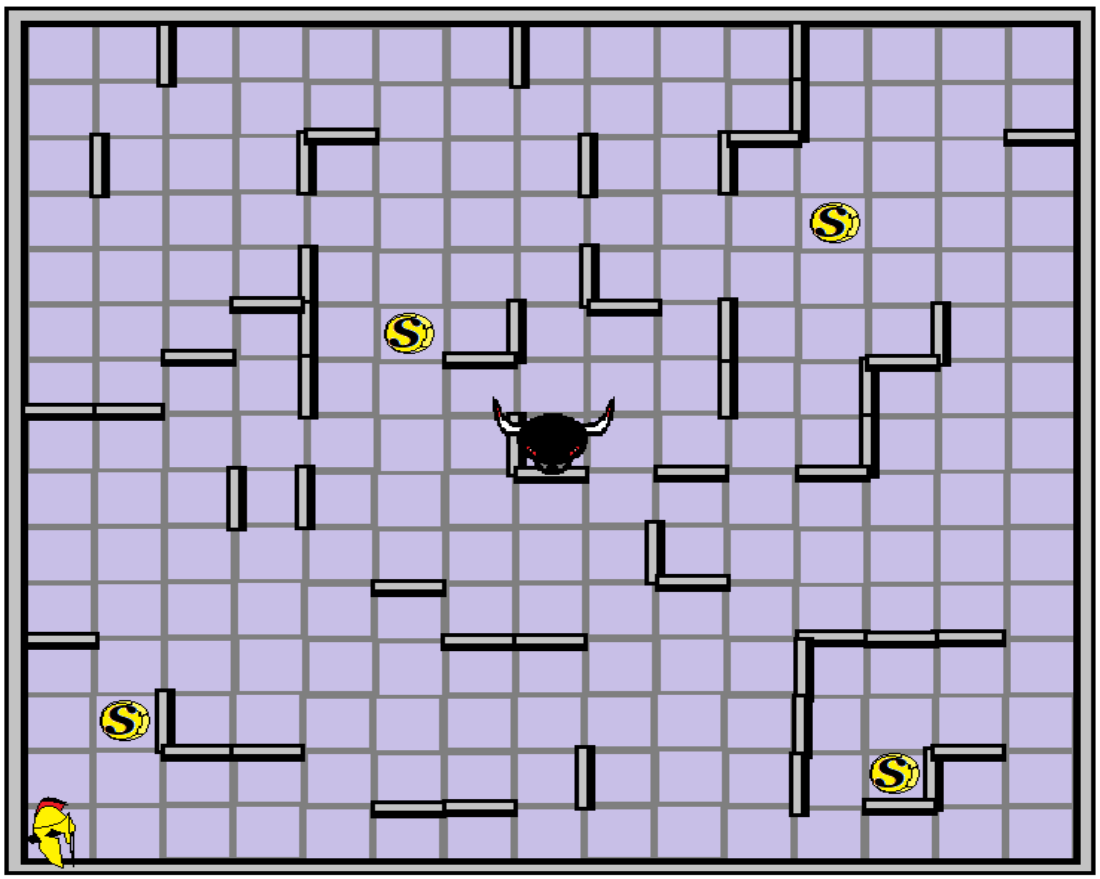
THESEUS VS MINOTAUR

Εργασία Α



Ταμπλό διάστασης 15x15, με 4 εφόδια

Στοιχεία μελών της ομάδας:

Ονοματεπώνυμο: Τσιανάκας Εμμανουήλ ΑΕΜ: 9992
κινητό: 6976903077 Email: tsianakas@ece.auth.gr

Ονοματεπώνυμο: Παπούλιας Μιχαήλ ΑΕΜ: 10204
κινητό: 6949823576 Email: mpapouli@ece.auth.gr

