Πρώτη Σειρά ασκήσεων
Ημερομηνία Παράδοσης: Παρασκευή 4 Απριλίου 11:59μ.μ.

Οι παρακάτω ασκήσεις πρέπει να παραδοθούν μέχρι τις 4 Απριλίου τα μεσάνυχτα. Στην υλοποίηση των κλάσεων σας δεν θα πρέπει να έχετε public μεταβλητές. 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.

Άσκηση 1

Για την άσκηση αυτή θα υλοποιήσετε σε Java τον Αφηρημένο Τύπο Δεδομένων **ταξινομημένος πίνακας**. Ο ταξινομημένος πίνακας είναι ένας πίνακας, ο οποίος αποθηκεύει στοιχεία σειριακά, ταξινομημένα ως προς κάποιο κριτήριο ταξινόμησης και επιτρέπει μια σειρά από λειτουργίες για την πρόσβαση των στοιχείων του. Ο ταξινομημένος πίνακας έχει σταθερή χωρητικότητα (capacity), που είναι ο μέγιστος αριθμός στοιχείων που μπορεί να αποθηκεύσει. Η ιδιότητα της ταξινόμησης διατηρείται μέσω της σωστής υλοποίησης της εισαγωγής και αφαίρεσης των στοιχείων.

Κατασκευάσετε μια κλάση **SortedArray** που υλοποιεί ένα ταξινομημένο πίνακα από ακεραίους, ταξινομημένο σε φθίνουσα σειρά. Η κλάση θα πρέπει να έχει τις εξής μεθόδους:

- Ένα Constructor χωρίς ορίσματα που αρχικοποιεί τον πίνακα με χωρητικότητα capacity = 100.
- Ένα Constructor που αρχικοποιεί τον πίνακα με όρισμα την χωρητικότητα του πίνακα.
- Μια **βοηθητική** μέθοδο **find(int x)** η οποία παίρνει σαν όρισμα ένα ακέραιο x και αν η τιμή x περιέχεται μέσα στον πίνακα επιστρέφει την θέση της, αλλιώς επιστρέφει την θέση στην οποία πρέπει να τοποθετηθεί το x. Αν το x εμφανίζεται πάνω από μία φορά, οποιαδήποτε από τις θέσεις του είναι αποδεκτή.
- Μία μέθοδο **containsElement(int x)** η οποία παίρνει σαν όρισμα ένα ακέραιο x και επιστρέφει true αν υπάρχει στον πίνακα και false αλλιώς.
- Μία μέθοδο **insert(int x)** η οποία παίρνει σαν όρισμα ένα ακέραιο x και εφόσον υπάρχει χώρος τον προσθέτει στην κατάλληλη θέση του πίνακα, και μετακινεί κατάλληλα όσα στοιχεία χρειάζεται. Επιστρέφει μία Boolean τιμή αν έγινε σωστή η προσθήκη του στοιχείου.
- Μία μέθοδο **delete(int x)** η οποία παίρνει σαν όρισμα ένα ακέραιο x και αν υπάρχει στον πίνακα το αφαιρεί, και μετακινεί κατάλληλα όσα στοιχεία χρειάζεται. Αν υπάρχουν πολλαπλές εμφανίσεις του x αφαιρεί μία από αυτές. Επιστρέφει μία Boolean τιμή αν έγινε σωστή η αφαίρεση του στοιχείου.
- Μια μέθοδο **get(int i)** η οποία επιστρέφει το στοιχείο στη θέση i.
- Τις μεθοδους **getSize()** και **getCapacity()** που επιστρέφουν το μέγεθος (αριθμό στοιχείων) και τη χωρητικότητα του πίνακα.
- Τις μεθόδους **min()** και **max()** που επιστρέφουν το μέγιστο και ελάχιστο στοιχείο του πίνακα.
- Την μέθοδο **toString()** η οποία επιστρέφει τα στοιχεία του πίνακα ταξινομημένα χωρισμένα με κενό.
- Την μέθοδο **equals(SortedArray other)** η οποία ελέγχει αν ο πίνακας έχει τα ίδια στοιχεία με τον other.

