Δεύτερη Σειρά ασκήσεων
Ημερομηνία Παράδοσης: 30 Μαΐου 2018, 5 μ.μ.

Οι παρακάτω ασκήσεις πρέπει να παραδοθούν μέχρι τις 30/5/2018 στις 5 μ.μ. Την πρώτη άσκηση θα την κάνετε turnin στο assignment2.1@myy205 και την δεύτερη στο assignment2.2@myy205. Και για τις δύο ασκήσεις παραδώσετε ένα αρχείο report.txt που περιγράφετε τι έχετε κάνει και πώς να τρέξουμε τον κώδικα σας. Η αναφορά θα είναι ιδιαίτερα σημαντική για την δεύτερη άσκηση όπου μπορεί να έχετε κάνει μόνο κάποιο από τα τρία μέρη. Στην αναφορά σας πρέπει να αναφέρετε και τα προβλήματα του κώδικα σας: σε ποιες περιπτώσεις δεν δουλεύει σωστά. Εργασίες στις οποίες τα προβλήματα του κώδικα περιγράφονται με ακρίβεια θα πάρουν +10% του βαθμού που θα παίρνανε κανονικά.

Άσκηση 1

Για την ερώτηση αυτή θα πρέπει να τροποποιήσετε την υλοποίηση που κάνατε για το Μέρος 2 της πρώτης σειράς ασκήσεων ώστε να δουλεύει με κληρονομικότητα. Συγκεκριμένα:

- Θα κάνετε την κλάση **ShipBoard** να είναι μια αφηρημένη κλάση που παράγει δύο νέες κλάσεις: την κλάση **HumanShipBoard** και την κλάση **ComputerShipBoard** ανάλογα αν έχουμε ένα Computer ή Human παίχτη. Η κλάση θα έχει την αφηρημένη μέθοδο **enterAllShips** η οποία θα τοποθετεί όλα τα πλοία. Οι παραγόμενες κλάσεις θα υλοποιούν την enterAllShips και θα τοποθετούν τα πλοία στον πίνακα, χειρωνακτικά ή τυχαία.
- Θα υλοποιήσετε μια αφηρημένη κλάση **Player** η οποία θα παράγει τις κλάσεις **HumanPlayer** και **ComputerPlayer**. Η κλάση θα έχει τις αφηρημένες μεθόδους **createBoard** η οποία θα καλείται στον constructor και **nextStrike**. Οι παραγόμενες κλάσεις θα υλοποιούν μόνο αυτές τις δύο μεθόδους. Στην κλάση Battleship θα έχετε πλέον μόνο μια βοηθητική μέθοδο **playGame** η οποία θα παίρνει σαν όρισμα δύο Player αντικείμενα και θα υλοποιεί το παιχνίδι τους. Η main ανάλογα με την επιλογή του χρήστη θα δημιουργεί τα κατάλληλα αντικείμενα και θα καλεί την playGame.

Η βαθμολόγηση της άσκησης θα γίνει με κριτήριο αν χρησιμοποιείτε σωστά τις ιδέες της κληρονομικότητας και όχι αν έχετε υλοποιήσει σωστά τα επιμέρους κομμάτια της πρώτης σειράς. Ο κώδικας θα πρέπει φυσικά να κάνει compile. Βοηθητικά σας δίνεται μια υλοποίηση της RandomStrategy.

Άσκηση 2

Ο στόχος της άσκησης αυτής είναι να εξασκηθείτε με διάφορες τεχνικές που μάθαμε στην τάξη: σύνθεση κλάσεων, κληρονομικότητα, χρήση δομών δεδομένων. Για το σκοπό αυτό θα υλοποιήσετε μια απλή μηχανή αναζήτησης για επιστημονικές δημοσιεύσεις. Η υλοποίηση σας θα αποτελείται από τρία μέρη. Στο πρώτο θα διαβάσετε από αρχείο τα δεδομένα και θα τα αποθηκεύσετε στις κατάλληλες δομές. Στο δεύτερο θα δημιουργήσετε ένα ευρετήριο (index) ώστε να μπορείτε να κάνετε αναζήτηση στα δεδομένα με λέξεις κλειδιά. Στο τρίτο θα δημιουργήσετε το σύστημα το οποίο θα κάνει τις ερωτήσεις και θα επιστρέφει τα αποτελέσματα.

