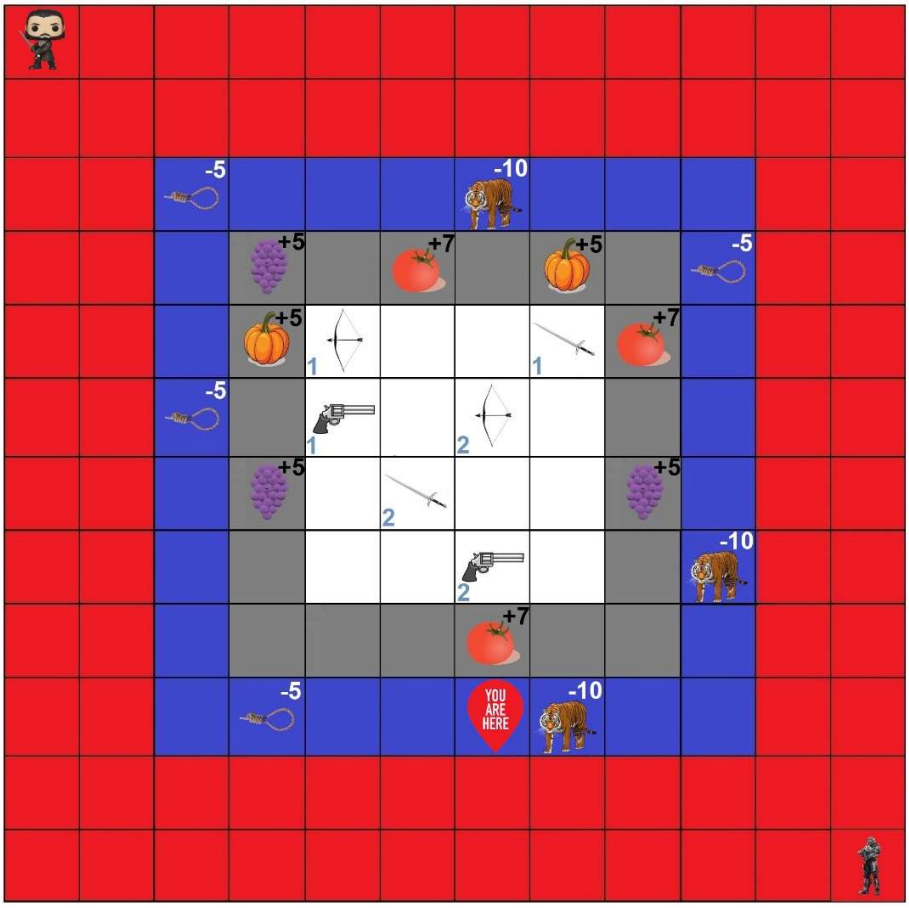
Εργαστήριο Επεξεργασίας Πληροφορίας και Υπολογισμών
Τομέας Ηλεκτρονικής και Υπολογιστών
ΤΗΜΜΥ Α.Π.Θ

ΔΟΜΕΣ ΔΕΔΟΜΕΝΩΝ

Hunger Games

May the odds be in your favor..

Το Hunger Games είναι ένα τηλεοπτικό ετήσιο γεγονός στο οποίο οι συμμετέχοντες είναι αναγκασμένοι να αγωνιστούν μέχρι θανάτου, έως ότου να υπάρξει κάποιος νικητής. Στην απλουστευμένη παραλλαγή του παιχνιδιού που καλείστε να υλοποιήσετε, συμμετέχουν 2 παίκτες, οι οποίοι παίζουν εναλλάξ και μετακινούνται κατά μία θέση πάνω στο ταμπλό κάθε φορά, έχοντας σαν στόχο να επιβιώσουν και να μαζέψουν πόντους. Ο ένας παίκτης μπορεί να σκοτώσει τον άλλον αν αυτός βρίσκεται σε κοντινή απόσταση και διαθέτει το κατάλληλο όπλο (πιστόλι). Νικητής του παιχνιδιού είναι εκείνος που θα μείνει ζωντανός ή, σε περίπτωση που το παιχνίδι τερματίσει επειδή το ταμπλό συρρικνώθηκε, εκείνος που έχει το υψηλότερο σκορ.



