Οικονομικό Πανεπιστήμιο Αθηνών

Τμήμα Πληροφορικής

Φθινοπωρινό Εξάμηνο 2016

Δομές Δεδομένων - Εργασία 1

Διδάσκων: Ε. Μαρκάκης

Στοίβες και Ουρές

Σκοπός της εργασίας είναι η εξοικείωση με βασικούς αφηρημένους τύπους δεδομένων όπως οι στοίβες, και οι ουρές FIFO. Η εργασία αποτελείται από υλοποιήσεις ΑΤΔ (Μερη Α και Γ) καθώς και 1 εφαρμογή (Μέρος Β). Διαβάστε προσεκτικά την εκφώνηση και τα ζητούμενα της εργασίας.

Μέρος Α. Στο παράρτημα της εκφώνησης δίνονται οι διεπαφές StringStack και StringQueue, που δηλώνουν τις βασικές μεθόδους για μια στοίβα και μια ουρά FIFO, με στοιχεία τύπου *String*. Δημιουργήστε μία υλοποίηση των ΑΤΔ StringStack και StringQueue, δηλαδή γράψτε 2 κλάσεις που υλοποιούν τις 2 διεπαφές.

Οδηγίες υλοποίησης:

- Οι κλάσεις σας **πρέπει να λέγονται** StringStackImpl και StringQueueImpl.
- Η υλοποίηση και για τις 2 διεπαφές θα πρέπει να γίνει χρησιμοποιώντας λίστα μονής σύνδεσης.
- Κάθε πράξη εισαγωγής ή εξαγωγής στοιχείου (δηλαδή κάθε εκτέλεση των μεθόδων push και pop στη στοίβα, και put και get στην ουρά) θα πρέπει να ολοκληρώνεται σε χρόνο O(1), δηλαδή σε χρόνο ανεξάρτητο από τον αριθμό των αντικειμένων που είναι μέσα στην ουρά.
- Όταν η στοίβα ή η ουρά είναι άδεια, οι μέθοδοι που διαβάζουν από την δομή θα πρέπει να πετάνε εξαίρεση τύπου NoSuchElementException. Η εξαίρεση NoSuchElementException ανήκει στην core βιβλιοθήκη της Java. Κάντε την import από το πακέτο **java.util**. Μην κατασκευάσετε δική σας εξαίρεση.
- Μπορείτε να χρησιμοποιήσετε τη λίστα μονής σύνδεσης που παρουσιάστηκε στο εργαστήριο 2 του μαθήματος ή να γράψετε εξ' ολοκλήρου τη δική σας λίστα ή να χρησιμοποιήσετε μόνο αντικείμενα τύπου Node μέσα στην κλάση της στοίβας/ουράς. Για να αποκτήσετε καλύτερη εξοικείωση προτείνουμε να ξεκινήσετε από την αρχή και να γράψετε τις δικές σας κλάσεις (σίγουρα δεν θα χάσετε μονάδες όμως χρησιμοποιώντας ό,τι έχετε δει στο εργαστήριο).
- **Προαιρετικά:** μπορείτε να κάνετε την υλοποίηση σας με χρήση generics για να μπορείτε να χειρίζεστε στοίβες και ουρές με οποιοδηποτε τύπο αντικειμένων. Υπάρχει ένα 10% bonus σε όσους χρησιμοποιήσουν generics για όλη την εργασία.
- **Δεν επιτρέπεται να χρησιμοποιήσετε έτοιμες υλοποιήσεις δομών τύπου λίστας, στοίβας, ουράς, από την βιβλιοθήκη της Java** (π.χ. Vector, ArrayList κλπ).

Τα αρχεία StringStack.java και StringQueue.java που περιέχουν τις δηλώσεις των διεπαφών είναι διαθέσιμα στο φάκελο 'Εγγραφα/Εργασίες στο eclass.

Μέρος Β. Χρησιμοποιώντας την υλοποίηση της στοίβας του μέρους Α, γράψτε ένα πρόγραμμα-πελάτη το οποίο θα κάνει διάσχιση σε έναν λαβύρινθο με σκοπό να βρίσκει την έξοδο. Το πρόγραμμά σας θα πρέπει να διαβάζει την είσοδο από ένα .txt αρχείο, το οποίο θα περιέχει τον λαβύρινθο σε μορφή πίνακα χαρακτήρων, διαστάσεων n x m, όπου n, m, ακέραιοι. Ο πίνακας μπορεί να περιέχει μόνο τους χαρακτηρες 0, 1 και Ε, όπου το Ε θα βρίσκεται μόνο σε ένα σημείο του πίνακα και θα υποδηλώνει την είσοδο στο λαβύρινθο. Όταν κάποιος μπαίνει στον λαβύρινθο, μπορεί να κινηθεί οριζόντια ή κάθετα (αλλά όχι διαγώνια) προς οποιαδήποτε κατεύθυνση που περιέχει 0 (δείτε το παράδειγμα παρακάτω). Αν φτάσετε στα σύνορα του πίνακα (πρώτη ή τελευταία γραμμή και πρώτη ή τελευταία στήλη), και βρείτε 0, τότε έχετε φτάσει σε μια έξοδο του λαβυρίνθου. Είναι δυνατόν να υπάρχουν πολλαπλές έξοδοι στο λαβύρινθο (ή και καμία). Το πρόγραμμά σας θα πρέπει να τυπώνει τις συντεταγμένες της εξόδου που βρήκε, ενώ αν δεν υπάρχει τρόπος διαφυγής, θα πρέπει να τυπώνει αντίστοιχο μήνυμα.

