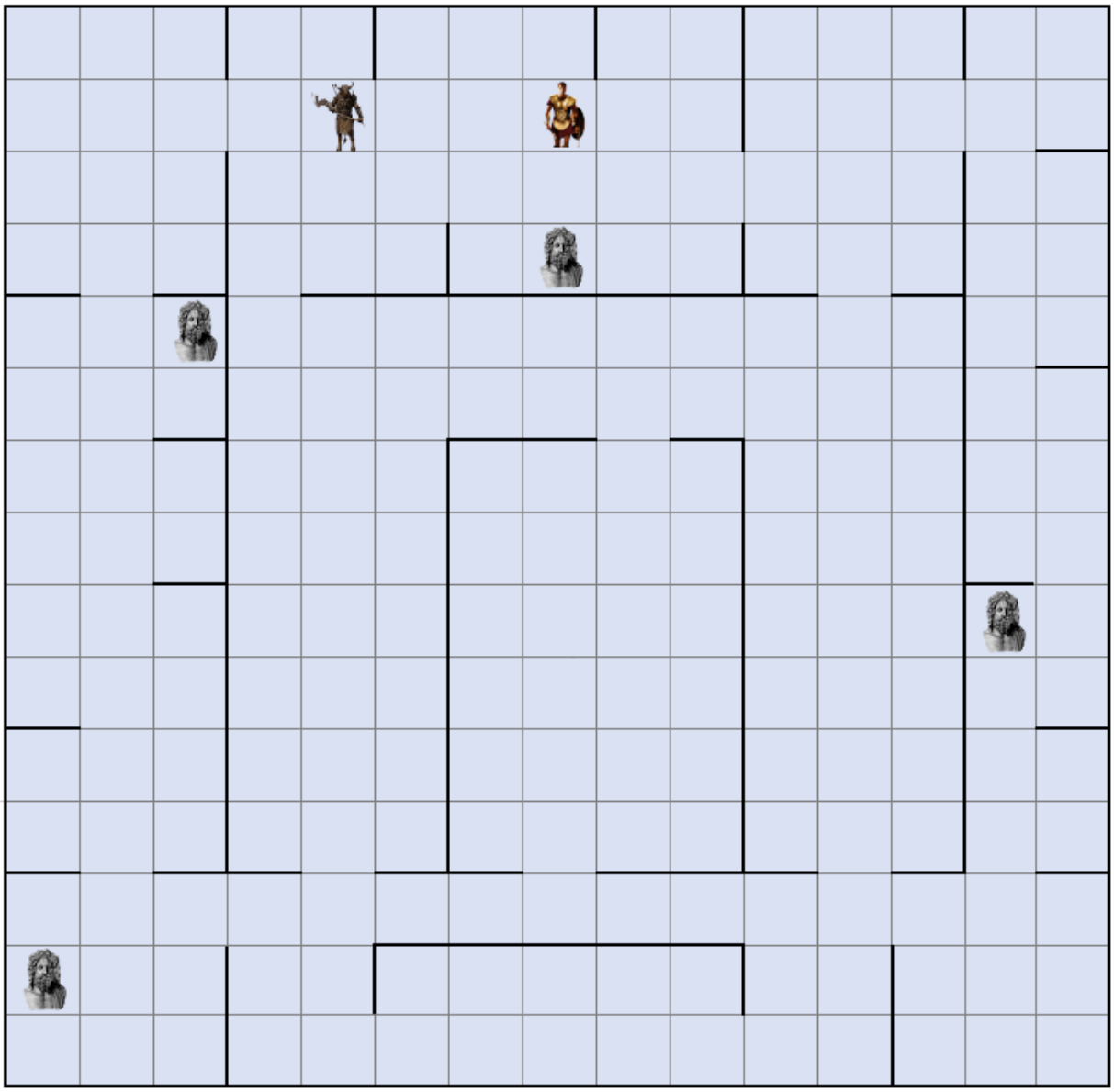
Εργαστήριο Επεξεργασίας Πληροφορίας και Υπολογισμών
Τομέας Ηλεκτρονικής και Υπολογιστών
ΤΗΜΜΥ Α.Π.Θ

ΔΟΜΕΣ ΔΕΔΟΜΕΝΩΝ

Λαβύρινθος

Ο Θησέας και ο Μινώταυρος

Στην ταινία «Μια νύχτα στο μουσείο» η οποία διαδραματίζεται στο Μουσείο Φυσικής Ιστορίας στην Αμερική, τα εκθέματα αποκτούν ζωή για ένα βράδυ και συμμετέχουν σε διάφορες περιπέτειες.