Εικόνα 1: Παράδειγμα ταμπλό παιχνιδιού διάστασης 12x12.

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 μαζέψει τα όπλα που χρειάζεται για να πολεμήσει (gainWeapons), να πέφτει σε λιγότερες παγίδες που δεν μπορεί να αντιμετωπίσει (avoidTraps), να κερδίζει τους περισσότερους δυνατούς πόντους (gainPoints) και να κινείται προς τον αντίπαλο με σκοπό να τον σκοτώσει (forceKill). Για να δώσετε τιμή στη μεταβλητή forceKill θα μπορούσατε να ελέγχεται την απόσταση του παίκτη σας από τον αντίπαλο (Βλέπε κλάση `HeuristicPlayer`, συνάρτηση `playersDistance()`). Όταν ο παίκτης μαζέψει τα όπλα που χρειάζεται (εσείς ορίζεται τη στρατηγική, δηλαδή ποια όπλα χρειάζεται), τότε θα μπορούσατε να μηδενίζεται το βάρος που αντιστοιχεί στη μεταβλητή gainWeapons στη συνάρτηση στόχου και να αυξάνετε το βάρος που αντιστοιχεί στη μεταβλητή forceKill. Η συνάρτηση στόχου που θα βελτιστοποιήσετε θα οριστεί από εσάς. Ένα παράδειγμα πιθανής συνάρτησης στόχου είναι η παρακάτω:

$$\text{f(gainWeapons, avoidTraps, gainPoints, forceKill)} = \text{gainWeapons} * 0{,}40 + \text{avoidTraps} * 0{,}30 + \text{gainPoints} * 0{,}30 + \text{forceKill} * 0$$

Ένα παράδειγμα εξέτασης της συγκεκριμένης συνάρτησης στόχου για το ταμπλό της Εικόνας 1 για κάποιον παίκτης που βρίσκεται στη θέση (1,4) περιγράφεται παρακάτω:

Ο παίκτης έχει στη διάθεσή του 6 δυνατές κινήσεις.

- *Στην περίπτωση που επιλέξει το ζάρι του να έχει τον αριθμό **1**, θα κάνει 1 βήμα προς τα πάνω και θα κερδίσει **7** πόντους ->* $\text{f}() = \mathbf{7*0{,}3 = 2{,}1}$.
- *Στην περίπτωση που επιλέξει το ζάρι του να έχει τον αριθμό **2**, θα κάνει 1 βήμα διαγώνια πάνω-δεξιά και θα κερδίσει **0** πόντους ->* $\text{f}() = \mathbf{0}$.
- *Στην περίπτωση που επιλέξει το ζάρι του να έχει τον αριθμό **3**, θα κάνει 1 βήμα δεξιά και θα χάσει **10** πόντους ->* $\text{f}() = \mathbf{(-10)*0{,}3 = -3}$.
- *Στην περίπτωση που επιλέξει το ζάρι του να έχει τον αριθμό **4**, θα κάνει 1 βήμα διαγώνια κάτω-δεξιά και θα κερδίσει **0** πόντους ->* $\text{f}() = \mathbf{0}$.
- *Στην περίπτωση που επιλέξει το ζάρι του να έχει τον αριθμό **5**, θα κάνει 1 βήμα κάτω και θα κερδίσει **0** πόντους ->* $\text{f}() = \mathbf{0}$.
- *Στην περίπτωση που επιλέξει το ζάρι του να έχει τον αριθμό **6**, θα κάνει 1 βήμα διαγώνια κάτω-αριστερά και θα κερδίσει **0** πόντους ->* $\text{f}() = \mathbf{0}$.
- *Στην περίπτωση που επιλέξει το ζάρι του να έχει τον αριθμό **7**, θα κάνει 1 βήμα αριστερά και θα κερδίσει **0** πόντους ->* $\text{f}() = \mathbf{0}$.
- *Στην περίπτωση που επιλέξει το ζάρι του να έχει τον αριθμό **8**, θα κάνει 1 βήμα διαγώνια πάνω αριστερά και θα κερδίσει **0** πόντους ->* $\text{f}() = \mathbf{0}$.

