**ΤΕΧΝΗΤΗ ΝΟΗΜΟΣΥΝΗ**

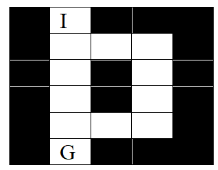
*ΟΝΟΜΑΤΕΠΩΝΥΜΟ: ΝΙΚΟΛΑΟΣ ΡΗΓΑΚΗΣ​*

*ΑΡΙΘΜΟΣ ΜΗΤΡΩΟΥ: Π17109​*

*E-MAIL:*[*nikos.rigakis@gmail.com*](mailto:nikos.rigakis@gmail.com)*​*

*ΥΠΕΥΘΥΝΟΣ ΚΑΘΗΓΗΤΗΣ: ΔΗΜΗΤΡΗΣ ΑΠΟΣΤΟΛΟΥ*

Είμαι στην 3η ομάδα, οπότε έχω αναλάβει το πρόγραμμα για τον 3ο λαβύρινθο.

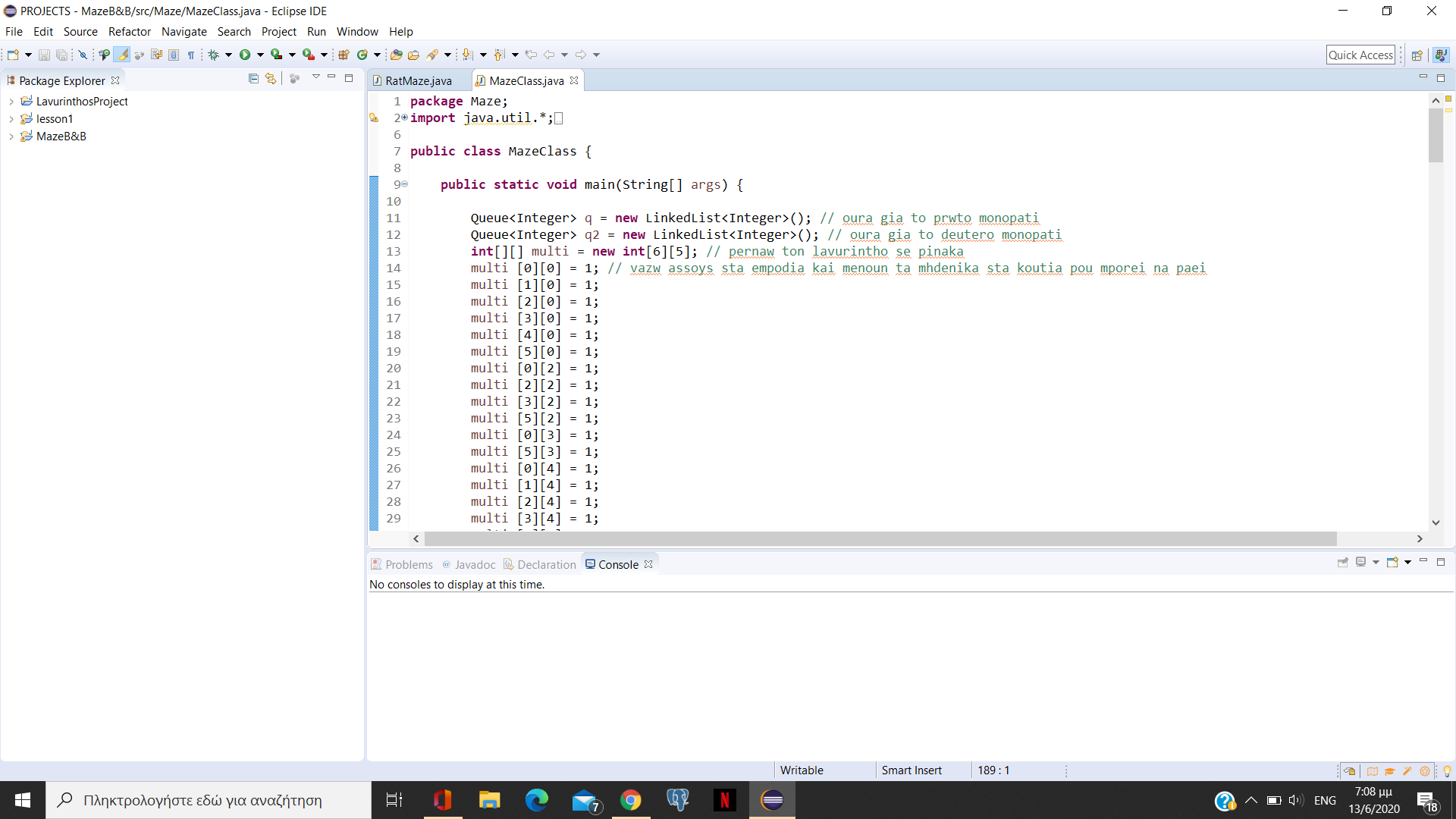


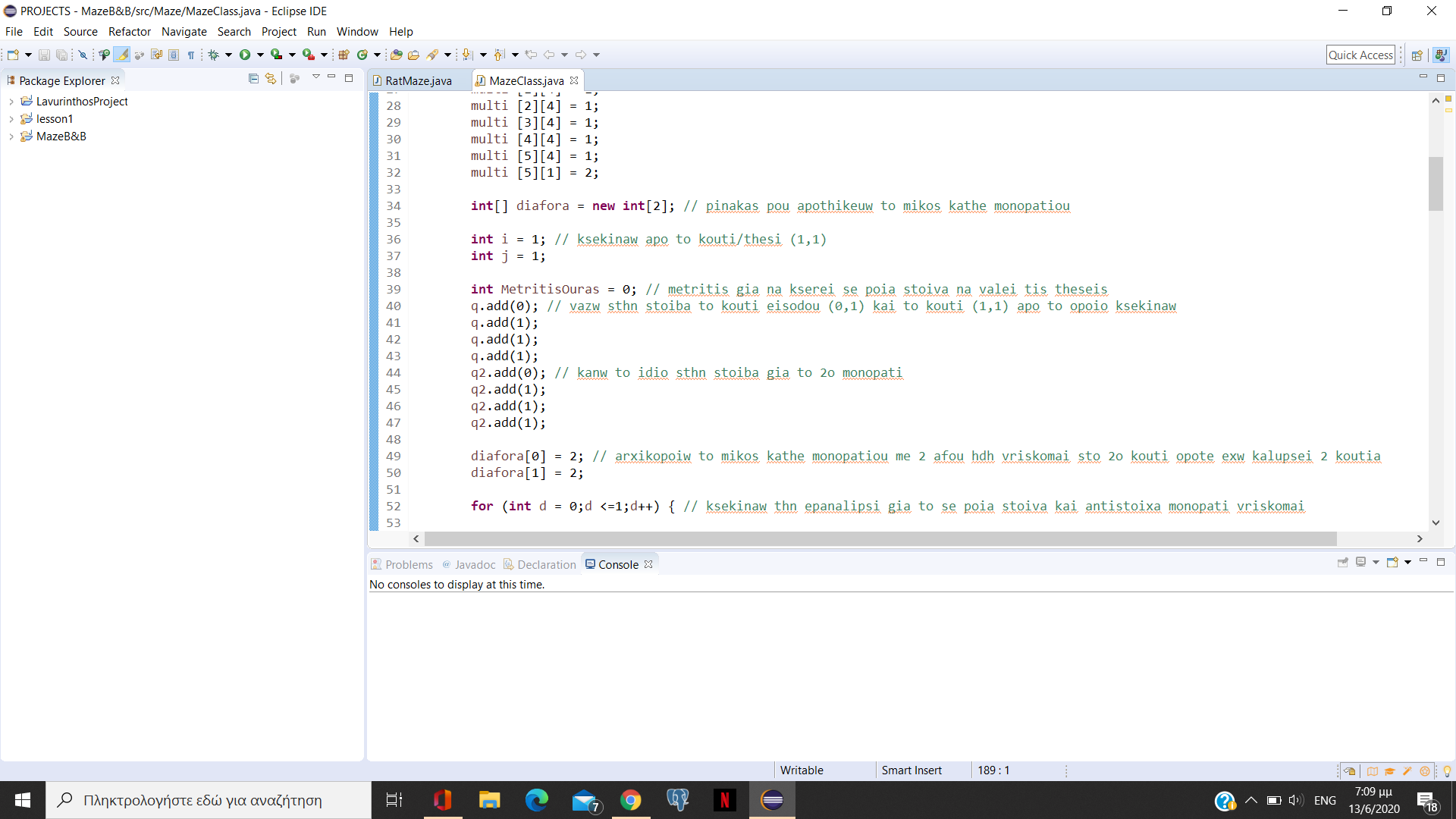
\*\* Για κάποιο λόγο το αρχείο αποθηκεύτηκε στο Intelji ενώ το έκανα στο eclipse. Γιαυτό έχω βάλει και ένα έγγραφο κειμένου στο φάκελο που σας έστειλα με ολόκληρο τον κώδικα για να το τρέξετε εσείς σε περίπτωση που δεν τρέχει το project.