<u>Σύντομη περιγραφή του παιχνιδιού:</u>

Κατά την έναρξη του παιχνιδιού, δημιουργείται ένα ταμπλό διάστασης 15x15 (225 πλακίδια), το οποίο περιέχει 4 εφόδια καθώς και ορισμένο αριθμό τειχών, τυχαία τοποθετημένα στο χώρο. Σε κάθε πλακίδιο μπορούν να τοποθετηθούν το πολύ 2 τείχη, ενώ στο περίγραμμα του ταμπλό υπάρχει υποχρεωτικά τείχος. Ταυτόχρονα, έχουμε και δημιουργία 2 παικτών (του Θησέα και του Μινώταυρου), οι οποίοι μπορούν να κινούνται εναλλάξ πάνω στο ταμπλό. Αρχικά, ο Θησέας βρίσκεται στην κάτω αριστερή γωνία του λαβυρίνθου, ενώ ο Μινώταυρος βρίσκεται στο μέσον του. Οι δύο παίκτες δεν μπορούν να διαπεράσουν τα τείχη, εάν αποφασίσουν να κινηθούν προς μία τέτοια κατεύθυνση. Στόχος του Θησέα είναι να συλλέξει και τα 4 εφόδια προτού τον βρει ο Μινώταυρος. Το παιχνίδι τελειώνει με νικητή το Θησέα, εάν ο ίδιος καταφέρει να συλλέξει τα 4 εφόδια και αποφύγει να συναντηθεί στο ίδιο πλακίδιο με το Μινώταυρο, προτού παρέλθουν 200 γύροι παιχνιδιού. Λήγει με νικητή το Μινώταυρο, εάν καταφέρει να βρεθεί στο ίδιο πλακίδιο με τον Θησέα, πριν τους 200 γύρους. Σε αντίθετη περίπτωση, το παιχνίδι λήγει με ισοπαλία.

Κλάση Tile:

Η κλάση αυτή αποτελεί το πλακίδιο του ταμπλό και περιέχει τις εξής μεταβλητές:

int tileId (το id του πλακιδίου),
int x (η τετμημένη του πλακιδίου),
int y (η τεταγμένη του πλακιδίου),
boolean up (η τοποθέτηση ή μη ενός άνω τείχους στο πλακίδιο),
boolean down (η τοποθέτηση ή μη ενός κάτω τείχους στο πλακίδιο),
boolean left (η τοποθέτηση ή μη ενός αριστερού τείχους στο πλακίδιο),
boolean right (η τοποθέτηση ή μη ενός δεξιού τείχους στο πλακίδιο).

Η Tile περιέχει 3 κατασκευαστές (constructors). Ο πρώτος αρχικοποιεί τις int μεταβλητές στο “μηδέν” και τις boolean στο “false”. Ο δεύτερος κατασκευαστής λαμβάνει ως όρισμα όλες τις μεταβλητές της κλάσης αυτής και τις προσπελάζει με τη χρήση του τελεστή “this”, ενώ ο τρίτος λαμβάνει ως όρισμα και δημιουργεί ένα αντικείμενο ob τύπου Tile.
Τέλος, η κλάση αυτή περιέχει τους αντίστοιχους getters και setters όλων των μεταβλητών της.

Κλάση Supply:

Η κλάση αυτή αποτελεί το εφόδιο που τοποθετείται στο ταμπλό και περιέχει τις εξής μεταβλητές:

int supplyId (το id του εφοδίου),
int x (η τετμημένη του εφοδίου),
int y (η τεταγμένη του εφοδίου),
int supplyTileId (το id του πλακιδίου στο οποίο βρίσκεται το εφόδιο).

Η Supply περιέχει 3 κατασκευαστές (constructors). Ο πρώτος αρχικοποιεί τις int μεταβλητές στο “μηδέν”. Ο δεύτερος κατασκευαστής λαμβάνει ως όρισμα όλες τις μεταβλητές της κλάσης αυτής και τις προσπελάζει με τη χρήση του τελεστή “this”, ενώ ο τρίτος λαμβάνει ως όρισμα και δημιουργεί ένα αντικείμενο ob τύπου Supply.
Τέλος, η κλάση αυτή περιέχει τους αντίστοιχους getters και setters όλων των μεταβλητών της.