Μέρος 1ο

Στο πρώτο κομμάτι θα υλοποιήσετε τις απαραίτητες κλάσεις που θα αποθηκεύουν τις διαφορετικές εγγραφές στην μηχανή αναζήτησης. Έχουμε τριών ειδών εγγραφές: **Paper** για μία δημοσίευση, **Researcher** για ένα συγγραφέα μιας δημοσίευσης, **Conference** για ένα συνέδριο στο οποίο έγινε μια δημοσίευση. Επίσης θα υλοποιήσετε το κομμάτι που διαβάζει το αρχείο και δημιουργεί μια βάση με τις εγγραφές. Συγκεκριμένα θα υλοποιήσετε τις παρακάτω κλάσεις:

Entry: Η κλάση Entry είναι αφηρημένη και αναφέρεται σε οποιαδήποτε μορφή εγγραφής η οποία συσχετίζεται με κείμενο. Έχει μόνο την αφηρημένη μέθοδο **display()** που δεν επιστρέφει τιμή και δεν παίρνει ορίσματα. (Η Entry μπορεί να οριστεί και ως interface αλλά στο δεύτερο μέρος θα χρειαστεί να της προσθέσουμε πεδία και επιπλέον μεθόδους).

Paper: Η κλάση κρατάει πληροφορία για μία δημοσίευση και κληρονομεί από την κλάση Entry. Έχει πεδία τον τίτλο της δημοσίευσης (**title**), τη χρονιά της δημοσίευσης (**year**), ένα ArrayList **authorList** με Researcher αντικείμενα για τους συγγραφείς της δημοσίευσης, και ένα πεδίο Conference (**conf**) για το συνέδριο της δημοσίευσης. Ο τίτλος και ο χρόνος αρχικοποιούνται στον constructor. Ορίστε επίσης μία μέθοδο **addAuthor** που προσθέτει ένα συγγραφέα. Ορίστε επίσης την **toString** η οποία επιστρέφει όλη την πληροφορία της δημοσίευσης σε μία γραμμή, και την **display** η οποία τυπώνει τις πληροφορίες της δημοσίευσης με περισσότερες λεπτομέρειες. Έτσι π.χ., για μια δημοσίευση με τίτλο "Data mining applications" από τους "Clark Kent" και "Bruce Wayne" στο συνέδριο KDD την χρονιά 2018 η toString θα επιστρέφει το String "*Data mining applications. Clark Kent, Bruce Wayne, KDD 2018*". Η display θα τυπώσει:
Paper:Data mining applications
Authors:Clark Kent,Bruce Wayne,
Conference:KDD 2018

Researcher: Η κλάση κρατάει πληροφορία για ένα ερευνητή/συγγραφέα και κληρονομεί από την κλάση Entry. Έχει πεδία το όνομα του ερευνητή (**name**), και ένα ArrayList **paperList** με Paper αντικείμενα με τις δημοσιεύσεις του συγγραφέα. Το όνομα αρχικοποιείται στον constructor. Ορίστε επίσης μία μέθοδο **addPaper** που προσθέτει μια δημοσίευση. Ορίστε επίσης την **toString** η οποία επιστρέφει το όνομα του ερευνητή, και την **display** η οποία τυπώνει το όνομα του ερευνητή και τη λίστα με τις δημοσιεύσεις του. Για τον ερευνητή Clark Kent με δύο δημοσιεύεις η display θα τυπώσει:
Researcher:Clark Kent
Papers:
"Data mining applications". Clark Kent,Bruce Wayne, KDD 2018
"Theory of data mining". Clark Kent,Peter Parker, KDD 2017

Conference: Η κλάση κρατάει πληροφορία για ένα συνέδριο και κληρονομεί από την κλάση Entry. Έχει πεδία το όνομα του συνεδρίου (**name**), και ένα ArrayList **paperList** με Paper αντικείμενα με τις δημοσιεύσεις στο συνέδριο. Το όνομα αρχικοποιείται στον constructor. Ορίστε επίσης μία μέθοδο **addPaper** που προσθέτει μια δημοσίευση. Ορίστε επίσης την **toString** η οποία επιστρέφει το όνομα του συνεδρίου, και την **display** η οποία τυπώνει το όνομα του συνεδρίου και τη λίστα με τις δημοσιεύσεις. Για το συνέδριο KDD με δύο δημοσιεύεις η display θα τυπώσει:
Conference:KDD
Papers:
"Data mining applications". Clark Kent,Bruce Wayne, KDD 2018
"Theory of data mining". Clark Kent,Peter Parker, KDD 2017