**ΓΕΝΙΚΗ ΙΔΕΑ**

Ο γενικός τρόπος που λειτουργεί ο αλγόριθμος μου είναι:​

* Τοποθετώ τον λαβύρινθο σε δισδιάστατο πίνακα [6,5]
* Βάζω 0 στα κουτιά που μπορώ να πάω, 1 σε αυτά που δεν μπορώ να πάω και 2 στο κουτί εξόδου
* Σε 2 επαναλήψεις βάζω σε 2 στοίβες τα 2 διαφορετικά μονοπάτια και στο τέλος συγκρίνω τα μήκη τους





Έχω την public class MazeClass, στην οποία υπάρχει η public static void main, όπου υπάρχει όλος ο κώδικας.

Αρχικά δημιουργώ 2 ουρές q και q2 και στην συνέχεια τον πίνακα multi[6][5], ο οποίος αντικατοπτρίζει ολόκληρο τον λαβύρινθο. Επίσης δίνω την τιμή 1 στα κελιά και αντίστοιχα στα κουτιά του λαβυρίνθου από τα οποία δεν μπορώ να περάσω. Τα βατά κουτιά έχουν τον αριθμό 0.

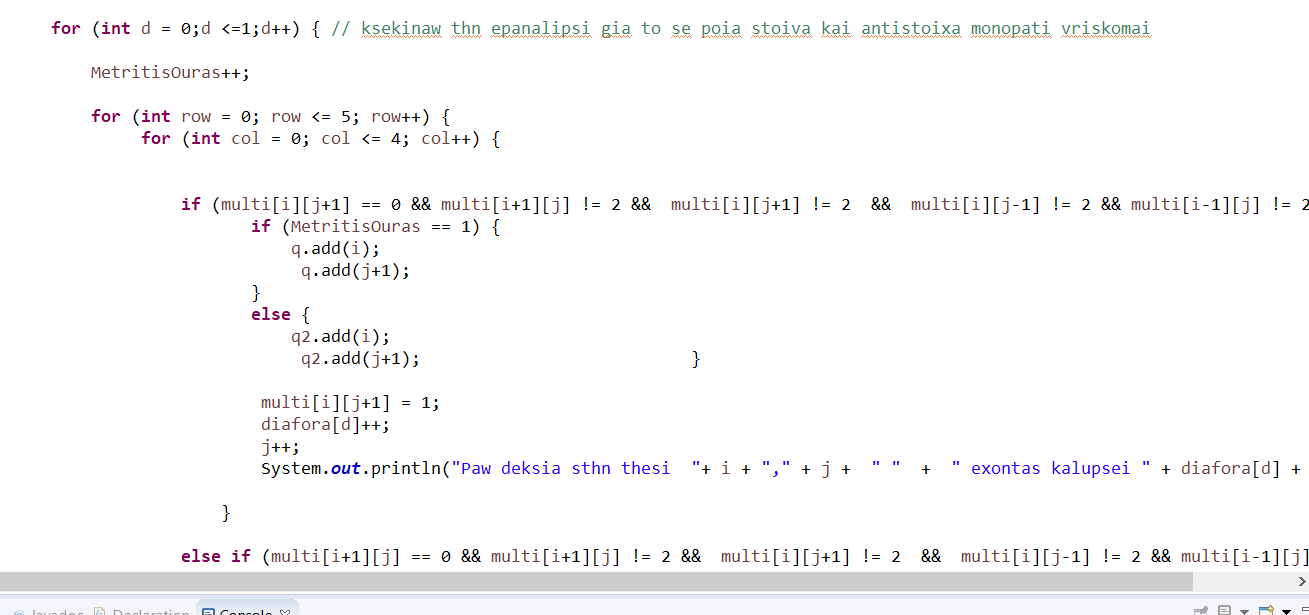
Μετά φτιάχνω έναν πίνακα diafora[2], ο οποίος είναι αυτός που αποθηκεύει το μήκος κάθε μονοπατιού καθώς και μια μεταβλητή-μετρητή MetritisOuras η οποία βοηθάει το πρόγραμμα να ξέρει σε ποια ουρά να τοποθετήσει τις θέσεις του λαβυρίνθου-πίνακα. Την αρχικοποιώ με 0.