Κλάση Board:

Η κλάση αυτή αποτελεί το ταμπλό του παιχνιδιού και περιέχει τις εξής μεταβλητές:

int N (η διάσταση του ταμπλό),
int S (ο αριθμός των εφοδίων που δημιουργούνται αρχικά),
int W (ο συνολικός αριθμός των τειχών του λαβυρίνθου),
Tile[] tiles (ένας πίνακας που περιέχει αντικείμενα τύπου Tile),
Supply[] supplies (ένας πίνακας που περιέχει αντικείμενα τύπου Supply).

Η Board περιέχει 3 κατασκευαστές (constructors). Ο πρώτος αρχικοποιεί τις int μεταβλητές στο “μηδέν”. Ο δεύτερος κατασκευαστής λαμβάνει ως όρισμα όλες τις μεταβλητές της κλάσης αυτής και τις προσπελάζει με τη χρήση του τελεστή “this”, ενώ ο τρίτος λαμβάνει ως όρισμα και δημιουργεί ένα αντικείμενο ob τύπου Board.
Περιέχει επίσης τους αντίστοιχους getters και setters όλων των μεταβλητών της, καθώς και τις συναρτήσεις:
createTile()
createSupply()
createBoard()
getStringRepresentation()

-Μέθοδος createTile()
Δημιουργεί συνολικά NxN tiles, τα οποία αρχικοποιεί με μία “διπλή for”, με βάση την αρίθμηση που ζητείται: (πχ το πρώτο πλακίδιο έχει id = 0, x = 0, y = 0 και καθόλου τείχη(false), το δεύτερο έχει id = 1, x = 1, y = 0 και καθόλου τείχη(false) κλπ). Η μεταβλητή id αποτελεί το id κάθε πλακιδίου, το i αποτελεί την τετμημένη και το j την τεταγμένη του:

```
for(i=0; i<N; i++)
{
        for (j=0; j<N; j++)
        {
                tiles[id] = new Tile(id, j, i, false, false, false, false);
                id++;
        }
}
```

-Μέθοδος createSupply()
Περιέχει μία λίστα μεγέθους S (όσο ο αριθμός των supply), η οποία τοποθετεί κάθε Supply σε ένα τυχαίο πλακίδιο του ταμπλό, με τον αυστηρό περιορισμό το πλακίδιο εκείνο να είναι κενό από παίκτη ή τυχόν άλλο εφόδιο. Ο περιορισμός αυτός γίνεται μέσα στην “if”, όπου αρχικοποιείται κατάλληλα το εφόδιο εφόσον το πλακίδιο εκείνο είναι κενό από άλλο εφόδιο και εφόσον ο τυχαίος αριθμός “rand_int” είναι διαφορετικός του 0 (αρχική θέση Θησέα) και του (N*N)/2 (αρχική θέση Μινώταυρου):

```
if(arr.indexOf(rand_int) == -1 && rand_int != 0 && rand_int != (int)(N*N)/2)
{
        supplies[i] = new Supply(i , tiles[rand_int].getx() , tiles[rand_int].gety() , rand_int);
        arr.add(rand_int);
}
```

Σε αντίθετη περίπτωση, η τιμή του i μειώνεται κατά ένα, προκειμένου να δημιουργηθεί το εφόδιο πληρώντας όλες τις

προϋποθέσεις (έτσι έχουμε πάντοτε δημιουργία ακριβώς 4 εφοδίων ορθά τοποθετημένα στο χώρο).

-Μέθοδος createBoard()
Περιέχει τις μεταβλητές i (το id του πλακιδίου) και την leftW (ο μέγιστος αριθμός τυχαίων εσωτερικών τειχών που μπορούν να δημιουργηθούν).
Αρχικά, δημιουργούνται τα εξωτερικά τείχη στο περίγραμμα του ταμπλό (εκτός της εισόδου, η οποία κλείνει κατά τη δημιουργία της συνάρτησης main()) με συνολικά 4 “for” – μία για κάθε κατεύθυνση. Στη συνέχεια έχουμε την κατάλληλη αρχικοποίηση του leftW, το οποίο αποτελεί τη διαφορά των εξωτερικών τειχών που μόλις δημιουργήθηκαν (4*N-1) από το μέγιστο αριθμό επιτρεπτών τειχών πάνω στο ταμπλό (W). Έτσι: int leftW = W-(4*N-1)