Εικόνα 1: Παράδειγμα ταμπλό παιχνιδιού διάστασης 15x15.

Μια από τις περιπέτειες στις οποίες συμμετέχουν τα εκθέματα είναι αυτή του Θησέα και του Μινώταυρου.

Ο ατρόμητος Θησέας αποφασίζει να σκοτώσει το Μινώταυρο. Όταν αποβιβάζεται στο νησί, ο Θησέας συναντάει την Αριάδνη, την κόρη του Μίνωα. Η Αριάδνη για να τον βοηθήσει να δραπετεύσει του *λέει «πρέπει να ανακαλύψεις τα διάσπαρτα λάφυρα μέσα στον λαβύρινθο. Μόλις τα μαζέψεις όλα μία καταπακτή θα ανοίξει και θα πέσει μέσα ο Μινώταυρος».*

Θα καταφέρει ο Θησέας να ανακαλύψει όλα τα λάφυρα-εκθέματα πριν πέσει στα χέρια του Μινώταυρου;

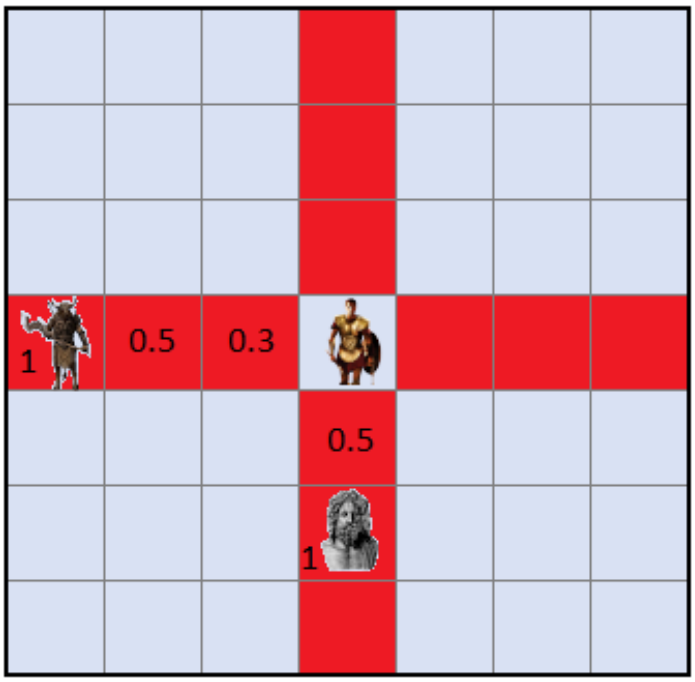
Στόχος του Θησέα είναι να βρει τα λάφυρα στα διάφορα σημεία του μουσείου χωρίς να τον εξοντώσει ο Μινώταυρος. Ο Θησέας χάνει την ζωή του αν βρεθεί στο ίδιο πλακίδιο με τον Μινώταυρο.

Ο Θησέας έχει ένα φωτόσπαθο το οποίο φωτίζει το δρόμο του συνεχώς με ένα άσπρο φως. Όταν βρεθεί σε απόσταση 2 κενών πλακιδίων (3 στο σύνολο) από ένα λάφυρο το φωτόσπαθο γίνεται πράσινο. Αντίστοιχα όταν βρεθεί ο Θησέας σε απόσταση 2 πλακιδίων από τον Μινώταυρο το φωτόσπαθο γίνεται κόκκινο. Μόλις βρει ο Θησέας όλα τα λάφυρα το παιχνίδι ολοκληρώνεται με νικητή τον Θησέα.