Συνεπώς, στο παραπάνω παράδειγμα, η βέλτιστη κίνηση που θα επέλεγε να κάνει ο παίκτης σύμφωνα με τη συνάρτηση στόχου είναι η 1.

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.

Κλάση HeuristicPlayer

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

i. **`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, δηλαδή τον κωδικό της κίνησης που επέλεξε, τον αριθμό των πόντων που προσέφερε η κίνηση, πληροφορίες για το αν έφαγε κάποιο εφόδιο, αν έπεσε σε κάποια παγίδα και αν μάζεψε κάποιο όπλο ο παίκτης στη συγκεκριμένη κίνησή του. Μπορείτε να αποθηκεύσετε και ό,τι άλλο μπορεί να σας φανεί χρήσιμο.

ii. **`static int r:` Κάθε παίκτης μπορεί να δει μόνο ένα τμήμα της πίστας που αντιστοιχεί σε ακτίνα r γύρω του.**

Οι συναρτήσεις που πρέπει να υλοποιήσετε είναι οι εξής:

a. Οι **constructors της κλάσης**.
b. Συνάρτηση **`float playersDistance(Player p)`**: Η συνάρτηση αυτή επιστρέφει την απόσταση του παίκτη από τον αντίπαλο `p` στο καρτεσιανό σύστημα συντεταγμένων που έχετε ορίσει εφόσον είναι στο οπτικό του πεδίο, σε κάθε άλλη περίπτωση να επιστρέφει -1.
c. Συνάρτηση **`double evaluate(int dice, Player p)`**: Η συνάρτηση αυτή αξιολογεί την κίνηση του παίκτη όταν αυτός έχει τη ζαριά `dice`, δεδομένου ότι βρίσκεται στη θέση `x,y` (μεταβλητές της κλάσης `Player` που έχουν κληρονομηθεί). Για την αξιολόγηση, θα χρειαστεί να χρησιμοποιήσετε και τη συνάρτηση `playersDistance(Player p)`, η οποία θα σας φανεί χρήσιμη αν ο παίκτης έχει μαζέψει το όπλο τύπου πιστόλι και μπορεί πλέον να σκοτώσει τον αντίπαλο (χρήση της μεταβλητής forceKill στο παράδειγμα της συνάρτησης στόχου σελ. 2). Η συνάρτηση επιστρέφει την αξιολόγηση της κίνησης, σύμφωνα με τη συνάρτηση στόχου που έχετε ορίσει.
d. Συνάρτηση **`static boolean kill(Player player1, Player player2, float d)`**: Η συνάρτηση αυτή ελέγχει αν η απόσταση μεταξύ των παικτών είναι μικρότερη του `d`. Σε περίπτωση που αυτό ισχύει, ο παίκτης που παίζει τη συγκεκριμένη στιγμή σκοτώνει τον αντίπαλο και επιστρέφει `true`.

e. Συνάρτηση **`int[2] move(Player p):`** 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 (`Map<Integer,Double>`) που θα αποθηκεύει τις **διαθέσιμες** πιθανές κινήσεις του παίκτη καθώς και την αξιολόγηση κάθε μιας από αυτές. **Προσοχή! Κάθε παίκτης μπορεί να δει μόνο ένα τμήμα της πίστας που αντιστοιχεί σε ακτίνα r γύρω από τον παίκτη.**

   ii. Αξιολόγηση κάθε πιθανής κίνησης με χρήση της συνάρτησης `double evaluate(int dice, Player p)` και αποθήκευση της αξιολόγησης στη δομή που έχετε φτιάξει.

