιείται με τη βοήθεια ενός πίνακα (S) και ενός ακέραιου ενδείκτη (top). Το S[top] είναι η κορυφή της στοίβας.



☞ Όταν η στοίβα δημιουργείται έχει τη μορφή:



Υλοποίηση Στοίβας με τη βοήθεια πίνακα

(κλάση ArrayStack)

Δημιουργία Στοίβας

Το μέγεθος της στοίβας μπορεί να καθορίζεται είτε έμμεσα π.χ 100 στοιχεία είτε άμεσα κατά τη δημιουργία της στοίβας. Οι δομητές της κλάσης ArrayStack για τις δύο αυτές περιπτώσεις φαίνονται παρακάτω:

```
public class ArrayStack implements Stack
{
        public static final int CAPACITY = 100;
        private int capacity;
        private Object[ ] S;
        private int top = -1;

        public ArrayStack( ) {
                this(CAPACITY);
        }

        public ArrayStack(int cap) {
                capacity = cap;
                S = new Object[capacity];
        }
. . . .
```


ΤΜΗΜΑ ΜΗΧ.ΠΛΗΡΟΦΟΡΙΚΗΣ & ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ ΔΙΠΑΕ

Υλοποίηση Στοίβας με τη βοήθεια πίνακα

Μέγεθος Στοίβας

```
public int size( )
{
  return (top+1);
}
```

Έλεγχος Κενής Στοίβας

```
public boolean isEmpty( )
{
  return (top < 0);
}
```

Έλεγχος της Κορυφής της Στοίβας

```
public Object top( ) throws StackEmptyException
{
  if (isEmpty( ))
    throw new StackEmptyException("Stack is empty");
  return S[top];
}
```

ΔΟΜΕΣ ΔΕΔΟΜΕΝΩΝ Γ' ΕΞΑΜΗΝΟ

Υλοποίηση Στοίβας με τη βοήθεια πίνακα

Εισαγωγή στοιχείου στη Στοίβα

//βλέπε interface Stack

Η διαδικασία εισαγωγής ενός στοιχείου στη στοίβα συνίσταται στην αύξηση του ενδείκτη **top** κατά ένα και την τοποθέτηση του στοιχείου στη θέση του πίνακα που δείχνει ο **top**. Στην περίπτωση που ο χώρος της στοίβας έχει εξαντληθεί από προηγούμενες εισαγωγές στοιχείων, δημιουργείται κατάσταση εξαίρεσης (υπερχείλιση στοίβας). Η μέθοδος **push** της **ArrayStack** είναι η εξής:

```
public void push(Object item) throws StackFullException
{
  if (size( )== capacity-1)
    throw new StackFullException("Stack overflow");
  S[++top] = item;
}
```

Η εξαίρεση **StackFullException** πρέπει να οριστεί με τη βοήθεια ξεχωριστής κλάσης, η οποία επεκτείνει την κλάση **RuntimeException**:

```
public class StackFullException extends RuntimeException
{
  public StackFullException(String err)
          { super(err); }
}
```


Υλοποίηση Στοίβας με τη βοήθεια πίνακα

Εξαγωγή στοιχείου από Στοίβα

//βλέπε interface Stack

Η διαδικασία εξαγωγής ενός στοιχείου από τη στοίβα συνίσταται στην επιστροφή του στοιχείου που βρίσκεται στην κορυφή της στοίβας και τη μείωση του ενδείκτη top κατά ένα. Εάν κατά την πράξη εξαγωγής στοιχείου η στοίβα βρεθεί άδεια δημιουργείται κατάσταση εξαίρεσης (άδεια στοίβα):

```
public Object pop( ) throws StackEmptyException {
  Object element;
  if (isEmpty())
     throw new StackEmptyException("Stack is empty");
  element = S[top];
  S[top--] = null; //για τον garbage collector!!!
  return element;
}
```

Η εξαίρεση **StackEmptyException** πρέπει να οριστεί με τη βοήθεια ξεχωριστής κλάσης, η οποία επεκτείνει την κλάση **RuntimeException**:

```
public class StackEmptyException extends RuntimeException {
  public StackEmptyException(String err)
          { super(err);  }
}
```

Ουρά (Queue)

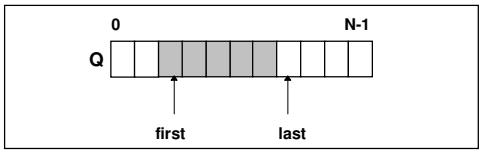
☞ Ουρά (queque) είναι μια λίστα στην οποία μπορούν να προστεθούν στοιχεία μόνο στη μία άκρη (πίσω) και να αφαιρεθούν μόνο από την άλλη (μπροστά). Μια ουρά αναφέρεται και σαν λίστα τύπου FIFO (First-In-First-Out).