Από την άλλη πλευρά, ο Μινώταυρος προσπαθεί να εντοπίσει το Θησέα και να τον σκοτώσει. Ο Μινώταυρος όμως είναι αρκετά έξυπνος και μπορεί διαισθανθεί τα εφόδια όταν βρίσκεται σε απόσταση 2 κενών πλακιδίων (3 στο σύνολο) από αυτά. Θέλει να παραμένει γύρω από τα σημεία αυτά μιας και ξέρει ότι έτσι θα μπορεί να πέσει πιο εύκολα πάνω στο Θησέα. Φυσικά, χρησιμοποιώντας την ιδιότητα του ως ζώο μπορεί να μυρίσει το θήραμά του όταν βρίσκεται σε απόσταση 2 κενών πλακιδίων (3 στο σύνολο) έτσι ώστε να μπορεί να τον πλησιάσει και όταν βρεθεί στο ίδιο τετραγωνάκι να τον σκοτώσει.

Το παιχνίδι τερματίζει και στην περίπτωση που εκτελεστούν $2n$ βήματα και δεν έχε βρεθεί νικητής, το παιχνίδι θεωρείται ισόπαλο, καθώς έχει ξημερώσει και τα εκθέματα θα πρέπει να πέσουν πάλι σε λήθαργο.

ΠΡΟΣΟΧΗ!!! **Όταν αναφέρονται αποστάσεις μεταξύ παικτών και εφοδίων, θεωρούμε αποστάσεις μόνο ως προς τις 4 κατευθύνσεις που μπορούν να κινηθούν οι παίκτες (όχι διαγώνια) και επίσης εάν είναι σε απόσταση 2 πλακιδίων ή κοντινότερη και παρεμβάλετε τείχος μεταξύ του παίκτη και του εφοδίου ή του άλλου παίκτη, δεν ενεργοποιούνται το φωτόσπαθο και οι αισθήσεις του Μινώταυρού αντίστοιχα. Για τις αποστάσεις κα την οπτική του παίκτη βλέπε την Εικόνα 2.**



Εικόνα 2 Η "οπτική" του Θησέα και οι πόντοι που θα κερδίζει στην αξιολόγηση

Εργασία Β – Heuristic Algorithm (0,75 βαθμοί)

Στο δεύτερο παραδοτέο καλείστε να υλοποιήσετε έναν παίκτη HeuristicPlayer ο οποίος έχει τη δυνατότητα να **ορίζει ο ίδιος το ζάρι** που χρειάζεται σε κάθε γύρο για να κάνει τη βέλτιστη δυνατή κίνηση. Ζητείται λοιπόν να δημιουργήσετε μια **συνάρτηση αξιολόγησης των διαθέσιμων κινήσεων** που έχει ο παίκτης σε κάθε γύρο παιχνιδιού και την αντίστοιχη **συνάρτηση επιλογής της καλύτερης κίνησης**. Σκοπός σας είναι να δημιουργήσετε μια όσο το δυνατόν πιο ολοκληρωμένη συνάρτηση αξιολόγησης. Αυτή θα λαμβάνει υπόψη τα διαθέσιμα δεδομένα (μέχρι και) εκείνη τη στιγμή και θα τα αξιολογεί κατάλληλα, εφαρμόζοντας διάφορα κριτήρια. Στόχος του παίκτη είναι να πλησιάσει (μαζέψει) τα εφόδια που χρειάζεται (NearSupplies) και να κινείται προς τον αντίπαλο (μακριά από τον αντίπαλο) με σκοπό να τον σκοτώσει (να μην το σκοτώσουν)(OpponentDist). Η συνάρτηση στόχου που θα βελτιστοποιήσετε θα οριστεί από εσάς. Ένα παράδειγμα πιθανής συνάρτησης στόχου είναι η παρακάτω:

f(NearSupplies, NearOpponent) = NearSupplies * 0,46 + OpponentDist * 0,54

Ένα παράδειγμα εξέτασης της συγκεκριμένης συνάρτησης στόχου για το ταμπλό της Εικόνας 1 για τον Θησέα στη θέση (12, 7) ή tile: 187 περιγράφεται παρακάτω:

Ο παίκτης έχει στη διάθεσή του 4 δυνατές κινήσεις.