Στη συνέχεια δημιουργείται μία συνθήκη do-while(η οποία επαναλαμβάνεται, όσο το leftW > 0. Μέσα στη συνθήκη αυτή δημιουργείται ένας τυχαίος αριθμός στο διάστημα [1, N*N-1), ο οποίος αποτελεί το id του πλακιδίου στο οποίο θα δουλέψουμε. Δε λάβαμε υπόψη το πλακίδιο με id=0, καθώς αυτό θα γεμίσει με κάτω τείχος κατά τη δημιουργία της main(). Ελέγχουμε αρχικά τον αριθμό των τειχών που υπάρχουν στο πλακίδιο αυτό, με τη χρήση 4 “for”. Ορίζουμε προηγουμένως μία μεταβλητή sumidW = 0. Εάν το πλακίδιο έχει τείχος πχ στο άνω μέρος του, τότε η sumidW αυξάνεται κατά ένα. Όμοια και στις υπόλοιπες 4 “for”. Έχουμε τώρα άλλες 4 “for” ίδιας λογικής, η οποίες αποφασίζουν εάν θα τοποθετηθεί τελικά τείχος πάνω, κάτω, δεξιά ή αριστερά του πλακιδίου. Θα αναλύσουμε μόνο τη πρώτη “for”, η οποία σχετίζεται με την τοποθέτηση ή μη ενός κάτω τείχους στο πλακίδιο με το τυχαίο id, στο οποίο δουλεύουμε. Αρχικά γίνεται έλεγχος εάν το sumidW < 2 (περιορισμός από την εκφώνηση της εργασίας-κάθε πλακίδιο έχει το πολύ 2 τείχη), εάν το leftW > 0 (εάν υπάρχει διαθέσιμο προς τοποθέτηση τείχος), καθώς και εάν το πλακίδιο που βρισκόμαστε δεν έχει ήδη τείχος στο κάτω μέρος του. Εάν οι 3 παραπάνω συνθήκες ισχύουν, τότε δημιουργείται ένας boolean τυχαίος αριθμός (0 ή 1). Εισερχόμαστε σε μία νέα “if”, η οποία ελέγχει κάτω πλακίδιο (θα πρέπει να υπάρχει κάτω πλακίδιο, το άθροισμα των συνολικών τειχών του να είναι μικρότερο του 2, καθώς και ο boolean αριθμός να είναι ίσος με 1). Εάν και τα τρία

παραπάνω ισχύουν ταυτόχρονα, τότε τοποθετείται τείχος στο κάτω μέρος του πλακιδίου που δουλεύουμε, τοποθετείται τείχος στο πάνω μέρος του πλακιδίου που βρίσκεται κάτω από αυτό στο οποίο δουλεύουμε, το sumidW αυξάνεται κατά 1, ενώ το leftW μειώνεται κατά 2(μιας και τοποθετήσαμε <<διπλό τείχος>>). Σε αντίθετη περίπτωση (εάν η "if" δεν ισχύει, τότε απλά μειώνουμε κατά ένα τον αριθμό leftW.

Όμοια λογική ακολουθούν και οι άλλες 3 μεγάλες "for". Η συνθήκη με τις 4 μεγάλες "for" παύει να ελέγχεται, όταν το leftW γίνει μηδέν.