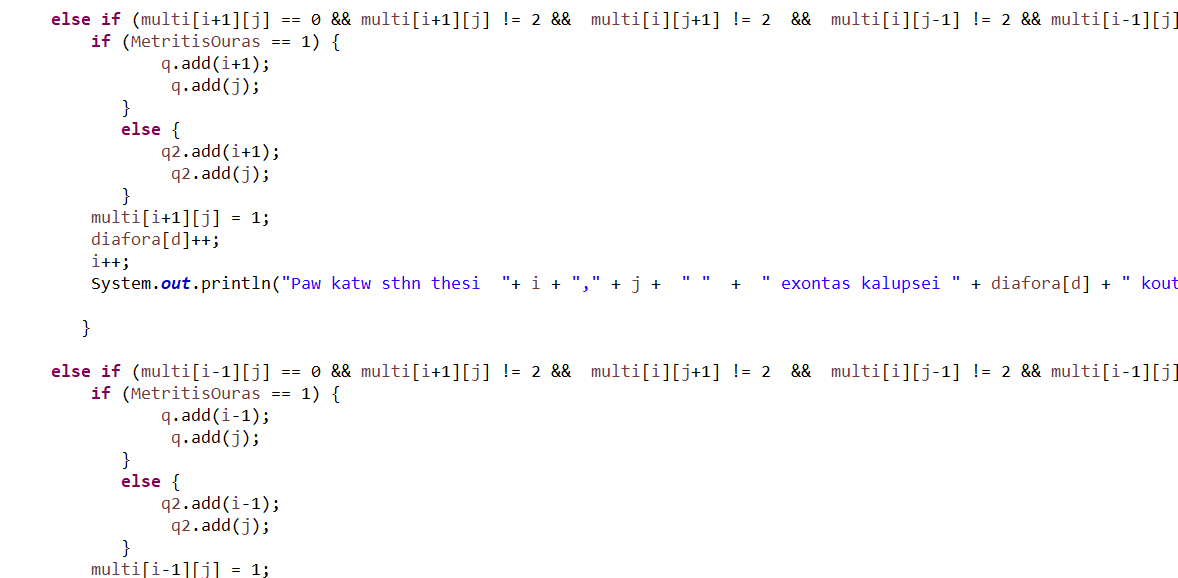
Αρχικοποιώ τα ι και j με 1 έτσι ώστε να ξεκινήσει από το κουτί-θέση (1,1) έτσι ώστε να μπορεί να ελέγξει τα γειτονικά του.

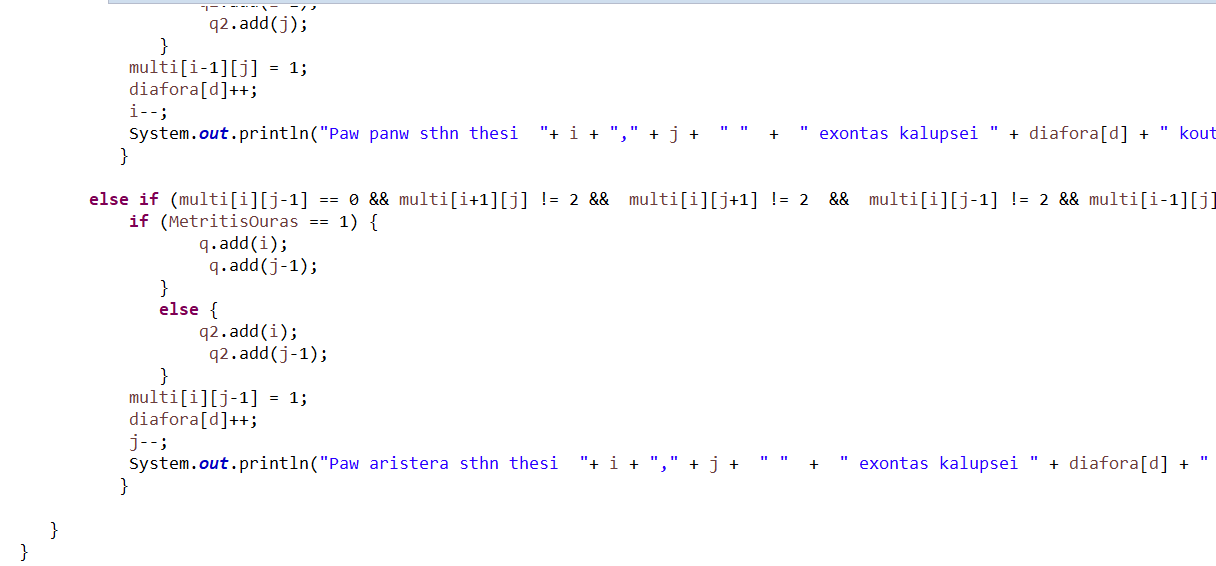
Οπότε, αφού βλέπουμε ότι οι 2 πρώτες κινήσεις είναι σίγουρα το κουτί εισόδου (0,1) πρώτα και το κουτί-θέση (1,1) δεύτερο τότε τα τοποθετούμε στις 2 ουρές.

