HY-252 Αντικειμενοστραφής Προγραμματισμός

Χειμερινό Εξάμηνο 2011
Διδάσκων: Χριστοφίδης Βασίλης

1η Σειρά Ασκήσεων

Ημερομηνία Παράδοσης: 28/10/2010

Άσκηση 1 – Loops, IO (10%)
Όνομα αρχείου : `Assign1_username.java`

Γράψτε μια μέθοδο printStarTree(int n) που παίρνει ως παράμετρο έναν ακέραιο n και τυπώνει ένα δέντρο ύψους n από αστεράκια. Την παράμετρο n θα τη περνάει ο χρήστης μέσω της γραμμής εντολών. Για παράδειγμα, η παρακάτω εντολή

```
>java Assign1_username 5
```

θα έχει ως αποτέλεσμα το παρακάτω δέντρο:

```
    *
   ***
  *****
 *******
*********
```

Το πρόγραμμα θα πρέπει να τυπώνει κατάλληλα μηνύματα, σε περίπτωση που υπάρχει λάθος στις παραμέτρους του χρήστη. Συγκεκριμένα, λάθη στις παραμέτρους θεωρούνται τα παρακάτω:

- Ο χρήστης έχει δώσει περισσότερες από μία παραμέτρους ή καμία.
- Η παράμετρος δεν είναι αριθμός.
- Η παράμετρος δεν είναι **θετικός** αριθμός.

Βοήθεια (Hints):

- Χρησιμοποιείστε την `Integer.parseInt(String)` για να μετατρέψετε την είσοδο σε int. Χειριστείτε κατάλληλα το Exception που σχετίζεται με την παραπάνω μέθοδο.

Βαθμολόγηση:

- Έλεγχος για σωστό αριθμό παραμέτρων: 1%
- Έλεγχος για ακέραιο *με exception handling*: 1%
- Έλεγχος για θετικό ακέραιο: 1%
- Μέθοδος printStarTree: 7%

Bonus Ερώτημα (10%)

Δώστε μια αναδρομική υλοποίηση της μεθόδου printStarTree(int n) χωρίς να χρησιμοποιήσετε κανένα επαναληπτικό loop for ή while. Η καινούρια μέθοδος θα ονομάζεται recursivePrintStarTree(int nLimit, int n), όπου το πρώτο όρισμα nLimit είναι το ύψος του δέντρου. Όταν καλείται αρχικά η συνάρτηση, στο όρισμα n βάζετε 0. Σε κάθε αναδρομικό βήμα θα πρέπει να αλλάζετε κατάλληλα αυτό το όρισμα.

Παράδειγμα: Η κλήση recursivePrintStarTree(5, 0) έχει έξοδο:

```
    *
   ***
  *****
 *******
*********
```

Βαθμολόγηση:

- Μέθοδος recursivePrintStarTree: 10%

Άσκηση 2 – String, File, I/O (30%)
Όνομα αρχείου : `Assign2_username.java`

Γράψτε ένα πρόγραμμα που αναζητά λέξεις σε ένα απλό αρχείο κειμένου. Το πρόγραμμά σας θα πρέπει να παίρνει το όνομα ενός αρχείου από τη γραμμή εντολών. Μετά, θα ζητά από τον χρήστη μία λέξη, την οποία θα αναζητά στο κείμενο. Το πρόγραμμα θα πρέπει να βρίσκει λέξεις ανεξάρτητα από το αν είναι γραμμένες με κεφαλαία ή μικρά γράμματα.
Η διαδικασία είναι η εξής: για κάθε λέξη, το πρόγραμμά σας θα ξεκινά να διαβάζει το αρχείο από την αρχή, γραμμή προς γραμμή. Κάθε γραμμή θα κατακερματίζεται σε λέξεις. Για κάθε γραμμή, θα συγκρίνονται οι λέξεις της με τη λέξη που έδωσε ο χρήστης. Αν βρεθεί κάποια λέξη που είναι ίδια, το πρόγραμμα σας θα ενημερώνει το χρήστη για τη γραμμή του κειμένου στην οποία εντοπίστηκε, αλλά δεν θα τερματίζει. Όταν το πρόγραμμα θα έχει σαρώσει ολόκληρο το αρχείο, θα πρέπει να έχουν τυπωθεί στην οθόνη όλες οι γραμμές στις οποίες υπάρχει η ζητούμενη λέξη. Για παράδειγμα, αν το αρχείο περιέχει τα παρακάτω:

```
Auto einai ena arxeio
pou periexei le3eis se
greeklish. Ena tetoio
arxeio, se ena tetoio
periballon, ti nohma exei?
```