   iii. Επιλογή της καλύτερης κίνησης με βάση την αξιολόγηση που κάνατε.

   iv. Μετακίνηση του παίκτη.

   v. Ανανέωση της μεταβλητής της κλάσης, `path`.

   vi. 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.**

f. Συνάρτηση **`void statistics()`**: Η συνάρτηση αυτή εκτυπώνει στοιχεία για τις κινήσεις του παίκτη σε κάθε γύρο του παιχνιδιού (ζάρι που επέλεξε, ενέργειες που έκανε και πόντους που μάζεψε π.χ. «ο παίκτης στο round 1 έθεσε το ζάρι ίσο με 4, μάζεψε ένα όπλο τύπου πιστόλι, και κέρδισε 0 πόντους»).

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 20x20, με 1 όπλο από το κάθε είδος και για τον κάθε παίκτη (συνολικά 6 όπλα), 10 εφόδια και 8 παγίδες. Τα όπλα βρίσκονται στην περιοχή που περικλείεται από τα πλακίδια με συντεταγμένες (-2,-2), (2,-2), (2,2), (-2,2). Τα εφόδια βρίσκονται περιμετρικά της περιοχής των όπλων στο τετράγωνο που σχηματίζεται από τα πλακίδια με συντεταγμένες (-3,-3), (3,-3), (3,3), (-3,3). Οι παγίδες βρίσκονται περιμετρικά της περιοχής των εφοδίων στο τετράγωνο που σχηματίζεται από τα πλακίδια με συντεταγμένες (-4,-4), (4,-4), (4,4), (-4,4).
- **Η ακτίνα που ορίζει το τμήμα της πίστας που μπορεί να δει ο παίκτης σε κάθε κίνησή του έχει τιμή r=3.**

- Ορισμός 2 παικτών, ένας παίκτης που παίζει με τυχαίες κινήσεις και ένας παίκτης τύπου HeuristicPlayer.
- Οι παίκτες παίζουν εναλλάξ (ρίχνουν το ζάρι και κινούνται στο ταμπλό) μέχρις ότου ένας από τους 2 σκοτωθεί ή το ταμπλό έχει μέγεθος 4x4. Σε περίπτωση που δε σκοτωθεί κάποιος από τους 2, νικητής είναι αυτός που έχει το μεγαλύτερο σκορ. Θα πρέπει λοιπόν, μετά από κάθε κίνηση καθενός από τους 2 παίκτες, να καλείτε τη συνάρτηση `kill()` για να ελέγχεται αν κάποιος από τους 2 παίκτες έχει σκοτώσει τον αντίπαλο. Η συνάρτηση καλείται με πρώτο όρισμα τον παίκτη που παίζει την τρέχουσα στιγμή, δεύτερο όρισμα τον αντίπαλο και τρίτο όρισμα την απόσταση d=2. Σε περίπτωση που γίνει κάτι τέτοιο, εμφανίζεται αντίστοιχο μήνυμα και το παιχνίδι τερματίζει.
- Κλήση της συνάρτησης **`statistics()`** για τον παίκτη τύπου HeuristicPlayer σε κάθε γύρο.
- Το ταμπλό του παιχνιδιού μικραίνει κατά μια μονάδα περιμετρικά κάθε 3 γύρους παιχνιδιού.
- Εκτύπωση του αριθμού των γύρων παιχνιδιού που έπαιξαν οι παίκτες, του σκορ του κάθε παίκτη και το νικητή 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, 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 **Δευτέρα 02 Δεκεμβρίου, 23:59** (ηλεκτρονικά)
Δε θα υπάρξει καμία παρέκκλιση από την παραπάνω προθεσμία.