Παράδειγμα: Η είσοδός θα πρέπει να είναι στην μορφή του παρακάτω παραδείγματος

```
9  7
0  3
1  1  1  E  1  1  1
1  1  1  0  1  1  1
1  0  0  0  1  0  1
1  0  1  0  1  0  0
1  1  1  0  1  1  1
1  0  0  0  0  0  1
1  0  1  1  1  0  1
1  0  1  1  0  0  1
0  1  1  1  0  1  1
```

Στην πρώτη γραμμή, δηλώνονται οι διαστάσεις του λαβυρίνθου (εδώ n=9, m=7). Στην δεύτερη γραμμή ακολουθούν οι συντεταγμένες του σημείου εισόδου, όπου εδώ είναι στην γραμμή 0 και στην στήλη 3 (υποθέτουμε ότι ονομάζουμε τις γραμμές από 0 ως n-1 και τις στήλες από 0 ως m-1). Στο παράδειγμα αυτό, η εξερεύνηση θα ξεκινήσει κινούμενοι προς τα κάτω. Αν στρίψετε αριστερά (όπως κοιτάμε τον πίνακα), θα δείτε ότι θα φτάσετε σύντομα σε αδιέξοδο και θα πρέπει να γυρίσετε πίσω. Εν τέλει, συνεχίζοντας το ψάξιμο, μια έξοδος που μπορείτε να φτάσετε είναι στις συντεταγμένες (8, 4).

Οδηγίες υλοποίησης:

- Το πρόγραμμα σας **πρέπει να λέγεται** Thiseas.java.
- Θα πρέπει να κάνετε χρήση της υλοποίησης της στοίβας από το Μέρος Α.
- Η χρήση της στοίβας ενδείκνυται για να μπορέσετε να υλοποιήσετε την αναζήτηση της εξόδου με **backtracking**. Σκεφτείτε τι πρέπει να κάνετε όταν φτάνετε σε αδιέξοδο, και με ποιο τρόπο θα μπορέσετε να συνεχίσετε την αναζήτηση.
- Είναι επιτρεπτό, αν αποθηκεύσετε τον λαβύρινθο σε πίνακα χαρακτήρων, να κάνετε μετέπειτα αλλαγές πάνω στα στοιχεία του πίνακα (π.χ. αν θέλετε να χρησιμοποιήσετε κάποιον άλλο χαρακτήρα για να δείξετε ότι έχετε ήδη επισκεφτεί κάποια θέση κατά τη διάρκεια εκτέλεσης του προγράμματος).
- Η εργασία σας θα εξεταστεί σε εισόδους της παραπάνω μορφής. Αν υπάρχει κάποιο λάθος στα δεδομένα εισόδου, το πρόγραμμα πρέπει να τερματίζει τυπώνοντας αντίστοιχο μήνυμα (π.χ. αν οι γραμμές και στήλες που διαβάσατε στην πρώτη σειρά εισόδου, δεν ταιριάζουν με τον πίνακα που διαβάζετε μετά ή αν δεν υπάρχει Ε στον λαβύρινθο, κτλ).
- Πρέπει να δίνετε ως όρισμα ολο το μονοπάτι για το .txt αρχείο εισόδου, δηλαδή η εκτέλεση του προγράμματος, αν το τρέξετε από τη γραμμή εντολών θα είναι ως εξής:

> java Thiseas path_to_file/filename.txt

Μέρος Γ. Η υλοποίηση της διεπαφής StringQueue με λίστα μονής σύνδεσης στο μέρος Α, θα πρέπει αναγκαστικά να χρησιμοποιεί 2 δείκτες, τους head και tail, για να μπορείτε να εκτελείτε σωστά τις εισαγωγές και εξαγωγές στοιχείων, όπως είδαμε και στο μάθημα. Το ζητούμενο στο μέρος Γ είναι να φτιάξετε μία νέα υλοποίηση της ουράς FIFO, χρησιμοποιώντας μόνο έναν από τους δείκτες αυτούς. **Hint:** Χρησιμοποιήστε κυκλική λίστα αντί για λίστα μονής σύνδεσης.

Οδηγίες υλοποίησης:

- Η κλάση σας **πρέπει να λέγεται** *StringQueueWithOnePointer.java*.
- Οι λειτουργίες εισαγωγής και εξαγωγής θα πρέπει να γίνονται σε O(1) και γενικότερα ισχύουν και εδώ όλες οι οδηγίες που αναγράφονται για το Μέρος Α.