- *Στην περίπτωση που επιλέξει το ζάρι του να έχει τον αριθμό **1**, θα κάνει **1** βήμα και θα κερδίσει **0** πόντους* -> f(NearSupplies, OpponentDist) = 0.
- *Στην περίπτωση που επιλέξει το ζάρι του να έχει τον αριθμό **3**, θα κάνει **1** βήμα και θα κερδίσει **0** πόντους* -> f(NearSupplies, OpponentDist) = 0.
- *Στην περίπτωση που επιλέξει το ζάρι του να έχει τον αριθμό **5**, θα κάνει **1** βήμα και θα κερδίσει **0.5** πόντο -> f(NearSupplies,* OpponentDist*) = 0,23.*

- *Στην περίπτωση που επιλέξει το ζάρι του να έχει τον αριθμό **7**, θα κάνει **1** βήμα και θα χάσει **0,3** πόντους* -> f(NearSupplies, OpponentDist) = -0.162.

Ο κώδικας του δεύτερου παραδοτέου θα συμπληρωθεί από εσάς στον κώδικα που ετοιμάσατε για το πρώτο παραδοτέο. Όσοι επιθυμείτε, μπορείτε να χρησιμοποιήσετε και την ενδεικτική λύση της πρώτης εργασίας που θα ανέβει στην ενότητα «Εργασίες και Τεστ» στην πλατφόρμα e-learning. Στη συνέχεια παρουσιάζεται η νέα κλάση **HeuristicPlayer** που σας ζητείται να υλοποιήσετε, καθώς και η κλάση **Game** όπου θα προσθέσετε επιπλέον λειτουργικότητα.

ΠΡΟΣΟΧΗ!!! Θέλουμε την υλοποίηση του HeuristicPlayer **ΜΟΝΟ** για το **Θησέα.** Φυσικά, όποιος επιθυμεί μπορεί να υλοποιήσει και τον HeuristicPlayer για τον Μινώταυρο (**ΔΕΝ ΑΠΑΙΤΕΙΤΑΙ)**.

Κλάση HeuristicPlayer

Η κλάση **HeuristicPlayer** θα αντιπροσωπεύει τον παίκτη που παίζει με στρατηγική. Κληρονομεί την κλάση **Player** και έχει τις εξής επιπλέον μεταβλητές:

1. **`ArrayList<Integer[]> path`**: πληροφορίες για την κάθε κίνηση του παίκτη κατά τη διάρκεια του παιχνιδιού. Πιο συγκεκριμένα, η δομή θα περιλαμβάνει πληροφορίες για το ζάρι, τον αριθμό των πόντων που προσέφερε η κίνηση του και αν βρίσκεται κοντά σε ένα αντικείμενο καθώς και αν βρίσκεται κοντά στον αντίπαλό του. Μπορείτε να αποθηκεύσετε και ό,τι άλλο μπορεί να σας φανεί χρήσιμο.

Το ζάρι που επιλέχθηκε	**Εάν πήρε ένα εφόδιο**	**Απόσταση από εφόδιο**	**Απόσταση από εφόδιο**
Πιθανές τιμές: **{1,3,5,7}**	Πιθανές τιμές: **{0, 1}**	Πιθανές τιμές: **{1,2,3}**	Πιθανές τιμές: {**1, 2, 3}**

Οι συναρτήσεις που πρέπει να υλοποιήσετε είναι οι εξής:

a. Οι **constructors** της κλάσης.
b. Συνάρτηση **`double evaluate(int currentPos, int dice)`:** Η συνάρτηση αυτή αξιολογεί την κίνηση του παίκτη όταν αυτός έχει τη ζαριά dice, δεδομένου ότι βρίσκεται στη θέση currentPos. Η συνάρτηση επιστρέφει την αξιολόγηση της κίνησης, σύμφωνα με τη συνάρτηση στόχου που έχετε ορίσει.
c. Συνάρτηση **`int getNextMove(int currentPos)`:** Η συνάρτηση αυτή είναι υπεύθυνη για την επιλογή της τελικής κίνησης του παίκτη. Η συνάρτηση θα πρέπει να περιλαμβάνει τα παρακάτω:
    i. Δημιουργία μιας δομής (Δυναμικός ή μη πίνακας) που θα αποθηκεύει τις πιθανές κινήσεις του παίκτη καθώς και την αξιολόγηση κάθε μιας από αυτές.
    ii. Αξιολόγηση κάθε πιθανής κίνησης με χρήση της συνάρτησης `double evaluate(int currentPos, int dice)` και αποθήκευση της αξιολόγησης στη δομή που έχετε φτιάξει.

iii. Επιλογή της καλύτερης κίνησης με βάση την αξιολόγηση που κάνατε.
iv. Ανανέωση της μεταβλητής της κλάσης path**.**
v. Επιστροφή της νέας θέσης του παίκτη.

ΠΡΟΣΟΧΗ!!! **Κατά τη διάρκεια των δοκιμών που θα κάνετε για την επιλογή της βέλτιστης κίνησης, θα πρέπει να προσέξετε ώστε να αφήσετε αναλλοίωτο το ταμπλό και το σκορ των παικτών. Αυτά θα πρέπει να αλλάξουν μόνο μετά την επιλογή της βέλτιστης κίνησης, δηλαδή κατά τη διάρκεια της πραγματικής μετακίνησης του παίκτη.**

d. Συνάρτηση **`void statistics():`** Η συνάρτηση αυτή εκτυπώνει στοιχεία για τις κινήσεις του παίκτη σε κάθε γύρο του παιχνιδιού (ζάρι που επέλεξε και ενέργειες που έκανε πχ «ο παίκτης στο round 1 έθεσε το ζάρι ίσο με 4 και πλησίασε ένα λάφυρο»), καθώς και στατιστικά στοιχεία για το σύνολο των κινήσεών του. Συγκεκριμένα, στο τέλος ζητείται να εκτυπώνονται:

Για κάθε γύρο:

- Ο αριθμός των εφοδίων που έχει μαζέψει (Θησέας).
- Η απόστασή του (σε πλακίδια) από το κοντινότερο εφόδιο. Εφόσον μπορεί να το δει, δηλαδή δεν υπάρχουν τείχη μεταξύ
- Η απόστασή του (σε πλακίδια) από το κοντινότερο εφόδιο. Εφόσον μπορεί να το δει, δηλαδή δεν υπάρχουν τείχη μεταξύ

Μια φορά συνολικά:

- Οι φορές που ο παίκτης επέλεξε να κινηθεί μπροστά (1).
- Οι φορές που ο παίκτης επέλεξε να κινηθεί δεξιά (3).
- Οι φορές που ο παίκτης επέλεξε να κινηθεί πίσω (5).
- Οι φορές που ο παίκτης επέλεξε να κινηθεί αριστερά (7).

Κλάση Game

Η κλάση **Game** θα αντιπροσωπεύει το παιχνίδι και θα έχει τις εξής μεταβλητές:

i. **`int round`**: ο τρέχον γύρος του παιχνιδιού.

Οι συναρτήσεις που πρέπει να υλοποιήσετε είναι οι εξής:

a. Οι **constructors** της κλάσης.
b. Όλες οι συναρτήσεις ***get*** και ***set*** για τη μεταβλητή της κλάσης.
c. Συνάρτηση **`public static void main(String[] args):`** συνάρτηση εκκίνησης του παιχνιδιού. Στη συνάρτηση αυτή θα πρέπει να γίνεται μια ακολουθία ενεργειών:
    - Δημιουργία ταμπλό διάστασης 15x15.
    - Εισαγωγή 4 εφοδίων στο ταμπλό.