☞ Η Ουρά σαν Αφηρημένος Τύπος Δεδομένων στη Java:

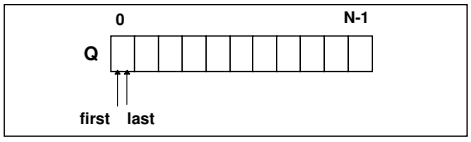
```
public interface Queue
{
public int size( );
  // επιστρέφει το μέγεθος (αριθμός στοιχείων) της ουράς

public boolean isEmpty( );
  // αληθεύει εάν η ουρά είναι κενή

public Object front( ) throws QueueEmptyException;
  // επιστρέφει το στοιχείο που βρίσκεται στο εμπρός μέρος της ουράς

public void enqueue(Object item) throws QueueFullException;
  // εισάγει ένα νέο στοιχείο στο πίσω μέρος της ουράς

public Object dequeue( ) throws QueueEmptyException;
  // εξάγει και επιστρέφει το στοιχείο που βρίσκεται
  // στο εμπρός μέρος της ουράς
}
```


Υλοποίηση Ουράς με τη βοήθεια πίνακα

☞ Υλοποιείται με τη βοήθεια ενός πίνακα (Q) και δύο ακέραιων ενδεικτών (first και last). Το Q[first] είναι το πρώτο και το Q[last] το τελευταίο στοιχείο της ουράς αντίστοιχα.



☞ Όταν η ουρά δημιουργείται έχει τη μορφή:



Υλοποίηση Ουράς με τη βοήθεια πίνακα

(κλάση **ArrayQueue**)

Δημιουργία Ουράς

Το μέγεθος της ουράς μπορεί να καθορίζεται είτε έμμεσα π.χ 1000 στοιχεία είτε άμεσα κατά τη δημιουργία της στοίβας. Οι δομητές της κλάσης ArrayQueue για τις δύο αυτές περιπτώσεις φαίνονται παρακάτω:

```
public class ArrayQueue implements Queue
{
        public static final int CAPACITY = 1000;
        private int capacity;
        private Object[ ] Q;
        private int first = 0;
        private int last = 0;

        public ArrayQueue() {
                this(CAPACITY);
        }

        public ArrayQueue(int cap) {
                capacity = cap;
                Q = new Object[capacity];
```


Υλοποίηση Ουράς με τη βοήθεια πίνακα

Μέγεθος Ουράς

```
public int size( )
{
  return (last-first);
}
```

Έλεγχος Κενής Ουράς

```
public boolean isEmpty( )
{
  return (first==last);
}
```

Επιστροφή Πρώτου Στοιχείου της Ουράς

```
public Object front( ) throws QueueEmptyException
{
  if (isEmpty( ))
      throw new QueueEmptyException("Queue is empty");
  return Q[first];
}
```

Υλοποίηση Ουράς με τη βοήθεια πίνακα

Εισαγωγή στοιχείου στην Ουρά

//βλέπε interface Queue

Η διαδικασία εισαγωγής ενός στοιχείου στην ουρά συνίσταται στην τοποθέτηση του στοιχείου στη θέση του πίνακα που δείχνει ο ενδείκτης **last** και στη συνέχεια την αύξηση του ενδείκτη κατά ένα. Στην περίπτωση που ο χώρος της ουράς έχει εξαντληθεί από προηγούμενες εισαγωγές στοιχείων, δημιουργείται κατάσταση εξαίρεσης (υπερχείλιση ουράς):

```
public void enqueue(Object item) throws QueueFullException
{
  if (last == capacity)  /*  Εικονική Υπερχείλιση??  */
       throw new QueueFullException("Queue overflow");
  Q[last++] = item;
}
```

Η εξαίρεση **QueueFullException** πρέπει να οριστεί με τη βοήθεια ξεχωριστής κλάσης, η οποία επεκτείνει την κλάση **RuntimeException**:

```
public class QueueFullException extends RuntimeException
{
  public QueueFullException(String err)
            { super(err); }
}
```


Υλοποίηση Ουράς με τη βοήθεια πίνακα

Εξαγωγή στοιχείου από Ουρά

//βλέπε interface Queue

Η διαδικασία εξαγωγής ενός στοιχείου από την ουρά συνίσταται στην επιστροφή του στοιχείου που βρίσκεται στη θέση first του πίνακα και στη συνέχεια την αύξηση του ενδείκτη κατά ένα. Στην περίπτωση που η ουρά είναι άδεια, δημιουργείται κατάσταση εξαίρεσης (άδεια ουρά):

```
public Object dequeue( ) throws QueueEmptyException
{
  Object item;
  if (isEmpty( ))
      throw new QueueEmptyException("Queue is empty");
  item = Q[first];
  Q[first++] = null; //!!! για τον garbage collector
  return item;
}
```

Η εξαίρεση **QueueEmptyException** πρέπει να οριστεί με τη βοήθεια ξεχωριστής κλάσης, η οποία επεκτείνει την κλάση **RuntimeException**:

```
public class QueueEmptyException extends RuntimeException
{
  public QueueEmptyException(String err)
            { super(err); }
}
```