**Δεύτερη Σειρά ασκήσεων**
**Ημερομηνία Παράδοσης: Δευτέρα 12/5 3:00 μ.μ.**

Οι παρακάτω ασκήσεις πρέπει να παραδοθούν μέχρι τις 12/5 πριν το μάθημα . 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

Στην πρώτη σειρά ασκήσεων υλοποιήσατε τον Αφηρημένο Τύπο Δεδομένων **ταξινομημένος πίνακας** χρησιμοποιώντας ένα πίνακα σταθερού μεγέθους. Σε αυτή την άσκηση θα υλοποιήσουμε μια **ταξινομημένη λίστα** χωρίς περιορισμό στο μέγεθος. Η ιδέα της υλοποίησης είναι παρόμοια με την υλοποίηση μιας στοίβας με συνδεμένα στοιχεία που είδαμε στην τάξη. Χρειαζόμαστε μόνο μία αναφορά που να δείχνει στην κορυφή της λίστας για να έχουμε πρόσβαση σε όλα τα στοιχεία της. Το πλεονέκτημα σε σχέση με τον πίνακα που υλοποιήσατε στην προηγούμενη άσκηση είναι ότι δεν έχουμε περιορισμό στο μέγεθος και μπορούμε πιο γρήγορα να προσθέσουμε και να αφαιρέσουμε στοιχεία.

Η λίστα μας θα κρατάει πληροφορίες για άτομα (**Person**). Το κάθε άτομο θα έχει ένα όνομα (**name**) και ένα αριθμό μητρώου (**AM**) και η ταξινόμηση θα γίνεται σε αύξουσα σειρά πρώτα ως προς όνομα και μετά ως προς τον αριθμό μητρώου. Η κλάση Person θα πρέπει να έχει μια μέθοδο:

**public int compareTo(Person other)**

η οποία επιστρέφει -1 αν το Person other είναι «μεγαλύτερο» από το Person this στην ταξινόμηση, 1 αν το Peson other είναι «μικρότερο» από το this στην ταξινόμηση, και 0 αν είναι ίδια. Η μέθοδος θα χρησιμοποιεί την compareTo που είναι ήδη ορισμένη για Strings. Θα χρησιμοποιήστε αυτή τη μέθοδο για την ταξινόμηση.

Κατασκευάσετε μια κλάση **SortedPersonList** που υλοποιεί μια ταξινομημένη λίστα από άτομα. Η κλάση θα πρέπει να έχει τις εξής μεθόδους:

- **public boolean containsElement(Person p):** παίρνει σαν όρισμα ένα αντικείμενο Person p και επιστρέφει true αν υπάρχει στλιστα και false αλλιώς.
- **public boolean insert(Person p):** παίρνει σαν όρισμα ένα αντικείμενο Person p και το προσθέτει στην κατάλληλη θέση στη λίστα εφόσον δεν υπάρχει ένα αντικείμενο με τις ίδιες τιμές ήδη στην λίστα. Επιστρέφει μία boolean τιμή αν έγινε η προσθήκη του στοιχείου.
- **public boolean delete(Person p):** παίρνει σαν όρισμα ένα αντικείμενο Person p αν υπάρχει στη λίστα το αφαιρεί. Επιστρέφει μία boolean τιμή αν έγινε η αφαίρεση του στοιχείου.
- **public Person get(int i)**: επιστρέφει το στοιχείο στη θέση i, εφόσον υπάρχουν αρκετά στοιχεία, αλλιώς επιστρέφει null.
- **public int size():** επιστρέφει τον αριθμό των στοιχείων στη λίστα.
- **public Person min()**: επιστρέφει το πρώτο στοιχείο στη λίστα.
- **public Person max()**επιστρέφει το τελευταίο στοιχείο στη λίστα.
- **public String toString():** επιστρέφει ένα String με τα στοιχεία της λίστας ταξινομημένα χωρισμένα με κενό. Η toString() θα υλοποιηθεί καλώντας φωλιασμένες toString(). Η μέθοδος toString() για την Person θα πρέπει να επιστρέφει το όνομα και το ΑΜ μέσα σε μία παρένθεση χωρισμένα με κόμμα.
- **public boolean equals(SortedPersonList other):** ελέγχει αν ο πίνακας έχει τα ίδια στοιχεία (όνομα και τηλέφωνο) με τον other. Χρησιμοποιήστε φωλιασμένες equals για να το υλοποιήσετε.
- **public SortedPersonList(SortedPersonList other):** copy constructor που δημιουργεί ένα βαθύ αντίγραφο του αντικειμένου other. Χρησιμοποιήστε φωλιασμένους copy constructors για να το υλοποιήσετε.

