Δομές Δεδομένων & Ανάλυση Αλγορίθμων

Ασκήσεις Πράξης: Επανάληψη

Δημοσθένης Σταμάτης

http:/www.iee.ihu.gr/~demos

Τμήμα Μηχανικών Πληροφορικής & Ηλεκτρονικών Συστημάτων

ΑΣΚΗΣΗ

Στην κλάση της συνδεδεμένης λίστας (της Άσκησης Πράξης 3), προσθέστε μία μέθοδο η οποία να διαγράφει το δεύτερο κόμβο, εφόσον αυτός υπάρχει, και να επιστρέφει το περιεχόμενό του. Αν ο κόμβος δεν υπάρχει θα πρέπει η μέθοδος να ρίχνει σχετική εξαίρεση. Δώστε ένα πλήρες πρόγραμμα για τον έλεγχο της ορθής λειτουργίας της μεθόδου σας.

```
public Object removeSecond() throws NoSuchElementException {
    if (size()<2)
      throw new NoSuchElementException("List with less than 2 elements");
    Object obj1 = removeFirst();
    Object obj2 = removeFirst();
    insertFirst(obj1);
    return obj2;
}
public static void main(String[] args)  {
    Student S1 = new Student(111,"onoma1",1,9);
    Student S2 = new Student(222,"onoma2",2,6);
    Student S3 = new Student(333,"onoma3",0,8);
    Student S4 = new Student(444,"onoma4",3,4);

    LinkedList L = new LinkedList();
    L.insertLast(S1);
    L.insertLast(S2);
    L.insertLast(S3);
    L.insertLast(S4);

    Student R=(Student)L.removeSecond();
    System.out.println(R.toString());
}
```

ΑΣΚΗΣΗ

Θεωρήστε ότι δίνεται η δομή δεδομένων **στοίβα (stack)**, η οποία είναι υλοποιημένη με τη βοήθεια πίνακα (κλάση ArrayStack της Ασκησης Πράξης 2). Θεωρήστε επίσης ότι έχει δημιουργηθεί μία στοίβα στην οποία έχει εισαχθεί ένας αριθμός από δεδομένα τύπου φοιτητή. Να γράψετε μία μέθοδο **count8s** η οποία να μετρά πόσοι από τους φοιτητές που βρίσκονται στη στοίβα έχουν βαθμολογηθεί με 8. Δώστε ένα πλήρες πρόγραμμα για τον έλεγχο της ορθής λειτουργίας της μεθόδου σας.

Περιορισμοί: **(α)** Η στοίβα μετά την ολοκλήρωση της καταμέτρησης θα πρέπει να έχει ακριβώς την ίδια μορφή με την αρχική. **(β)** Δεν επιτρέπεται να τροποποιήσετε την κλάση υλοποίησης της στοίβας ArrayStack.

```
public static int count8s(ArrayStack S){
    int count=0;
    int size=S.size();
    ArrayStack SB= new ArrayStack(size);
    for(int i=0;i<size;i++) {
        Object obj = S.top();
        if (((Student)obj).getVathmos()>=8) count++;
        SB.push(S.pop());
    }
    for(int i=0;i<size;i++) {
        S.push(SB.pop());
    }
    return count;
}
```

```
public static void main(String[] args)  {
    Student S1 = new Student(111,"onoma1",1,9);
    Student S2 = new Student(222,"onoma2",2,6);
    Student S3 = new Student(333,"onoma3",0,8);
    Student S4 = new Student(444,"onoma4",3,4);
    ArrayStack ST=new ArrayStack();
    ST.push(S1);
    ST.push(S2);
    ST.push(S3);
    ST.push(S4);
    System.out.println(count8s(ST));
}
```