Οδηγίες Παράδοσης

Η εργασία σας θα πρέπει να μην έχει συντακτικά λάθη και να μπορεί να μεταγλωττίζεται. Εργασίες που δεν μεταγλωττίζονται χάνουν το 50% της συνολικής αξίας.

Η εργασία θα αποτελείται από:

1. Τον πηγαίο κώδικα (source code). Τοποθετήστε σε ένα φάκελο με όνομα **src** τα αρχεία java που έχετε φτιάξει. Ενδεικτικά θα πρέπει να περιέχει (χρησιμοποιήστε τα όνοματα των κλάσεων όπως ακριβώς δίνονται στην εκφώνηση):
    a. Τις διεπαφές StringStack και StringQueue, οι οποίες είναι ήδη διαθέσιμες στο eclass.
    b. Όλες τις υλοποιήσεις των διεπαφών από το Μέρος Α και το Μέρος Γ.
    c. Το πρόγραμμα Thiseas.java.

    Επιπλέον, φροντίστε να συμπεριλάβετε όποια άλλα αρχεία πηγαίου κώδικα απαιτούνται για να μεταγλωττίζεται η εργασία σας (π.χ. αν χρησιμοποιήσετε την λίστα μονής σύνδεσης του εργαστηρίου 2, να συμπεριλάβετε τα σχετικά αρχεία). Φροντίστε επίσης να προσθέσετε επεξηγηματικά σχόλια όπου κρίνετε απαραίτητο στον κώδικά σας.

2. Γράψτε μία σύντομη αναφορά σε pdf αρχείο (οχι Word ή txt!) με όνομα project1-report.pdf, στην οποία θα αναφερθείτε στα εξής:
    a. Εξηγήστε συνοπτικά πώς υλοποιήσατε τις διεπαφές στα μέρη Α και Γ (άνω όριο 2 σελίδες).
    b. Για το μέρος Β, εξηγήστε πώς χρησιμοποιήσατε την υλοποίηση από το μέρος Α για να φτιάξετε το πρόγραμμα που ζητείται (άνω όριο 2 σελίδες).

Όλα τα παραπάνω αρχεία θα πρέπει να μπουν σε ένα αρχείο zip. Το όνομα που θα δώσετε στο αρχείο αυτό θα είναι ο αριθμός μητρώου σας πχ. 3030056_3030066.zip ή 3030056.zip (αν δεν είστε σε ομάδα). Στη συνέχεια, θα υποβάλλετε το zip αρχείο σας στην περιοχή του μαθήματος «Εργασίες» στο e-class. Δεν χρειάζεται υποβολή και από τους 2 φοιτητές μιας ομάδας.

Η προθεσμία παράδοσης της εργασίας είναι Δευτέρα, 15 Νοεμβρίου 2016 και ώρα 23:59.

Παράρτημα

Διεπαφή StringStack

```
import java.io.PrintStream;
import java.util.NoSuchElementException;

/**
 * Defines the methods for a Stack that handles String items
 */
public interface StringStack {

	/**
	 * @return true if the stack is empty
	 */
	public boolean isEmpty();

	/**
	 * Push a String item to the stack
	 */
	public void push(String item);

	/**
	 * remove and return the item on the top of the stack
	 * @return the item on the top of the stack
	 * @throws a NoSuchElementException if the stack is empty
	 */
	public String pop() throws NoSuchElementException;

	/**
	 * return without removing the item on the top of the stack
	 * @return the item on the top of the stack
	 * @throws a NoSuchElementException if the stack is empty
	 */
	public String peek() throws NoSuchElementException;

	/**
	 * print the contents of the stack, starting from the item
	 * on the top,
	 * to the stream given as argument. For example,
	 * to print to the standard output you need to pass System.out as
	 * an argument. E.g.,
	 * printStack(System.out);
	 */
	public void printStack(PrintStream stream);

	/**
	 * return the size of the stack, 0 if empty
	 * @return the number of items currently in the stack
	 */
	public int size();

}
```

Διεπαφή StringQueue

```
import java.io.PrintStream;
import java.util.NoSuchElementException;

/**
 * Defines the methods for a FIFO queue that handles String items
 */
public interface StringQueue {

	/**
	 * @return true if the queue is empty
	 */
	public boolean isEmpty();

	/**
	 * insert a String item to the queue
	 */
	public void put(String item);

	/**
	 * remove and return the oldest item of the queue
	 * @return oldest item of the queue
	 * @throws NoSuchElementException if the queue is empty
	 */
	public String get() throws NoSuchElementException;

	/**
	 * return without removing the oldest item of the queue
	 * @return oldest item of the queue
	 * @throws NoSuchElementException if the queue is empty
	 */
	public String peek() throws NoSuchElementException;


	/**
	 * print the elements of the queue, starting from the oldest
	 * item, to the print stream given as argument. For example, to
	 * print the elements to the
	 * standard output, pass System.out as parameter. E.g.,
	 * printQueue(System.out);
	 */
	public void printQueue(PrintStream stream);

	/**
	 * return the size of the queue, 0 if it is empty
	 * @return number of elements in the queue
	 */
	public int size();
}
```