Database: Η κλάση κρατάει πληροφορίες για μια «βάση» από πληροφορίες για δημοσιεύσεις. Θα έχει πεδίο ένα ArrayList με αντικείμενα Entries που θα έχει όλες τις δημοσιεύσεις, συγγραφείς και συνέδρια. Τα αντικείμενα για τους συγγραφείς και τα συνέδρια θα τα έχετε επίσης και σε ένα HashMap με κλειδί το όνομα τους.
Η κλάση θα έχει μια μέθοδο **createDB** η οποία παίρνει σαν όρισμα το όνομα ενός αρχείου και διαβάζει από εκεί τις πληροφορίες για τις δημοσιεύσεις. Μία δημοσίευση είναι σε τέσσερεις γραμμές του αρχείου: η πρώτη είναι ο τίτλος, η δεύτερη η λίστα των συγγραφέων χωρισμένη με κόμμα, η τρίτη το συνέδριο και η τέταρτη η χρονιά. Ένα παράδειγμα του αρχείου σας δίνεται στο "toy.txt".
Η createDB θα πρέπει να διαβάσει το αρχείο και να δημιουργήσει ένα αντικείμενο για κάθε δημοσίευση, για κάθε διαφορετικό ερευνητή/συγγραφέα, και για κάθε διαφορετικό συνέδριο. Ορίστε βοηθητικές μεθόδους αν το χρειάζεστε για να χειριστείτε την δημιουργία των διαφορετικών αντικειμένων.
Η κλάση θα έχει επίσης μια μέθοδο **printDB** η οποία τυπώνει όλα τα αντικείμενα στο ArrayList καλώντας την display.
Δημιουργήστε και μία μέθοδο **main** η οποία παίρνει σαν όρισμα γραμμής εισόδου το όνομα του αρχείου, δημιουργεί την βάση και τυπώνει τα περιεχόμενα της. Ένα παράδειγμα εξόδου για το αρχείο toy.txt σας δίνεται στο αρχείο output1.txt.
Σημείωση: Το διάβασμα αρχείων θα το μάθουμε στα επόμενα μαθήματα. Μέχρι τότε μπορείτε να δίνεται τις πληροφορίες για τις δημοσιεύσεις μέσω standard input (μπορείτε να χρησιμοποιήσετε ανακατεύθυνση).
Υπόδειξη: Χρησιμοποιήστε την μέθοδο split της κλάσης String για να σπάσετε το κείμενο σε λέξεις, και την μέθοδο trim για να αφαιρέσετε κενά στην αρχή και το τέλος ενός String.

Μέρος 2ο

Στο δεύτερο κομμάτι θα υλοποιήσετε το ανάστροφο ευρετήριο που θα χρησιμοποιεί η μηχανή αναζήτησης. Ένα ανάστροφο ευρετήριο για κάθε λέξη κρατάει τις εγγραφές στις οποίες εμφανίζεται η λέξη. Για την υλοποίηση θα χρειαστεί να τροποποιήσετε τις κλάσεις που δημιουργήσατε στο πρώτο μέρος και να ορίσετε νέες.

Entry: Η κλάση Entry θα κρατάει όλες τις λέξεις που εμφανίζονται σε μια εγγραφή και την συχνότητα τους. Αυτές είναι οι λέξεις που θα μπουν στο ευρετήριο. Θα έχει ένα πεδίο HashMap **tokenMap** που για κάθε λέξη θα κρατάει τον αριθμό των εμφανίσεων της. Επίσης θα έχει τις εξής μεθόδους:

- **addText** η οποία παίρνει σαν όρισμα ένα κείμενο, το κανονικοποιεί μετατρέποντας τα πάντα σε μικρά (χρησιμοποιείστε τη μέθοδο lowerCase), το σπάει σε λέξεις και ενημερώνει κατάλληλα το tokenMap.
- **getTokens** η οποία επιστρέφει όλες τις λέξεις-κλειδία στο tokenMap είτε ως σύνολο, είτε ως πίνακα.

Database: Για κάθε είδους εγγραφή κρατάμε/προσθέτουμε διαφορετικό κείμενο. Για τις δημοσιεύσεις κρατάμε τον τίτλο, τα ονόματα των συγγραφέων, το όνομα του συνεδρίου και την χρονολογία (ως String). Για τους συγγραφείς κρατάμε το όνομα τους και τους τίτλους όλων των δημοσιεύσεων στις οποίες συμμετέχουν. Για τα συνέδρια κρατάμε μόνο το όνομα τους. Τροποποιήστε την createDB ώστε να προσθέτει (addText) το κατάλληλο κείμενο σε κάθε νέο Entry που δημιουργεί. Η προσθήκη μπορεί να γίνει και στις κλάσεις που κληρονομούν από την Entry.

Index: Η κλάση υλοποιεί το ευρετήριο για την μηχανή αναζήτησης. Έχει σαν πεδίο ένα HashMap **index** με κλειδιά τις λέξεις, και τιμές ένα σύνολο (HashSet) από Entry αντικείμενα στα οποία εμφανίζονται οι λέξεις. Έχει τις εξής μεθόδους:

- **indexDB:** Παίρνει σαν όρισμα την βάση και για κάθε εγγραφή στην βάση ενημερώνει κατάλληλα το index για τις λέξεις στο tokenMap της εγγραφής.
- **printIndex:** Τυπώνει την πληροφορία στο ευρετήριο, δηλαδή για κάθε λέξη τις εγγραφές που σχετίζονται με αυτή την λέξη.

