

Δομές δεδομένων

Εργασία C



Σιμίτσης Βασίλης ΑΕΜ: 8458 Email: [vsimitsis@gmail.com](mailto:vsimitsis@gmail.com)

Κωστινούδης Ευάγγελος ΑΕΜ: 8708 Email: ekostinou@auth.gr

Contents

[1 Περιγραφή προβλήματος 2](#_Toc27697749)

[2 Κλάση Node 2](#_Toc27697750)

[2.1 Μεταβλητές 2](#_Toc27697751)

[2.2 Μέθοδοι 2](#_Toc27697752)

[3 Κλάση MinMaxPlayer 3](#_Toc27697753)

[3.1 Μεταβλητές 3](#_Toc27697754)

[3.2 Μέθοδοι 3](#_Toc27697755)

[4 Κλάση Game 4](#_Toc27697756)

[4.1 Μεταβλητές 4](#_Toc27697757)

[4.2 Μέθοδοι 4](#_Toc27697758)

# Περιγραφή προβλήματος

Στην εργασία αυτή ζητήθηκε η υλοποίηση σε java μίας παραλλαγής τους παιχνιδιού hunger games. Υπάρχουν δύο παίκτες που μπορούν να κινηθούν μία θέση σε κάθε γύρο. Οι παίκτες μπορούν να μαζέψουν όπλα που υπάρχουν στο ταμπλό και εφόδιο. Ακόμα, υπάρχουν παγίδες που μπορούν να πέσουν οι παίκτες. Το ταμπλό μικραίνει μέχρις να γίνει 4x4 όπου τελειώνει το παιχνίδι. Ακόμα, ένας παίκτης μπορεί να σκοτώσει τον αντίπαλο του όπου και πάλι το παιχνίδι τελειώνει.

# Κλάση Node

Η κλάση Node αντιπροσωπεύει ένα κόμβο στο MinMax δένδρο.

## Μεταβλητές

* private Node parent: Ο κόμβος-πατέρας του κόμβου.
* private ArrayList<Node> children: Τα παιδιά του κόμβου.
* private int nodeDepth: Το βάθος του κόμβου στο δέντρο του MinMax Αλγορίθμου.
* private int[] nodeMove: Η κίνηση που αντιπροσωπεύει το κόμβο. Περιέχει την θέση (x,y) του παίκτη και το ζάρι.
* private Board nodeBoard: Το ταμπλό του παιχνιδιού για τη κίνηση που αντιπροσωπεύει ο κόμβος.
* private double nodeEvaluation: Η τιμή της συνάρτησης αξιολόγησης για την κίνηση που αντιπροσωπεύει ο κόμβος.

Για κάθε μεταβλητή υπάρχουν setter και getter μέθοδοι.

## Μέθοδοι

**Setters**:

* void setParent(Node parent)
* void setChildren(ArrayList<Node> children)
* void setNodeDepth(int nodeDepth)
* void setNodeMove(int[] nodeMove)
* void setNodeBoard(Board nodeBoard)
* void setNodeEvaluation(double nodeEvaluation)

**Getters**:

* Node getParent()
* ArrayList<Node> getChildren()
* int getNodeDepth()
* int[] getNodeMove()
* Board getNodeBoard()
* double getNodeEvaluation()

**Constructors**:

* Node(): Κενός constructor.
* Node(Node parent, int nodeDepth, int[] nodeMove, Board nodeBoard, double nodeEvaluation): Constructor με όλα τα ορίσματα τις μεταβλητές της Node εκτός της Children.
* Node(Node parent, ArrayList<Node> children, int nodeDepth, int[] nodeMove, Board nodeBoard, double nodeEvaluation): Constructor με όλα τα ορίσματα τις μεταβλητές της Node.

**Μέθοδοι**:

* void addChild(Node child): Προσθέτει παιδία στη children.

# Κλάση MinMaxPlayer

Η κλάση MinMaxPlayer αντιπροσωπεύει ένα παίκτη που παίζει με τον αλγόριθμο MinMax. Κληρονομεί την κλάση Player.

## Μεταβλητές

* private ArrayList<Integer[]> path: Πληροφορίες για την κάθε κίνηση του παίκτη. Σε κάθε γύρο αποθηκεύονται οι τιμές του ζαριού, των πόντων που κερδίζει-χάνει ο παίκτης αν πήρε πιστόλι, τόξο ή σπαθί, αν μάζεψε εφόδιο και αν έπεσε σε παγίδα ζώων ή σκοινιών.
* private static int r: Η απόσταση που μπορεί να δει κάθε παίκτης.

Για κάθε μεταβλητή υπάρχουν setter και getter μέθοδοι.