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

**Υποδείξεις**

- Για να πάρετε όλους τους βαθμούς θα πρέπει να φροντίσετε και τις οριακές καταστάσεις.
- **Bonus**: Για να πάρετε το μέγιστο στοιχείο στην λίστα θα πρέπει να διατρέξετε όλα τα στοιχεία της λίστας. Τροποποιήστε το πρόγραμμα ώστε να έχετε μια αναφορά που δείχνει στο τέλος της λίστας και μας δίνει κατευθείαν το τελευταίο στοιχείο. Σημειώστε σε σχόλιο στην αρχή της υλοποίησης σας αν έχετε υλοποιήσει το bonus.

## Άσκηση 2

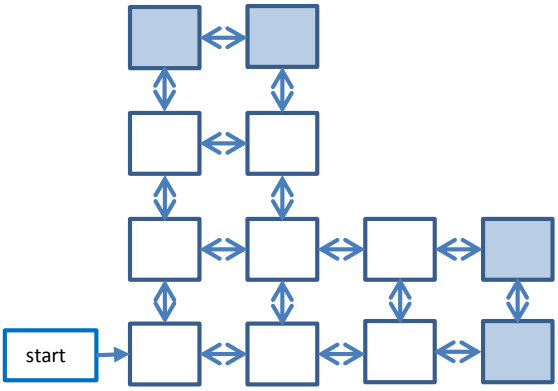
Για την άσκηση αυτή θα υλοποιήσετε σε Java το εξής παιχνίδι: έχουμε έναν παίχτη (Player) που ελέγχει ο χρήστης ο οποίος κινείται μέσα σε ένα λαβύρινθο (Maze) προσπαθώντας να βρει την άκρη του λαβύρινθου. Στον λαβύρινθο κινούνται επίσης και άλλοι παίχτες που ελέγχονται από τον υπολογιστή. Όταν ο παίχτης σας βρεθεί στο ίδιο σημείο με παίχτες του υπολογιστή γίνεται μια μάχη και σκοτώνεται ο πιο αδύναμος. Το παιχνίδι συνεχίζεται μέχρι ο παίχτης σας να φτάσει στην έξοδο, ή να σκοτωθεί.

Η υλοποίηση σας θα ακολουθήσει τα παρακάτω βήματα. Παραδώσετε το κάθε βήμα ξεχωριστά μιας και θα βαθμολογηθούν ανεξάρτητα.

**Βήμα 1: Δημιουργία του λαβύρινθου.**

Ο λαβύρινθος που θα φτιάξετε αποτελείται από κελιά (Cell) τα οποία συνδέονται μεταξύ τους. Το κάθε κελί έχει τέσσερα γειτονικά κελιά: πάνω (βοράς), κάτω (νότος), δεξιά (ανατολή) και αριστερά (δύση). Υλοποιήστε μια κλάση **Cell** η οποία κρατάει πληροφορία για το κελί. Εκτός από τους γείτονες θα πρέπει να κρατάει πληροφορία και για το αν το κελί είναι έξοδος ή όχι.

