Μερος Δ

Ιωάννης Ματσούκας p3170106

Δημήτρης Μπεκιάρης p3170116

Μέρος Α

Και στις δυο δομές έχουμε χρησιμοποιήσει generics.
Στοίβα:

Υπάρχει η κλάση ListNode<T> με ιδιότητες data τύπου T και next τύπου ListNode και μεθόδους get set αντίστοιχα και μια toString. Η StringStackImpl<T> έχει ως ιδιότητες το first τύπου ListNode και το size τύπου int. Το first ουσιαστικά είναι ένας pointer προς το πρώτο ListNode της στοίβας, το οποίο περιέχει το πρώτο της αντικείμενο τύπου T. Το size είναι ο αριθμός των αντικειμένων που έχει η στοίβα. Η κλάση αυτή επιπλέον υλοποιεί την διεπαφή StringStack με μεθόδους : isEmpty(), push(T item), pop(), peek(), printStack(PrintStream stream) και size(). Η εισαγωγή γίνεται με την ακόλουθη μέθοδο:

```
public void push(T item) {
        if (this.first != null) {
                this.first = new ListNode<T>(item,this.first);
        } else {
                his.first = new ListNode<T>(item);
        }
        this.size++;
}
```

και η αφαίρεση με τη μέθοδο:

```
public T pop() throws NoSuchElementException {
        if (this.size > 0) {
                T ob = this.first.getData();
                this.first = this.first.getNext();
                this.size--;
                return ob;
        } else {
                throw new java.util.NoSuchElementException();
        }
}
```

Και οι 2 μέθοδοι δεν επηρεάζονται από το μέγεθος της στοίβας (έχουν δηλαδή πολυπλοκότητα O(1)). Η υλοποίηση των άλλων μεθόδων είναι οι παρακάτω:

```
public boolean isEmpty() {
	return this.size == 0;
}

public int size() {
	return this.size;
}

public T peek() throws NoSuchElementException {
	if (this.size > 0) {
		return this.first.getData();
	} else {
		throw new NoSuchElementException();
	}
}

public void printStack(PrintStream stream) {
	ListNode cur = this.first;
	while (cur != null) {
		stream.println(cur.getData());
		cur = cur.getNext();
	}
}
```

Ουρά FIFO:

Η IntQueueImpl<T> έχει ως ιδιότητες το head,tail τύπου ListNode και το size τύπου int. Το head ουσιαστικά είναι ένας pointer προς το πρώτο ListNode της στοίβας, το οποίο περιέχει το πρώτο της αντικείμενο τύπου T,το tail ειναι ενας pointer προς το τελευταίο ListNode. Το size είναι ο αριθμός των αντικειμένων που έχει η στοίβα. Η κλάση αυτή επιπλέον υλοποιεί την διεπαφή IntQueue με μεθόδους : isEmpty(), put(T item), get(), peek(), printQueue(PrintStream stream) και size(). Η εισαγωγή γίνεται με την ακόλουθη μέθοδο:

```
 public void put(T data) {
if(isEmpty()){
```

```
 head=tail=new ListNode<T>(data,null);
 }else{
 tail.setNext(new ListNode<T>(data,null));
 tail = tail.getNext();
}
size++
 }
```

και η αφαίρεση με τη μέθοδο:

```
 public T get() throws NoSuchElementException{
if(isEmpty()){
   throw  new NoSuchElementException();
}
 T v=head.getData();
 ListNode<T> t=head.getNext();
 head=t;
 size--;
return v;
}
```

Και οι 2 μέθοδοι δεν επηρεάζονται από το μέγεθος της στοίβας (έχουν δηλαδή πολυπλοκότητα O(1)). Η υλοποίηση των άλλων μεθόδων είναι οι παρακάτω:

```
public T peek() throws NoSuchElementException{
if(isEmpty()){
  throw  new NoSuchElementException();
 }else{
   return head.getData()
```

```
 }


public int size(){

return size;

}

1. public boolean  isEmpty(){

return head==null;

}


 public void printQueue(PrintStream stream) {
ListNode cur = this.head;

                while (cur != null) {

                    stream.println(cur.getData());

                    cur = cur.getNext();

                }
```

Μέρος Β

Ανοίγουμε το html αρχείο και βρίσκουμε τα tags κάθε γραμμής. Αν το tag είναι της μορφής <tag> το κάνουμε push σε μια στοίβα. Αν είναι της μορφής </tag> ελέγχουμε αν είναι η στοίβα άδεια. Αν είναι, το result γίνεται false, αλλιώς ελέγχουμε αν το tag που επιστρέφει η στοίβα με τη μέθοδο pop() είναι το ταιριαστό ζευγάρι, οπότε το result γίνεται false αν δεν είναι. Επαναλαμβάνουμε τον παραπάνω αλγόριθμο μέχρι την τελευταία γραμμή ή μέχρι να βρούμε μη ταιριαστές ετικέτες.