-Μέθοδος getStringRepresentation()
Η συνάρτηση αυτή αρχικοποιεί και επιστρέφει έναν δισδιάστατο πίνακα τύπου String, μεγέθους (2*N+1) x (N), ο οποίος αποτελεί την αναπαράσταση του ταμπλό. Δημιουργούμε λοιπόν έναν τέτοιο πίνακα array, αρχικοποιούμε το id = 0 και εισερχόμαστε σε μία διπλή "for", η οποία <<σκανάρει όλο το ταμπλό>>. Αρχικά, εάν το j = 0, το στοιχείο array[id] του πίνακα array αρχικοποιείται με βάση τις συνθήκες που επικρατούν στο πλακίδιο που βρισκόμαστε (ύπαρξη τείχους σε κάποια κατεύθυνση), συμπεριλαμβανομένου και όλων των αριστερών συνθηκών (πχ τείχος στα αριστερά), ενώ, εάν το j>0, πάλι αρχικοποιείται το στοιχείο array[id] του πίνακα array με την ίδια λογική, με τη διαφορά όμως ότι τώρα οι αριστερές συνθήκες δεν συμπεριλαμβάνονται στην εκτύπωση (Η διαδικασία αυτή γίνεται ώστε να έχουμε μορφοποιημένη έξοδο, πχ να μην εμφανίζεται 2 φορές το ίδιο τείχος('|') ή το ίδιο σύμβολο ('+').
Με την ίδια ακριβώς λογική δουλεύουμε και για την αρχικοποίηση όλων των υπολοίπων συνθηκών (εμφάνιση του Θησέα (T) στο πλακίδιο, εμφάνιση του Μινώταυρου (M) στο πλακίδιο, εμφάνιση ενός εφοδίου (S) στο πλακίδιο, εμφάνιση ενός εφοδίου (S) και του Μινώταυρου (M) στο πλακίδιο-εάν ο μινώταυρος βρεθεί σε ένα πλακίδιο που περιέχει εφόδιο).
Το id μηδενίζεται στην αρχή της κάθε μεγάλης "διπλής for", ενώ αυξάνεται κατά ένα στο τέλος κάθε "διπλής for", ώστε να αρχικοποιηθούν όλα τα πλακίδια με τον κατάλληλο συμβολισμό τύπου-String. Τέλος, επιστρέφεται ο δισδιάστατος πίνακας array.

Κλάση Player:
Η κλάση αυτή αποτελεί το τον παίκτη και περιέχει τις εξής μεταβλητές:

int playerId (το id του παίκτη),
String name(το όνομα του παίκτη)
Board board(το ταμπλό του παιχνιδιού)
int score(το σκορ του παίκτη)
int x (την τετμημένη του παίκτη),
int y (την τεταγμένη του παίκτη),

Η Player περιέχει 3 κατασκευαστές (constructors). Ο πρώτος αρχικοποιεί τις int μεταβλητές στο “μηδέν”, τη string μεταβλητή σε κενό(‘ ’) και τη board μεταβλητή σε δημιουργία νέου ταμπλό. Ο δεύτερος κατασκευαστής λαμβάνει ως όρισμα όλες τις μεταβλητές της κλάσης αυτής και τις προσπελάζει με τη χρήση του τελεστή “this”, ενώ ο τρίτος λαμβάνει ως όρισμα και δημιουργεί ένα αντικείμενο ob τύπου Player.
Περιέχει επίσης τους αντίστοιχους getters και setters όλων των μεταβλητών της, καθώς και τη συνάρτηση:
move()

-Μέθοδος move()
Η συνάρτηση παίρνει ως όρισμα το id του πλακιδίου στο οποίο βρίσκεται ο παίκτης πριν την κίνησή του, αρχικοποιεί και επιστρέφει έναν μονοδιάστατο ακέραιο πίνακα μεγέθους 4. Αρχικά, δημιουργείται ένας τυχαίος αριθμός στο διάστημα: {1, 3, 5, 7}, ο οποίος συμβολίζει την κίνηση που καλείται να εκτελέσει ο παίκτης (1=πάνω, 3=δεξιά, 5=κάτω, 7=αριστερά). Έπειτα, δημιουργείται ένας ακέραιος πίνακας array μεγέθους 4, όπου:
array[0] = το id του πλακιδίου στο οποίο θα βρεθεί ο παίκτης μετά την κίνησή του,
array[1] = η τετμημένη του πλακιδίου στο οποίο θα βρεθεί ο παίκτης μετά την κίνησή του,
array[2] = η τεταγμένη του πλακιδίου στο οποίο θα βρεθεί ο παίκτης μετά την κίνησή του,
array[3] = το σκορ του παίκτη, μετά την κίνησή του.

Στη συνέχεια ακολουθούν οι 4 μεγάλες “if”, οι οποίες σχετίζονται με την κίνηση του παίκτη εάν η ζαριά του είναι 1, 3, 5 ή 7, αντίστοιχα για κάθε “for”. Και οι 4 “if” έχουν την ίδια ακριβώς λογική, επομένως θα αναλύσουμε μόνο την πρώτη, η οποία σχετίζεται με τη δυνατότητα ή μη της κίνησης του παίκτη πάνω:

Η πρώτη “if” ελέγχει εάν η ζαριά του παίκτη είναι ίση με 1 (εντολή κίνησης πάνω). Εάν ισχύει, τότε εισερχόμαστε στη δεύτερη “if”, η οποία ελέγχει εάν υπάρχει τείχος στο πάνω μέρος του πλακιδίου.

Εάν η συνθήκη αυτή ισχύει, τότε εμφανίζεται στην οθόνη το μήνυμα: <<παίκτης δεν μπορεί να κινηθεί πάνω>>, οπότε ο πίνακας array αρχικοποιείται με τα δεδομένα του πλακιδίου στο οποίο βρισκόταν ο παίκτης πριν την κίνησή του, δηλαδή:

```
array[0] = id;
array[1] = board.tiles[id].getx();
array[2] = board.tiles[id].gety();
array[3] = score;
```

εάν η συνθήκη αυτή δεν αληθεύει, τότε ελέγχεται εάν δεν υπάρχει τείχος στο πάνω μέρος του πλακιδίου και επιπλέον, εάν υπάρχει πλακίδιο πάνω από το πλακίδιο στο οποίο βρίσκεται ο παίκτης αρχικά. Στη περίπτωση που και οι δύο συνθήκες ισχύουν, εμφανίζεται στην οθόνη το μήνυμα: <<παίκτης κινήθηκε πάνω>>, και αρχικοποιούνται τα 3 πρώτα στοιχεία του πίνακα array με βάση τα δεδομένα του άνω πλακιδίου στο οποίο βρίσκεται, πλέον, ο παίκτης, ως εξής:

```
array[0] = id + N;
array[1] = board.tiles[id+N].getx();
array[2] = board.tiles[id+N].gety();
```

Για το 4° στοιχείο του πίνακα, πρέπει να ελέγξουμε πρώτα εάν ο παίκτης έχει id=0 (Θησέας) και εάν στο πάνω πλακίδιο βρίσκεται κάποιο από τα 4 εφόδια. Εάν και οι 2 συνθήκες ισχύουν, τότε αρχικά αυξάνουμε το σκορ κατά ένα, μηδενίζουμε τις συντεταγμένες του συγκεκριμένου εφοδίου:

```
board.supplies[i].setx(-1);
board.supplies[i].sety(-1);
board.supplies[i].setsupplyTileId(-1);
```

και στη συνέχεια θέτουμε το 4° στοιχείο του πίνακα ίσο με το score (το οποίο έχει αυξηθεί μέσα στη συνθήκη αυτή). Διαφορετικά, εάν η τελευταία “if” δεν ισχύει, αρχικοποιούμε το 4° στοιχείο του πίνακα πάλι ίσο με score (τώρα όμως το σκορ δεν έχει αυξηθεί).

Όμοια λογική ακολουθούν και οι υπόλοιπες 3 μεγάλες “if”. Είναι σίγουρο ότι θα τρέξει μόνον μία από τις 4 αυτές συνθήκες. Στο τέλος των 4 “if” επιστρέφεται ο πίνακας array.

Κλάση Game:

Η κλάση αυτή αποτελεί το ίδιο το παιχνίδι και έχει τη μεταβλητή:

int round(ο γύρος του παιχνιδιού).