και η ζητούμενη λέξη είναι το «ena», τότε το πρόγραμμα θα πρέπει να τυπώσει

```
ena found in line 1
ena found in line 3
ena found in line 4
```

Για ευκολία θεωρούμε πως τα κείμενα θα περιέχουν μόνο λέξεις και κενά, δηλαδή δεν υπάρχουν στα κείμενα ούτε αριθμοί, ούτε σημεία στίξης. Επίσης, θεωρούμε ότι κάθε γραμμή του κειμένου δεν θα έχει πάνω από 20 λέξεις.

Βοήθεια (Hints):

- Για ανάγνωση από αρχείο, χρησιμοποιήστε τις κλάσεις **`java.io.BufferedReader`** και **`java.io.FileReader`**. Χρησιμοποιήστε το μηχανισμό των Exceptions για να γνωρίζετε πότε πρέπει να τυπώνονται κατάλληλα μηνύματα λάθους.
- Η κλάση **`String`** διαθέτει μερικές πολύ χρήσιμες μεθόδους για διαχείριση αλφαριθμητικών. Ανάμεσά τους υπάρχουν μέθοδοι που μετατρέπουν όλα τα κεφαλαία σε μικρά (και το ανάποδο) (δείτε το API της Java για περισσότερες λεπτομέρειες).
- Για τον χωρισμό ενός String σε επιμέρους tokens μπορείτε να χρησιμοποιήσετε την κλάση **`StringTokenizer`** (Λεπτομέρειες εδώ).
- Προκειμένου να χειριστείτε μια εξαίρεση (exception) η οποία είναι πιθανόν να προκληθεί απο κάποιο κομμάτι του κώδικά σας πρέπει να περικλείσετε το κομμάτι αυτό σε ένα try-catch block. Για παράδειγμα:

```
try {
    κομμάτι κώδικα που μπορεί να προκληθεί εξαίρεση
}
catch (τύπος_εξαίρεσης όνομα_αντικειμένου_εξαίρεσης) {
    εντολές χειρισμού της εξαίρεσης
}
```

  Για περισσότερες λεπτομέρειες ανατρέξτε εδώ ή και στο αντίστοιχο φροντιστήριο.

Βαθμολόγηση:

- Έλεγχος ορθότητας παραμέτρων: 4%
- Έλεγχος εισόδου εξόδου: 6%
- Σάρωση κειμένου και αγνόηση κεφαλαίων από αρχείο και System.in: 8%
- String tokenization: 7%
- Έλεγχος για εύρεση ή μη της λέξης: 5%

Άσκηση 3 – Arrays (30%)
Όνομα αρχείου : `Assign3_username.java`

Καλείστε να υλοποιήσετε το πρόγραμμα κονσόλας **`Painter`**, στο οποίο ο χρήστης θα έχει στη διάθεσή του έναν δισδιάστατο χώρο για να σχεδιάζει γραμμές διαφόρων μηκών και ειδών.

Συγκεκριμένα, οι διαστάσεις του δισδιάστατου χώρου θα δίνονται στις παραμέτρους του προγράμματος, το οποίο και θα τον αναπαριστά εσωτερικά με την μορφή ενός πίνακα `char[`*γραμμές*`][`*στήλες*`]`. Ο πίνακας παραμέτρων (`String[] args`) της main θα έχει δύο συμβολοσειρές, που το πρώτο θα είναι το πλήθος των γραμμών και το δεύτερο το πλήθος των

στηλών. Μετά, θα ζητείται από τον χρήστη να εισάγει μία από τις επόμενες τρεις εντολές:

1. **exit**
   Το πρόγραμμα θα τερματίζει.
2. **show**
   Το πρόγραμμα θα τυπώνει τον πίνακα
3. **draw c x0 y0 stepX stepY length**
   Το πρόγραμμα θα πρέπει να σχεδιάσει μία γραμμή, χρησιμοποιώντας τον χαρακτήρα **c**, ξεκινώντας από το σημείο (γραμμή = **x0**, στήλη = **y0**), προχωρώντας σε κάθε βήμα με **stepX** και **stepY** αντίστοιχα, για **length** βήματα.