Μερος Γ

Χρησιμοποιώντας τις έτοιμες μεθόδους της java στο πακετο java.io διαβάζουμε το αρχείο.Ελέγχουμε αν στην γραμμή που διαβάζουμε υπάρχει buy η sell.Αν υπάρχει buy τότε βαζουμε στον πρώτο κόμβο της ουράς τον

αριθμό των μετοχών και στον δεύτερο την τιμή(τον αριθμού που βρίσκεται μετά το price)(δηλαδή η κάθε πληροφορία θα είναι ανά δύο κόμβους)με χρηση της μεθοδου q1.put() 2 φορές .Αν υπάρχει sell,τότε βάζουμε σε μια μεταβλητή τον αριθμό των μετοχών που θέλουμε να πουλήσουμε(metoxes_pwlhshs) και την τιμή που τις πουλάμε(timi_pwlhshs).Ελέγχουμε 3 περιπτώσεις

1η Περίπτωση:Αν το ενα Node που έχουμε(δεν χρειάζεται να συμπεριλάβουμε και το άλλο που κραταει την τιμή των μετοχών διότι οταν τελειώσουμε με τα nodes,τα θέτουμε και τα δυο σε null και ούτως η άλλως έχουν αρχικοποιηθεί ως null)ειναι null και η ουρά που έχουμε δεν ειναι empty(αυτό υπάρχει στην περίπτωση που πουλάμε περισσότερες μετοχές από όσες έχουμε,ετασι ώστε να μην βγάλει NoSuchElement Exeption) ετσι ώστε να βαλουμε στο άθροισμα μας για το τελικό κέρδος ή ζημία την τιμή από την ουρά ελέγχουμε δύο περιπτώσεις,αν οι μετοχές που πουλάμε ειναι περισσότερες από τις μετοχές ,τότε υπολογίζουμε το άθροισμα και αφαιρούμε τις μετοχές που πουλήσαμε.Αν οι μετοχές ειναι λιγότερες βάζουμε αυτές που περίσσεύουν στα 2 nodes ετσι ώστε να τις χρησιμοποιήσουμε πρώτες στις επόμενες αγορές(αφου αγοραστηκαν σε μικρότερη χρονικη στιγμή tbuy1<tbuy2<tbuy3<…<tbuyn).Μαζί με την τιμή της υπολογίζουμε το κέρδος που προκύπτει με τις μετοχές που έχουν μείνει

2η περίπτωση:ελέγχουμε αν το ένα Node δεν είναι άδειο τότε θα δουλέψουμε το Node όπως την πρώτη περίπτωση(το node οταν δεν είναι άδειο,εχει προτεραιότητα,καθώς ειναι οι μετοχές που έχουν περισσέψει και θα τις χρησιμοποιήσουμε πρώτες σε επόμενες αγορές).

3η περίπωση:αν τώρα η ουρά ειναι άδεια δηλαδή έχουν ληφθεί κάποιες μετοχές ή το node είναι άδειο(είτε έχουν ληφθεί είτε δεν έχουν μπει καθόλου και στις 2 περιπτωσεις δεν εχει άλλες μετοχές) τότε θα τυπώνει μήνυμα λάθους στο συγκεκριμένο sell(με έναν ξεχωριστό μετρητή) έτσι ώστε ο χρήστης να γνωρίζει οτι πουλούνται περισσότερες μετοχές απο οτι υπάρχουν στην κατοχή του και σε ποιό sell).

Κάθε sell στο τέλος θα τυπώνει και το συνολικό κέρδος η ζημία(Αν εμφανιστεί τιμή με αρνητικό πρόσημο) η θα τυπώνει οτι λιγότερες μετοχές αν πουλάμε περισσότερες απο όσες έχουμε