- Ορισμός 2 παικτών, ένας παίκτης που παίζει με τυχαίες κινήσεις (Μινώταυρος) και ένας παίκτης τύπου HeuristicPlayer (Θησέας). (Όποιος υλοποίησε και το Μινώταυρο ως HeuristicPlayer, θα έχει 2 παίκτες τύπου HeuristicPlayer)
- Οι παίκτες παίζουν εναλλάξ (ρίχνουν το ζάρι και κινούνται στο ταμπλό) μέχρις ότου ένας από τους 2 κερδίσει ή να παρέλθουν οι 2n (όπου n = 100) ζαριές.
- Τα τείχη που μπορούν να δημιουργηθούν έχουν ως ανώτατο όριο την τιμή: $(N * N * 3 + 1)/2$
- Σε κάθε γύρο θα εκτυπώνονται τα εξής: Το round, το ταμπλό του παιχνιδιού και η κίνηση του κάθε παίκτη. Για την εκτύπωση του ταμπλό θα χρησιμοποιηθεί η συνάρτηση `getStringRepresentation()` της κλάσης `Board`.
- Εκτύπωση στατιστικών στοιχείων για τον παίκτη τύπου HeuristicPlayer.
- Στο τέλος του παιχνιδιού θα εκτυπώνεται ο νικητής του παιχνιδιού.

ΠΡΟΣΟΧΗ!!!

- **Αν θέλετε να διαβάσετε τις τιμές των μεταβλητών ενός αντικειμένου μιας κλάσης ή να θέσετε τιμές στις μεταβλητές θα πρέπει να χρησιμοποιήσετε τους αντίστοιχους getters** και **setters.**
- **Μπορείτε να προσθέσετε επιπλέον μεταβλητές και συναρτήσεις στις κλάσεις εφόσον αυτό σας εξυπηρετεί, αρκεί να φροντίσετε για την αναλυτική περιγραφή τους στην αναφορά.**

Οδηγίες

Τα προγράμματα θα πρέπει να υλοποιηθούν σε Java, με πλήρη τεκμηρίωση του κώδικα. Το πρόγραμμά σας πρέπει να περιέχει επικεφαλίδα σε μορφή σχολίων με τα στοιχεία σας (ονοματεπώνυμα, ΑΕΜ, τηλέφωνα και ηλεκτρονικές διευθύνσεις). Επίσης, πριν από κάθε κλάση ή μέθοδο θα υπάρχει επικεφαλίδα σε μορφή σχολίων με σύντομη περιγραφή της λειτουργικότητας του κώδικα. Στην περίπτωση των μεθόδων, πρέπει να περιγράφονται και οι μεταβλητές τους.

Οι εργασίες που περιέχουν λάθη μεταγλώττισης θα μηδενίζονται αυτομάτως.

Είναι δική σας ευθύνη η απόδειξη καλής λειτουργίας του προγράμματος.

Παραδοτέα:

1. **Ηλεκτρονική αναφορά** που θα περιέχει: εξώφυλλο με το **ΑΕΜ, την ομάδα και το ονοματεπώνυμο σας**, περιγραφή του προβλήματος, του αλγορίθμου και των διαδικασιών που υλοποιήσατε και τυχόν ανάλυσή τους. Σε καμία περίπτωση να μην αντιγράφεται ολόκληρος ο κώδικας μέσα στην αναφορά (εννοείται ότι εξαιρούνται τμήματα κώδικα τα οποία έχουν ως στόχο τη διευκρίνιση του αλγορίθμου).
   Προσοχή: Ορθογραφικά και συντακτικά λάθη πληρώνονται.

2. **Ένα αρχείο σε μορφή .zip με όνομα "AEM1_AEM2_PartB.zip",** το οποίο θα περιέχει **όλο** το project σας στον eclipse καθώς και το αρχείο της γραπτής αναφοράς σε pdf (**αυστηρά**). Το αρχείο .zip θα γίνεται upload στο site του μαθήματος ***στην ενότητα των ομαδικών εργασιών και μόνο***. Τα ονόματα των αρχείων πρέπει να είναι με **λατινικούς χαρακτήρες**.

Προθεσμία υποβολής:

Κώδικας και αναφορά **Τρίτη 8 Δεκεμβρίου, 23:59** (ηλεκτρονικά)
Δε θα υπάρξει καμία παρέκκλιση από την παραπάνω προθεσμία.