Ένα παράδειγμα εισόδων σας δίνεται στη συνέχεια. Για παράδειγμα, θεωρήστε την παρακάτω εντολή:

draw * 2 2 0 1 5

Η σημασία των 6 ορισμάτων είναι:
`args[0]` == "*": ο χαρακτήρας που θα χρησιμοποιηθεί για να «ζωγραφιστεί» η γραμμή.
`args[1]` == "2": η γραμμή του πίνακα από την οποία ξεκινάει να σχεδιάζεται η γραμμή (εδώ ξεκινάμε από την τρίτη γραμμή).
`args[2]` == "2": η στήλη του πίνακα από την οποία ξεκινάει να σχεδιάζεται η γραμμή (εδώ ξεκινάμε από την τρίτη στήλη).
`args[3]` == "0": βήμα γραμμής: «σε κάθε βήμα, το επόμενο σημείο θα είναι στην ίδια (+0) γραμμή από το προηγούμενο σημείο που ζωγραφίστηκε».
`args[4]` == "1": βήμα στήλης: «σε κάθε βήμα, το επόμενο σημείο θα είναι +1 στήλη από το προηγούμενο σημείο που ζωγραφίστηκε».
`args[5]` == "5": Οι φορές οι οποίες θα επιχειρηθεί να «ζωγραφιστεί» ο χαρακτήρας (Υπάρχει περίπτωση να πέσει εκτός ορίων του πίνακα κι έτσι να μην ζωγραφιστεί).

Η παραπάνω εντολή θα έχει ως αποτέλεσμα να τυπωθεί μία γραμμή με τον χαρακτήρα '*', στα σημεία (2, 2), (2, 3), (2, 4), (2, 5), (2, 6) του πίνακα (τα περιεχόμενα του πίνακα δε θα τυπώνονται στην οθόνη παρά μόνο μετά την σχετική εντολή του χρήστη). Θα πρέπει να γίνεται έλεγχος ορίων σε κάθε σημείο, έτσι ώστε να μην προσπελαστούν σημεία εκτός του δισδιάστατου πίνακα (διαφορετικά, αν γίνει κάτι τέτοιο και δεν υπάρχει επαρκής έλεγχος, θα δημιουργηθεί μία εξαίρεση κατά τη διάρκεια της εκτέλεσης και το πρόγραμμα θα τερματιστεί πρόωρα). Αν εναλλακτικά δοθεί στην είσοδο η εντολή **draw % 4 4 -1 -1 3,** τότε θα πρέπει να τυπωθεί το σύμβολο '%' στα σημεία: `(4, 4), (3, 3), (2, 2).`
Αν ο πίνακας έχει διαστάσεις 5x5, και η είσοδος είναι **draw @ -3 -1 2 1 6,** τότε τα σημεία θα ζωγραφιστούν είναι:

(-3, -1) --- εκτός του πίνακα, αγνοείται
(-1, 0) --- εκτός του πίνακα, αγνοείται
(1, 1) --- αποδεκτό
(3, 2) --- αποδεκτό
(5, 3) --- εκτός του πίνακα, αγνοείται
(7, 4) --- εκτός του πίνακα, αγνοείται

Ένα παράδειγμα εκτέλεσης του προγράμματος **Painter** για έναν χώρο 5 γραμμών επί 5 στηλών, σας δίνεται στη συνέχεια:

```
draw * 0 2 0 1 5
draw % 1 0 1 1 3
draw @ -3 -1 2 1 6
draw # 1 4 0 1 3
show
```

```
|-----|
|  ***|
|%@  #|
| %   |
|  @  |
|     |
|-----|
```

Δηλαδή, δημιουργούμε ένα πλαίσιο γύρω από τον δισδιάστατο χώρο, με χαρακτήρες '-' και '|', και μέσα σε αυτόν τυπώνουμε τα περιεχόμενα του χώρου.

Βοήθεια (Hints): Αν έχετε ένα String από το οποίο θέλετε να πάρετε τα κομμάτια του αγνοώντας τα κενά (δηλαδή από ένα string "cat dog elephant" να πάρετε "cat", "dog", "elephant"), μπορείτε χρησιμοποιήσετε τη μέθοδο split του `String`.

```
String s = "cat dog elephant";
String[] parts = s.split(" "); //υπάρχει ακριβώς ένα space στην παράμετρο
//parts[0] είναι "cat"
//parts[1] είναι "dog"
//parts[2] είναι "elephant"
```

Αν υπάρχουν δυο διαδοχικά κενά, θα πάρετε ως αποτέλεσμα ένα κενό string (μηδενικού μήκους). Αν θέλετε να χωρίσετε ένα string με κενά, αγνοώντας τα διαδοχικά κενά και χειριζόμενοι όλα τα είδη κενών (tabs, newlines), μπορείτε να χρησιμοποιήσετε τις παρακάτω εντολές:

```
String s = "cat         \t    dog   \n elephant";
String[] parts = s.split("\\s+");
//parts[0] είναι "cat"
//parts[1] είναι "dog"
//parts[2] είναι "elephant"
```

Αν θέλετε να χωρίσετε το String με κάποιο άλλο διαχωριστικό (delimiter), τότε κάνετε το ίδιο, πχ: "cat###dog####elephant".split("###") → Δίνει ["cat", "dog", "#elephant"]

Βαθμολόγηση:

- Έλεγχος ορθότητας παραμέτρων: 5%
- Έλεγχος ορίων πίνακα: 3%
- Αναγνώριση και parse εντολής: 8%

- «Ζωγράφισμα» γραμμής: 7%
- Σωστή εμφάνιση πίνακα: 7%

Άσκηση 4 – Recursion (30%)
Όνομα αρχείου : Assign4_username.java

Η απόσταση μεταξύ δυο bit strings μεγέθους n είναι ίση με τον αριθμό των bits στα οποία διαφέρουν τα δυο bit strings. Γράψτε ένα πρόγραμμα που διαβάζει έναν ακέραιο k και ένα bit string s από τη γραμμή εντολών και τυπώνει όλα τα bit strings που έχουν απόσταση k από το s. Για παράδειγμα, η παρακάτω εντολή:

```
>java Assign4_username 2 0000
```

θα έχει ως αποτέλεσμα:

```
0011 0101 0110 1001 1010 1100
```

Το πρόγραμμα θα πρέπει να τυπώνει κατάλληλα μηνύματα, σε περίπτωση που υπάρχει λάθος στις παραμέτρους του χρήστη.

Βοήθεια (Hint): βρείτε όλους τους συνδιασμούς μεγέθους k από n θέσεις και αντιστρέψτε τα bits στις θέσεις αυτές. Ένας συνδιασμός είναι ένα υποσύνολο n στοιχείων ανεξαρτήτως σειράς. Το *πλήθος* C(n, k) των συνδιασμών μεγέθους k από n στοιχεία δίνεται από τον εξής αναδρομικό τύπο:

C(n, k) = C(n - 1, k - 1) + C(n - 1, k) για n, k >0
C(n, 0) = 1 για n >= 0
C(0, k) = 0 για k > 0

ο οποίος προκύπτει μετρώντας τους συνδιασμούς μεγέθους k του συνόλου {1, 2, ..., n} που περιέχουν και δεν περιέχουν το n αντίστοιχα.

Βαθμολόγηση:

- Έλεγχος για σωστό αριθμό παραμέτρων: 2%
- Έλεγχος για μη αρνητικό ακέραιο k (έλεγχος για ακέραιο k με *exception handling*): 5%
- Έλεγχος για bit string s: 3%
- *Αναδρομική* εύρεση των bit strings με απόσταση k: 20%