Προσθέσετε και μια μέθοδο main η οποία να παίρνει σαν όρισμα γραμμής εισόδου το όνομα του αρχείου, δημιουργεί την βάση, φτιάχνει το ευρετήριο και το τυπώνει. Ένα παράδειγμα εξόδου για το αρχείο toy.txt σας δίνεται στο αρχείο output2.txt.

Μέρος 3ο

Στο τρίτο κομμάτι θα υλοποιήσετε το σύστημα που θέτει ερωτήματα στην μηχανή αναζήτησης. Για το σκοπό αυτό θα κάνετε τροποποιήσεις στις υπάρχουσες κλάσεις και να υλοποιήσετε επιπλέον κλάσεις.

Entry: Προσθέστε στην κλάση Entry την μέθοδο **computeScore** η οποία παίρνει σαν όρισμα το ερώτημα, το σπάει σε λέξεις και επιστρέφει σαν σκορ το άθροισμα των συχνοτήτων των λέξεων στην εγγραφή.

Paper: Υπερβείτε την **computeScore** ώστε αν το ερώτημα ταυτίζεται με τον τίτλο της δημοσίευσης να προσθέτει επιπλέον +100 στο σκορ, ενώ αν το ερώτημα περιέχει τον τίτλο της δημοσίευσης να προσθέσει επιπλέον +50 στο σκορ.

Researcher: Υπερβείτε την **computeScore** ώστε αν το ερώτημα ταυτίζεται με το όνομα του ερευνητή να προσθέτει επιπλέον +100 στο σκορ.

Conference: Υπερβείτε την **computeScore** ώστε αν το ερώτημα ταυτίζεται με το όνομα του συνεδρίου να προσθέτει επιπλέον +100 στο σκορ.

Index: Προσθέσετε την μέθοδο **retrieve** η οποία παίρνει σαν όρισμα μια λέξη και επιστρέφει το σύνολο των εγγραφών (Entry) που περιέχουν αυτή την λέξη.

Result: (Προαιρετικά) Μια κλάση που απλά κρατάει μία εγγραφή και το σκορ της εγγραφής. Έχει και τις κατάλληλες μεθόδους πρόσβασης.

QueryProcessor: Η κλάση αυτή υλοποιεί το front-end της μηχανής αναζήτησης. Αρχικοποιείται με το ευρετήριο. Είναι υπεύθυνη να κάνει τα παρακάτω:

- Ζητάει από τον χρήστη το ερώτημα, παίρνει το ερώτημα, το κανονικοποιεί και το σπάει σε λέξεις.
- Για κάθε λέξη τραβάει τις εγγραφές, και τις συγχωνεύει ώστε να έχουμε μόνο τις εγγραφές που έχουν όλες τις λέξεις.
- Για κάθε εγγραφή υπολογίζει το σκορ της και δημιουργεί μια ταξινομημένη λίστα με τα αποτελέσματα σε φθίνουσα σειρά των σκορ.
- Τυπώνει τα αποτελέσματα

Δημιουργήστε μεθόδους για να υλοποιήσετε τα παραπάνω βήματα και μια μέθοδο **run** που τρέχει όλα τα βήματα.

SearchEngine: Υλοποιεί την μηχανή αναζήτησης. Έχει μόνο την μέθοδο main η οποία να παίρνει σαν όρισμα γραμμής εισόδου το όνομα του αρχείου, δημιουργεί την βάση, φτιάχνει το ευρετήριο, δημιουργεί το αντικείμενο QueryProcessor και καλεί την run. Παράδειγμα της λειτουργίας της μηχανής αναζήτησης δίνεται στο output3.txt.

<u>Οδηγίες υλοποίησης</u>

Στην υλοποίηση των κλάσεων σας όλα τα πεδία θα ορίζονται private. Βαθμοί θα αφαιρεθούν για προγράμματα που δεν είναι καλά γραμμένα, δηλαδή δεν είναι σωστά στοιχισμένα ή δεν έχουν καλά επιλεγμένα ονόματα μεταβλητών ώστε να διαβάζονται εύκολα. Προγράμματα που δεν κάνουν compile θα πάρουν το πολύ 1 μονάδα από τις 10. Θα πάρετε περισσότερες μονάδες αν έχετε ολοκληρωμένες σωστές μεθόδους, ή μια ολόκληρη κλάση η οποία είναι σωστή, παρά αν έχετε πολύ κώδικα που όμως δεν κάνει compile ή δεν τρέχει.

Τα αρχεία που θα παραδώσετε θα ελεγχθούν για αντιγραφές. Οι αντιγραφές μηδενίζονται.