Η Game περιέχει 3 κατασκευαστές (constructors). Ο πρώτος αρχικοποιεί την int μεταβλητή στο “μηδέν”. Ο δεύτερος κατασκευαστής λαμβάνει ως όρισμα τη μεταβλητή της κλάσης αυτής και τη προσπελάζει με τη χρήση του τελεστή “this”, ενώ ο τρίτος λαμβάνει ως όρισμα και δημιουργεί ένα αντικείμενο ob τύπου Game.
Περιέχει επίσης το αντίστοιχο getter και setter της μεταβλητής round, καθώς και τη συνάρτηση:
main()

-Μέθοδος main()
Στη συνάρτηση αυτή γίνονται οι απαραίτητες αρχικοποιήσεις και εκτελούνται οι τελικοί έλεγχοι, ώστε να καταστεί ο κώδικας λειτουργικός. Αρχικά, θέτουμε τη διάσταση N του ταμπλό ίση με 15, ορίζουμε 4 εφόδια S, το συνολικό δυνατό αριθμό τειχών W = (N*N*3+1) / 2, δημιουργούμε τα αντικείμενα τύπου Tile(πλακίδια) , τα αντικείμενα τύπου Supply (εφόδια), το Board (ταμπλό) και δύο Players(παίκτες) a και b. Τέλος, δημιουργούμε το Game του παιχνιδιού και εκτυπώνουμε την αρχική μορφή του λαβυρίνθου μαζί με τους παίκτες και τα εφόδια, ως εξής:

```
System.out.println("The board game:\n");

for(i=2*N; i>=0; i--)
{
        for(j=0; j<N; j++)
        {
                System.out.print(board.getStringRepresentation(0, N*N/2)[i][j]);
        }
        System.out.println();
}
```

Χρησιμοποιούμε δύο ακέραιες μεταβλητές th, min, οι οποίες αντιπροσωπεύουν τις αρχικές θέσεις του Θησέα και του Μινώταυρου, αντίστοιχα.

Με έναν έλεγχο τύπου do-while αρχικά τοποθετούμε τείχος στην είσοδο του λαβυρίνθου, ώστε να μη μπορεί κανένας από τους 2 παίκτες να βγει εκτός ορίων του ταμπλό, στη συνέχεια αυξάνουμε το γύρο του παιχνιδιού κατά ένα και τον εμφανίζουμε στην οθόνη.

Πριν την έναρξη των γύρων, θέτουμε ως πρώτο παίκτη που θα παίξει τον Θησέα. Στο τέλος του γύρου γίνεται αλλαγή του παίκτη που θα παίξει αμέσως μετά. Εάν είναι η σειρά του Θησέα να παίξει, το εμφανίζουμε στην οθόνη, καθώς επίσης και το id του πλακιδίου, την τετμημένη, την τεταγμένη και το σκορ του Θησέα αμέσως μετά την κίνησή του (δηλ. καλούμε τη συνάρτηση move()). Αμέσως μετά καλούμε τη συνάρτηση εκτύπωσης του λαβυρίνθου στην οθόνη.

Όμοια, εάν είναι η σειρά του Μινώταυρου να παίξει.

Η συνθήκη do-while τερματίζει, όταν μία από τις παρακάτω συνθήκες ισχύσει:

1) Ο Θησέας και ο Μινώταυρος βρεθούν στο ίδιο πλακίδιο (κερδίζει ο Μινώταυρος):

```
if(th == min)
{
	if(a.getScore() == S)
	{
		System.out.println("Theseus picked up all supplies!\nTheseus won!");
		break;
	}
	else
	{
		System.out.println("Minotaur killed Theseus.\nMinotaur won.");
		break;
	}
}
```

2) Το σκορ του Θησέα γίνει ίσο με τον αριθμό των εφοδίων (ο Θησέας τότε έχει μαζέψει όλα τα εφόδια – κερδίζει ο Θησέας):

```
if(a.getScore() == S)

{

	System.out.println("Theseus picked up all supplies!\nTheseus won!");

	break;

}
```

3) Παρέλθουν 200 γύροι (δεν κερδίζει κανένας-ισοπαλία):

```
if(game.getRound() == 200)
{
    System.out.println("Draw.");
    break;
}
```

Κατά την έξοδο από τη do-while εμφανίζεται μήνυμα τέλους και το παιχνίδι τελειώνει.