Ο λαβύρινθος που θα κατασκευάσετε θα έχει σχήμα L, όπου η κάθε πλευρά του L έχει μήκος length (παράμετρος που περνάμε στον constructor) και πάχος 2. Στο διπλανό σχήμα φαίνεται ένας λαβύρινθος με length = 4. Κάθε τετράγωνο είναι ένα κελί και τα βελάκια μεταξύ των κελιών δείχνουν ότι μπορούμε να μετακινηθούμε από το ένα κελί στο άλλο. Η έξοδος του λαβύρινθού θα είναι ένα από τα πιο βόρεια, ή πιο ανατολικά κελιά (τα χρωματισμένα κελιά στο σχήμα μας). Το κελί της εξόδου επιλέγεται τυχαία από ένα από αυτά.




Υλοποιήστε την κλάση **Maze** που δημιουργεί τον λαβύρινθο. Η κλάση κρατάει πληροφορία για το σημείο εκκίνησης (start) το οποίο είναι η κάτω αριστερά γωνία του L όπως φαίνεται και στο σχήμα. Από αυτό το σημείο μπορούμε να διατρέξουμε όλο το λαβύρινθο. Ο constructor της Maze θα παίρνει σαν όρισμα το length και θα καλεί μια βοηθητική μέθοδο constructMaze η οποία θα τον κατασκευάζει. Η constructMaze, θα κατασκευάζει τον λαβύρινθο κελί-κελί. Ξεκινάει δημιουργώντας το κελί start. Μετά δημιουργεί τα κελιά της εξωτερικής νότιας πλευράς του L, μετά τα κελιά της εξωτερικής δυτικής πλευράς του L, μετά την εσωτερική βόρια πλευρά και τέλος την εσωτερική ανατολική πλευρά. Για κάθε κελί που δημιουργούμε θα πρέπει να φροντίζουμε να το συνδέουμε κατάλληλα με τα προηγούμενα κελιά ώστε να είναι δυνατή η μετακίνηση μεταξύ των κελιών. Μια περιγραφή του πως μπορεί να γίνει η κατασκευή του λαβύρινθου δίνεται στις διαφάνειες που είναι στη σελίδα του μαθήματος. Ένα από τα ακριανά κελιά επιλέγεται τυχαία για είναι η έξοδος.

Τέλος η Maze θα κρατάει και ένα ArrayList στο οποίο αποθηκεύει όλα τα κελιά του λαβύρινθου που δημιουργήσαμε. Η λίστα αυτή χρειάζεται για να μπορούμε να διαλέξουμε τυχαία ένα από τα κελιά του λαβύρινθου όταν τοποθετούμε ένα παίχτη.

**Bonus**: Δημιουργήστε ένα επιπλέον (σύνθετο) σχήμα για τον λαβύρινθο (π.χ. σταυρός).

**Βήμα 2: Υλοποίηση των παιχτών**

Υλοποιήστε μια κλάση **Player** η οποία κρατάει πληροφορία για ένα παίχτη. Ο παίχτης έχει μια δύναμη (strength), η οποία μπορεί να δίνεται σαν όρισμα στον constructor. Ο default constructor δίνει στο strength μια τυχαία τιμή μεταξύ 0 και 99. Ο παίχτης κρατάει επίσης και πληροφορία για το κελί στο οποίο βρίσκεται. Έχουμε και μία μέθοδο place(Cell c) η οποία τοποθετεί τον παίχτη στο κελί c. Αντίστοιχα, το κάθε κελί πλέον χρειάζεται να κρατάει επιπλέον ένα ArrayList με όλους τους παίχτες που βρίσκονται σε αυτό το κελί και να έχει μία μέθοδο που προσθέτει (και αφαιρεί) ένα παίχτη στην (από την) λίστα.