Ορίστε μια μέθοδο main για να τεστάρετε την κλάση σας. Για να σας βαθμολογήσουμε θα φτιάξουμε μια κλάση **SortedArrayTest** που θα χρησιμοποιεί την SortedArray που θα παραδώσετε. Ένα παράδειγμα δίνεται στην σελίδα του μαθήματος.

Υποδείξη

- Για να πάρετε όλους τους βαθμούς θα πρέπει να φροντίσετε και τις οριακές καταστάσεις (δηλαδή τις περιπτώσεις που μπορεί να δημιουργηθεί κάποιο λάθος, π.χ. να προσπαθήσουμε να διαβάσουμε κάποια θέση εκτός ορίων ενός πίνακα).

Άσκηση 2

Για την άσκηση αυτή θα υλοποιήσετε σε Java ένα πρόγραμμα που θα υλοποιεί μια εκδοχή του παιχνιδιού MineSweeper. Παρόμοια με το γνωστό παιχνίδι, υπάρχει ένα ναρκοπέδιο το οποίο αποτελείται από ένα δισδιάστατο πλέγμα στο οποίο υπάρχουν κρυμμένες νάρκες. Ο παίκτης μπορεί να ανοίξει ένα κελί του πλέγματος και να πληροφορηθεί αν υπάρχει νάρκη, ή τον μετρητή του κελιού, δηλαδή πόσες νάρκες υπάρχουν γειτονικά του κελιού (ένα κελί που δεν είναι στα άκρα έχει 8 γειτονικά κελιά). Σε αντίθεση με την καθιερωμένη εκδοχή του παιχνιδιού στην εκδοχή που θα υλοποιήσετε υπάρχουν δύο παίχτες και ο στόχος τους είναι να βρούνε τις νάρκες. Κάθε φορά που κάποιος παίχτης ανοίγει ένα κελί με νάρκη κερδίζει ένα πόντο. Αν δεν έχει νάρκη το κελί αποκαλύπτεται ο μετρητής του. Οι παίχτες παίζουν εναλλάξ και το παιχνίδι τελειώνει όταν δεν υπάρχουν άλλες νάρκες να αποκαλυφθούν. Κερδίζει ο παίχτης που έχει αποκαλύψει τις περισσότερες νάρκες.

Για την υλοποίηση σας θα πρέπει να δημιουργήσετε 3 κλάσεις. Την κλάση **MineField** η οποία κρατάει πληροφορία για το πλέγμα με τις νάρκες, την κλάση **Player** που υλοποιεί το παιχνίδι του κάθε παίκτη και την κλάση **MineSweeper** που υλοποιεί την ροή του παιχνιδιού.

MineField: Η πιο σημαντική κλάση κρατάει πληροφορίες για το που βρίσκονται οι νάρκες και τους μετρητές για κάθε κελί. Κρατήστε το πλέγμα σε ένα δισδιάστατο πίνακα με ακέραιους. Η τιμή -1 σημαίνει ότι υπάρχει νάρκη. Αν δεν υπάρχει νάρκη ο πίνακας κρατάει τον μετρητή του κελιού. Για την άσκηση σας θα υλοποιήσετε ένα πλέγμα μεγέθους 10×10, στο οποίο θα τοποθετήσετε 20 νάρκες. Χρειάζεστε επίσης πληροφορία για το ποια κελιά του πλέγματος έχουν «ανοίξει» οι παίκτες, πόσες νάρκες υπάρχουν συνολικά και πόσες έχουν βρεθεί.
Η κλάση θα πρέπει να έχει τις εξής μεθόδους:

- Τον **constructor** ο οποίος θα αρχικοποιεί το πλέγμα, δηλαδή θα τοποθετεί τις νάρκες σε τυχαίες θέσεις μέσα στο πλέγμα, και θα υπολογίζει τους μετρητές για κάθε κελί στο οποίο δεν υπάρχει νάρκη. Υλοποιήστε δύο **βοηθητικές** μεθόδους για κάθε μια από τις δύο αυτές λειτουργίες, μια για να τοποθετεί τις νάρκες και μία για να υπολογίζει τους μετρητές. Σημείωση: δεν μπορείτε να βάλετε δύο νάρκες στο ίδιο κελί.
- Υπερφορτώστε τον **constructor** ώστε να παίρνει τον αριθμό των ναρκών σαν όρισμα.
- Την μέθοδο **check** η οποία παίρνει σαν όρισμα τις συντεταγμένες ενός κελιού και επιστρέφει true ή false, ανάλογα με το αν υπάρχει νάρκη ή όχι. Η μέθοδος αυτή αντιστοιχεί το άνοιγμα ενός κελιού από τον παίχτη. Θα πρέπει να γίνεται έλεγχος αν οι συντεταγμένες είναι εντός των ορίων και αν το κελί αυτό έχει ανοιχθεί παλαιότερα. Σε αυτή την περίπτωση επιστρέφει false.
- Την μέθοδο **print**, η οποία τυπώνει το πλέγμα. Για κάθε ανοιχτό κελί τυπώνει την τιμή του, ενώ για κάθε κλειστό κελί τυπώνει 'x'. Στην ίδια γραμμή τα γειτονικά κελιά χωρίζονται με tab. (Αν έχετε κάποιο καλύτερο τρόπο να αναπαραστήσετε το πλέγμα μπορείτε να το υλοποιήσετε).
- Την μέθοδο allMinesFound η οποία επιστρέφει true αν όλα τα κελιά με νάρκη έχουν ανοιχτεί.

Player: Η κλάση αυτή κρατάει πληροφορία για κάθε παίκτη και υλοποιεί το παιχνίδι του. Για κάθε παίκτη χρειαζόμαστε ένα όνομα (αρχικοποιείται στον constructor) και το score του, δηλαδή τον αριθμό των ναρκών που έχει βρει μέχρι τώρα.
Η βασική μέθοδος της κλάσης είναι η **play**, η οποία παίρνει σαν όρισμα ένα αντικείμενο MineField, και υλοποιεί το παιχνίδι του παίχτη. Ρωτάει τον παίχτη (με το όνομα του) ποιο κελί θέλει να ανοίξει, και το ανοίγει. Αν υπάρχει νάρκη τυπώνεται ένα μήνυμα και ενημερώνεται το score. Αν ο παίχτης έκανε λάθος στις συντεταγμένες που έδωσε (εκτός ορίων ή πάνω σε ανοιγμένο κελί) δεν γίνεται τίποτα.
Υλοποιήστε επίσης accessor μεθόδους για τα πεδία της κλάσης, ή όποια μέθοδο χρειάζεστε για να τυπώσετε τις πληροφορίες που ζητάει η κλάση MineSweeper παρακάτω.

MineSweeper: Η κλάση που περιέχει την μέθοδο **main**, η οποία υλοποιεί τη βασική ροή του παιχνιδιού. Δημιουργεί τα αντικείμενα για το MineField και τους παίχτες. Οι παίχτες παίζουν εναλλάξ. Πριν παίξει ένας παίκτης τυπώνεται το πλέγμα και αφού παίξει τυπώνονται τα scores και των δύο παιχτών. Το παιχνίδι σταματάει όταν βρεθούν όλες οι νάρκες και ανακηρύσσεται ο τελικός νικητής (αν δεν είναι ισοπαλία).

(Μικρό) Bonus: Κρατώντας την μέθοδο check σταθερή τροποποιήστε την υλοποίηση σας ώστε ένας παίχτης να πρέπει να δώσει σωστές συντεταγμένες (εντός ορίων και σε κλειστό κελί) για να προχωρήσει το παιχνίδι.