ΑΣΚΗΣΗ

Δίνεται το interface "LabInterface" που περιγράφει τις βασικές λειτουργίες του Εργαστηριακού Τμήματος (Άσκηση Πράξης 1). Να υλοποιήσετε το "LabInterface" με μία κλάση στην οποία να χρησιμοποιήσετε μία συνδεδεμένη λίστα (όπως αυτή ορίζεται στην Άσκηση Πράξης 3) αντί του πίνακα που είχε χρησιμοποιηθεί αρχικά.

```
public interface LabInterface {

    public void insert(Student S);
    // Εισαγωγή ενός φοιτητή στο εργαστηριακό τμήμα

    public boolean enroll(Student S) throws AlreadyInLabException;
    // Εισαγωγή ενός φοιτητή στο εργαστηριακό τμήμα
    // με αποφυγή διπλοεγγραφής

    public void delete(int AM) throws NoStudentException;
    // Διαγραφή ενός φοιτητή από το τμήμα με βάση τον Αρ. Μητρώου

    public double averageGrade();
    // Βρίσκει και επιστρέφει το μέσο όρο βαθμολογίας των φοιτητών
    // του εργαστηριακού τμήματος

    public Student maxGrade();
    // Βρίσκει και επιστρέφει τον φοιτητή του εργαστηριακού τμήματος
    // με το μεγαλύτερο βαθμό

}
```

```
public class Lab1 implements LabInterface {

    private String name;
    private LinkedList Lab;

    public Lab1(String name) {
        this.name=name;
        Lab = new LinkedList();
    }

    public void insert(Student S){
        Lab.insertLast(S);
    }
```

```
public boolean enroll(Student S) throws AlreadyInLabException {
    if (foundInLab(S))
        throw new AlreadyInLabException("Already enrolled");
    else {
        Lab.insertLast(S);
        return true;
    }
}

private boolean foundInLab(Student S) {
    int size= Lab.size();
    for (int i=0;  i<size; i++) {
        Student St = (Student)Lab.removeFirst();
        Lab.insertLast(St);
        if (S.getAM()==St.getAM())return true;
    }
    return false;
}
```

```
public void delete(int AM) throws NoStudentException {
    int size= Lab.size();
    boolean found = false;
    for (int i=0;  i<size; i++) {
       Student St = (Student)Lab.removeFirst();
       if (St.getAM()==AM)
           found = true;
       else Lab.insertLast(St);
       }
    if (!found)
       throw new NoStudentException("No such Student in Lab "+name);
}
```

```
public double averageGrade() {
    int size= Lab.size();
    if (size==0)return 0;
    double Sum=0.0;
    for (int i=0;  i<size; i++) {
       Student St = (Student)Lab.removeFirst();
       Sum+=St.getVathmos();
       Lab.insertLast(St);
    }
    return Sum/size;
}
```

```
public Student maxGrade() {
    int size= Lab.size();
    if (size==0)return null;
    Student BestStudent=(Student)Lab.removeFirst();
    Lab.insertLast(BestStudent);
    int max= BestStudent.getVathmos();
    for (int i=0;  i<size-1; i++) {
       Student St = (Student)Lab.removeFirst();
       Lab.insertLast(St);
       if (max < St.getVathmos()){
           max=St.getVathmos();
           BestStudent=St;
       }
    }
    return BestStudent;
    }
```

ΑΣΚΗΣΗ

Θεωρήστε ότι δίνεται η δομή δεδομένων **συνδεδεμένη λίστα (linked list)** (όπως είναι υλοποιημένη στην Ασκηση Πράξης 3). Θεωρήστε επίσης ότι έχει δημιουργηθεί μία συνδεδεμένη λίστα στην οποία έχει εισαχθεί ένας αριθμός από δεδομένα τύπου φοιτητή. Να γράψετε μία μέθοδο **succeeded** η οποία να μετρά πόσοι από τους φοιτητές που βρίσκονται στη συνδεδεμένη λίστα έχουν περάσει το μάθημα (βαθμός >=5). Δώστε ένα πλήρες πρόγραμμα για τον έλεγχο της ορθής λειτουργίας της μεθόδου σας.

Περιορισμοί: **(α)** Η λίστα μετά την ολοκλήρωση της καταμέτρησης θα πρέπει να έχει ακριβώς την ίδια μορφή με την αρχική. **(β)** Δεν επιτρέπεται να τροποποιήσετε την κλάση υλοποίησης της συνδεδεμένης λίστας.

```
public static int succeeded(LinkedList L){
    SLListNode current = L.getFirst();
    int count = 0;
    while(current != null){
        if((((Student)current.getNodeData()).getVathmos()) >= 5)
                count++;
        current = current.getNextNode();
    }
    return count;
}

public static void main(String[] args)  {
    Student S1 = new Student(111,"onoma1",1,3);
    Student S2 = new Student(222,"onoma2",2,6);
    Student S3 = new Student(333,"onoma3",0,8);
    Student S4 = new Student(444,"onoma4",3,8);

    LinkedList L = new LinkedList();
    L.insertLast(S1);
    L.insertLast(S2);
    L.insertLast(S3);
    L.insertLast(S4);

    System.out.println(succeeded(L));
}
```

ΑΣΚΗΣΗ

Θεωρήστε ότι δίνεται η δομή δεδομένων **στοίβα (stack)**, η οποία είναι υλοποιημένη με τη βοήθεια συνδεδεμένης λίστας (σχετική κλάση της Ασκησης Πράξης 4). Στην κλάση υλοποίησης προσθέστε μία **boolean μέθοδο** η οποία δοθέντος ενός αντικειμένου να επιστρέφει true εάν αυτό υπάρχει μέσα στη στοίβα και false στην αντίθετη περίπτωση. Δώστε ένα πλήρες πρόγραμμα για τον έλεγχο της ορθής λειτουργίας της μεθόδου σας.

```
public boolean exists(Object item) throws StackEmptyException {
    LinkedStack temp = new LinkedStack();
    boolean found = false;
    int size = size();
    for(int i = 0; i < size; i++) {
        Object obj = pop();
        temp.push(obj);
        if(obj.equals(item)) {
            found = true;
            break;
        }
    }
    while(!temp.isEmpty()) {
        push(temp.pop());
    }
    return found;
}
```

```
public static void main(String[] args)  {
    Student S1 = new Student(111,"onoma1",1,9);
    Student S2 = new Student(222,"onoma2",2,6);
    Student S3 = new Student(333,"onoma3",0,8);
    Student S4 = new Student(444,"onoma4",3,4);
    Student S5 = new Student(555,"onoma5",1,7);

    LinkedStack ST=new LinkedStack();
    ST.push(S1);
    ST.push(S2);
    ST.push(S3);
    ST.push(S4);

    System.out.println(ST.exists(S3));
    System.out.println(ST.exists(S3));
}
```