Θα υλοποιήσετε δύο μεθόδους στην Player για την κίνηση των παιχτών. Η μέθοδος **computerMove()** μετακινεί ένα παίχτη που ελέγχεται από τον υπολογιστή, σε ένα τυχαίο γειτονικό κελί. Για την υλοποίηση θα σας είναι χρήσιμο να φτιάξετε μια μέθοδο στην κλάση Cell που θα επιστρέφει ένα τυχαίο γειτονικό κελί. Η μέθοδος **humanMove()** μετακινεί τον παίχτη που ελέγχεται από τον χρήστη. Πρώτα εκτυπώνει την κατάσταση των γειτονικών κελιών: "X" αν δεν υπάρχει το κελί, αλλιώς τον αριθμό των παιχτών που υπάρχουν σε αυτό το κελί. Ζητάμε από τον χρήστη να διαλέξει αν θέλει να πάει πάνω, κάτω, δεξιά ή αριστερά. Αν η κίνηση είναι δυνατή μετακινούμε τον παίχτη αλλιώς μένει εκεί που είναι. Για την υλοποίηση θα σας είναι χρήσιμο να φτιάξετε μια μέθοδο στην κλάση Cell που θα τυπώνει τον αριθμό των παιχτών στους γείτονες του κελιού. Και για τις δύο μεθόδους όταν γίνονται οι μετακινήσεις θα πρέπει να είσαστε προσεκτικοί να ενημερώνετε και την κατάσταση των κελιών από και προς τα οποία μετακινείται ένας παίκτης.

**Βήμα 3: Υλοποίηση του υπόλοιπου παιχνιδιού**

Στο βήμα αυτό θα υλοποιήσετε το υπόλοιπο παιχνίδι. Χρειάζεστε (μεταξύ άλλων) να υλοποιήσετε τα παρακάτω:

- Μια μέθοδο στην κλάση Cell που υλοποιεί την μάχη, όπου από τους παίχτες που βρίσκονται σε αυτό το κελί θα βρίσκετε τον πιο αδύναμο και θα τον αφαιρείτε. Θα πρέπει να ξέρετε αν στο τέλος της μάχης αυτός που σκοτώθηκε είναι ο παίχτης του χρήστη.
- Μια μέθοδο που θα δημιουργεί τους παίχτες του υπολογιστή, και θα τους τοποθετεί σε τυχαία κελιά μέσα στον λαβύρινθο. Είναι αποδεκτό παραπάνω από ένας παίχτης να τοποθετηθούν στο ίδιο κελί. Η μέθοδος θα επιστρέφει ένα ArrayList με τους παίχτες του υπολογιστή.
- Έναν τρόπο να ελέγχετε αν ο παίχτης έφτασε στην έξοδο του λαβύρινθου.

Στη συνέχεια θα υλοποιήσετε την κλάση MazeGame η οποία θα έχει την main που υλοποιεί τον κορμό του παιχνιδιού:

(1) Δημιουργεί τον λαβύρινθο με μέγεθος που δίνεται σαν command line παράμετρος.
(2) Δημιουργεί τον παίχτη του χρήστη με δύναμη 50 και τον τοποθετεί στην αρχή του λαβύρινθου.
(3) Δημιουργεί 4*length παίχτες του υπολογιστή και τους τοποθετεί σε τυχαία κελιά στον λαβύρινθο.
(4) Όσο ο παίχτης του χρήστη δεν έχει φτάσει στην έξοδο ή δεν έχει σκοτωθεί:
    a. κινούνται οι παίχτες του υπολογιστή,
    b. κινείται ο παίχτης του χρήστη
    c. γίνεται η μάχη στο κελί του χρήστη

Ο στόχος της άσκησης είναι να υλοποιήσετε τις τέσσερεις κλάσεις ώστε να τρέχει το παιχνίδι. Θα βαθμολογήσουμε όμως και τα επιμέρους βήματα, οπότε θα πρέπει να παραδώσετε ξεχωριστά αρχεία για κάθε βήμα τα οποία θα υλοποιούν τον κώδικα του συγκεκριμένου βήματος. Οπότε, για το Βήμα 1 θα παραδώσετε τις κλάσεις Cell1, Maze1. Για το Βήμα 2 τις κλάσεις Cell2, Playerr2 και τέλος για το Βήμα 3, τις κλάσεις Cell, Maze, Player, MazeGame ώστε να τρέχει το παιχνίδι.

**Υπόδειξη**: Δεν είναι απαραίτητο, αλλά σε περίπτωση που χρειάζεστε να δημιουργήσετε μια βοηθητική μέθοδο που θα καλεί η main, η μέθοδος θα πρέπει να οριστεί ως static.