## Μέθοδοι

**Setters**:

* void setR(int r)
* void setPath(ArrayList<Integer[]> path)

**Getters**:

* int getR()
* ArrayList<Integer[]> getPath()

**Constructors**:

* MinMaxPlayer(): Κενός constructor.
* MinMaxPlayer(ArrayList<Integer[]> path): Constructor με μοναδικό όρισμα το path.
* MinMaxPlayer(int id, String name, Board board, int score, int x, int y): Constructor με ορίσματα όλες τις τιμές της κλάσης Player εκτός των όπλων.
* MinMaxPlayer(int id, String name, Board board, int score, int x, int y, ArrayList<Integer[]> path): Constructor με ορίσματα όλες τις τιμές της κλάσης Player εκτός των όπλων και του αρχικού path.
* MinMaxPlayer(int id, String name, Board board, int score, int x, int y, Weapon bow, Weapon pistol, Weapon sword): Constructor με ορίσματα όλες τις μεταβλητές της κλάσης Player.
* MinMaxPlayer(int id, String name, Board board, int score, int x, int y, Weapon bow, Weapon pistol, Weapon sword, ArrayList<Integer[]> path): Constructor με ορίσματα όλες τις μεταβλητές της κλάσης Player και του αρχικού path.
* MinMaxPlayer(Player player): Constructor που δέχεται ένα αντικείμενο Player ως είσοδο και αντιγράφει όλες τις τιμές του στο νέο.
* MinMaxPlayer(Player player, ArrayList<Integer[]> path): Constructor που δέχεται αντικείμενο Player και το αρχικό path.
* MinMaxPlayer(MinMaxPlayer minMaxPlayer): Constructor που δέχεται ως όρισμα αντικείμενο MinMaxPlayer και το αντιγράφει.

**Μέθοδοι**:

* float playersDistance(Player p): Επιστρέφει την ευκλείδεια απόσταση του παίκτη από τον αντίπαλο p, αν ο αντίπαλος είναι στο οπτικό πεδίο, αλλιώς επιστρέφει -1.
* double evaluate(int dice, int x, int y, Player opponent): Αξιολογεί την κίνηση του παίκτη για ζάρι dice και τη θέση x,y. Συγκεκριμένα, αν ο παίκτης έχει πιστόλι και ο αντίπαλος είναι στο οπτικό πεδίο κερδίζει (100 / απόσταση από αντίπαλο) πόντους. Αν μαζέψει πιστόλι 10 πόντους, 3 πόντους για σπαθί ή τόξο και τους πόντους που κερδίζει-χάνει αν μαζέψει ή πέσει σε παγίδα. Τέλος, αν βρίσκεται εκτός του τραγώνου [5,5] χάνει (Manhattan απόσταση από το 0,0) / 1000 πόντους.
* static boolean kill(Player player1, Player player2, float d): Αν η απόσταση των παικτών είναι μικρότερη του d επιστρέφει true.
* void statistics(): Τυπώνει τη τελευταία κίνηση του παίκτη. Συγκεκριμένα, τυπώνει το ζάρι που επέλεξε ο παίκτης, ακόμα αν μάζεψε κάποιο όπλο ή κάποιο εφόδιο ή αν έπεσε σε κάποια παγίδα τυπώνει κατάλληλο μήνυμα.
* void createMySubtree(Node root, int depth, int xCurrentPos, int yCurrentPos, int xOpponentCurrentPos, int yOpponentCurrentPos): Προσθέτει στον κόμβο root του MinMax δένδρου τις επιτρεπτές κινήσεις του MinMaxPlayer και καλέι την createOpponentSubtree για να προσθέσει τις κινήσεις του αντιπάλου για κάθε επιτρεπτή κίνηση.
* void createOpponentSubtree(Node parent, int depth, int xCurrentPos, int yCurrentPos, int xOpponentCurrentPos, int yOpponentCurrentPos): Προσθέτει στον κόμβο parent του MinMax δένδρου τις επιτρεπτές κινήσεις του αντιπάλου.
* int chooseMinMaxMove(Node root): Επιλέγει την καλύτερη κίνηση με βάση τον αλγόριθμο MinMax με ρίζα του δένδρου root και επιστρέφει την επιστρέφει.
* int[] getNextMove(int xCurrentPos, int yCurrentPos, int xOpponentCurrentPos, int yOpponentCurrentPos): Βρίσκει την καλύτερη κίνηση του παίκτη βάση του αλγορίθμου MinMax. Εκτελεί την κίνηση και επιστρέφει τη νέα θέση του παίκτη.

# Κλάση Game

Η κλάση Game αντιπροσωπεύει το παιχνίδι.

## Μεταβλητές

* int round: Ο γύρος του παιχνιδιού.

Για κάθε μεταβλητή υπάρχουν setter και getter μέθοδοι.

## Μέθοδοι

**Setters**:

* void setRound(int round)

**Getters**:

* int getRound()

**Constructor**:

* Game(): Θέτει το γύρο τη τιμή 0.

**Main**:

Δημιουργεί το παιχνίδι, το ταμπλό και τους δύο παίκτες (αντικείμενα Game, Board, Player, MinMaxPlayer αντίστοιχα). Οι αρχικές θέσεις των παικτών είναι κάτω δεξιά γωνία του ταμπλό. Ακόμα το αρχικό σκορ τους είναι 15. Θέτει τα όρια των περιοχών των όπλων, εφοδίων και παγίδων. Για κάθε γύρο, μετακινεί του παίκτες και εμφανίζει τον γύρο, το ταμπλό και τις κινήσεις των παικτών. Σταματάει όταν το ταμπλό έχει μέγεθος 4x4 ή όταν ένας παίκτης σκοτωθεί ή αν ένας παίκτης έχει αρνητικό σκορ. Τέλος, εμφανίζει τους πόντους κάθε παίκτη και το αποτέλεσμα του παιχνιδιού.