Έτσι ξεκινάμε με τα μήκη των 2 μονοπατιών να είναι 2.

Πάμε τώρα στις επαναλήψεις.







Αρχικά ξεκινάω μια επανάληψη που γίνεται 2 φορές όσα είναι και τα μονοπάτια, έτσι ώστε να ξεχωρίζω σε ποια ουρά θα μπουν οι θέσεις του πίνακα. Μόλις μπαίνω στην επανάληψη αυξάνω τον μετρητή MetritisOuras κατά 1.

Στην συνέχεια ξεκινάω να διασχίζω τον λαβύρινθο-πίνακα σε μία εμφωλευμένη επανάληψη από 0 μέχρι 5 (γραμμές) και από 0 μέχρι 4(στήλες).

Μέσα εχω 4 μεγάλα if τα οποία ελέγχουν τα γειτονικά κουτιά από αυτό που βρισκάμαστε στην αρχή δηλαδή το (1,1). Συγκεκριμένα:

**if** (multi[i][j+1] == 0 && multi[i+1][j] != 2 && multi[i][j+1] != 2 && multi[i][j-1] != 2 && multi[i-1][j] != 2)

**Η πρώτη συνθήκη** αφορά το δεξιά γειτονικό κουτί. Αν αυτό είναι 0 τότε του βάζω την τιμή 1 έτσι ώστε να γίνει εμπόδιο ( multi[i][j+1] = 1; ), μετά αυξάνω το μήκος του μονοπατιού κατά 1 ( diafora[d]++; ) και στην συνέχεια αυξάνω το j κατά 1 ώστε να μεταφερθώ σε αυτό το κουτί. Τέλος εκτυπώνω προς τα που κινούμαι καθώς και το μήκος του μονοπατιού.

Επίσης τοποθετώ στην ουρά την θέση του κουτιού που έγινε 1 διασφαλίζοντας πρώτα με ένα if ότι θα μπει στην σωστή ουρά.

**if** (MetritisOuras == 1) {

q.add(i);

q.add(j+1);

}

**else** {

q2.add(i);

q2.add(j+1);

**Οι 4 άλλες συνθήκες του if** διασφαλίζουν ότι δεν είναι το γειτονικό κουτί που θέλω να ψάξω το κουτί εξόδου.

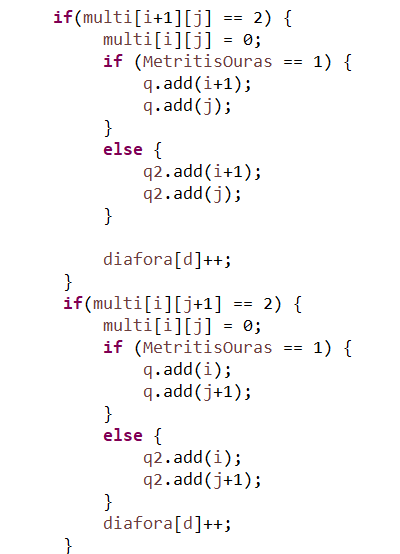
Με τον ίδιο τρόπο λειτουργούν και τα άλλα 3 if αντίστοιχα για τα υπόλοιπα γειτονικά κουτιά. Δηλαδή:

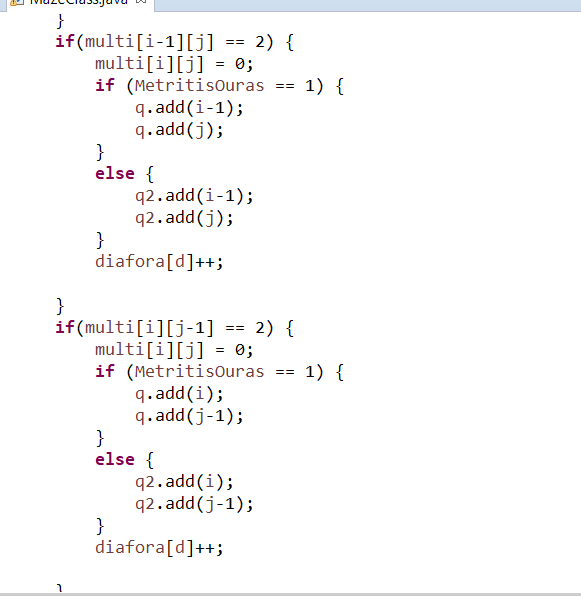
Για αριστερά multi[i][j-1]

Για κάτω multi[i+1][j]

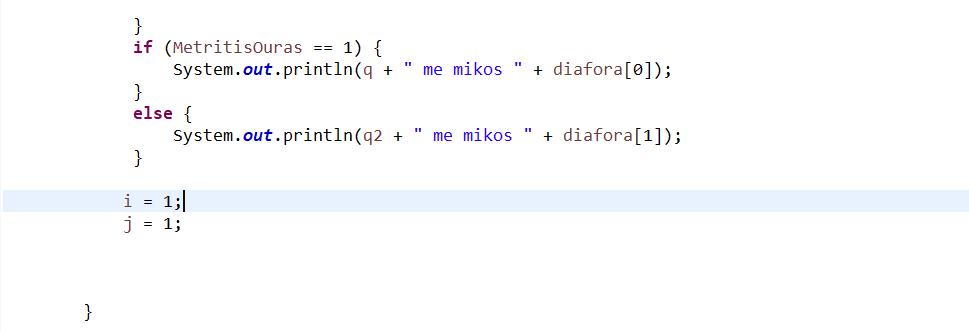
Για πάνω multi[i-1][j]

με τις κατάλληλες αλλαγές βέβαια.

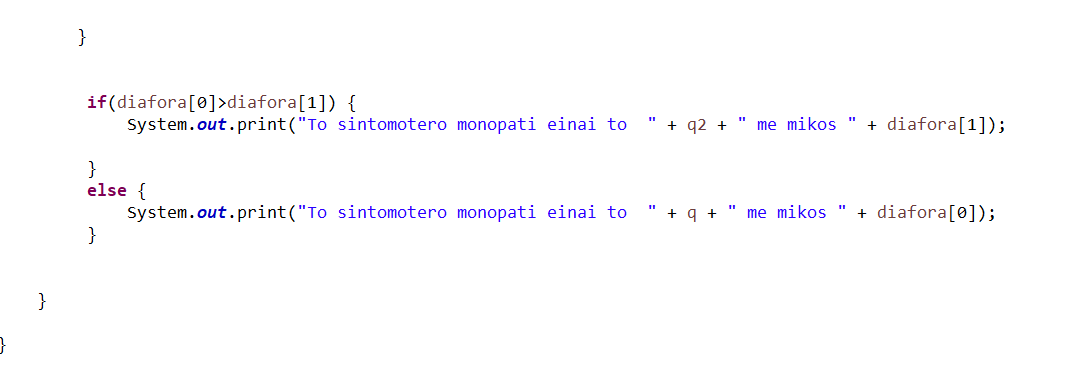




Έτσι τελειώνει η διάσχιση του πίνακα έχοντας τοποθετήσει εμπόδια σε όλα τα κουτιά και έχοντας βάλει αυτές τις θέσεις στις αντίστοιχες ουρές. Όμως δεν έχει μπει ακόμα σε κάθε ουρά το κουτί εξόδου, αφού στο τέλος της επανάληψης βρισκόμαστε ένα βήμα πριν το κουτί εξόδου. Άρα με τα 4 if βρίσκουμε το κουτί εξόδου και το βάζουμε και αυτό σε κάθε ουρά. Επίσης αυξάνουμε το μήκος την μονοπατιών.



Τέλος εκτυπώνουμε το μήκος κάθε μονοπατιού και θέτουμε την τιμή 1 στα i, j έτσι ώστε να ξεκινήσει ξανά η σάρωση του πίνακα για να βρει και το 2ο μονοπάτι.



Αφού τελειώνει λοιπόν ολοκληρωτικά η εύρεση και των 2 μονοπατιών τα μήκη τους είναι αποθηκευμένα στον πίνακα diafora[] οπότε με μια απλή σύγκριση εκτυπώνουμε το συντομότερο μονοπάτι.

Το αποτέλεσμα της κονσόλας είναι:

