Εργαστήριο Επεξεργασίας Πληροφορίας και Υπολογισμών
Τομέας Ηλεκτρονικής και Υπολογιστών **5ο Εξάμηνο**
ΤΗΜΜΥ Α.Π.Θ **2015-2016**

ΔΟΜΕΣ ΔΕΔΟΜΕΝΩΝ

DS – Proximity

Το παιχνίδι

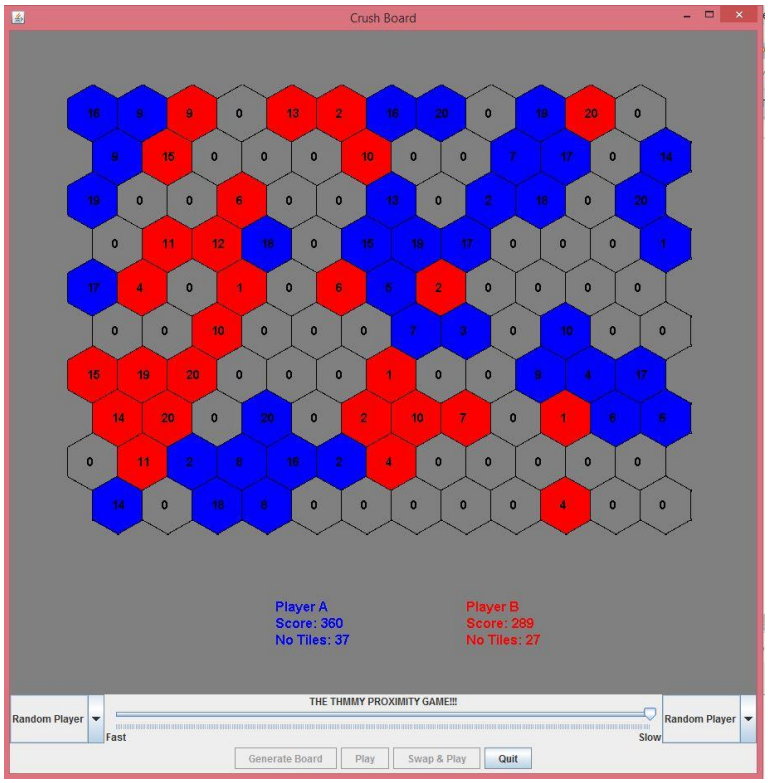
Το Proximity είναι ένα παιχνίδι στρατηγικής, εναλλασσόμενης σειράς (turned-based strategy game). Οι δύο παίκτες (που απεικονίζονται με μπλε και κόκκινο χρώμα), τοποθετούν εναλλάξ πλακίδια σε ένα εξάγωνο ταμπλό 12x10 θέσεων. Κάθε πλακίδιο μπορεί να έχει μια ακέραια τιμή στο πεδίο [1,20], η οποία είναι είτε τυχαία, είτε προκαθορισμένη από τον παίκτη που παίζει.

Κάθε φορά που ένας παίκτης τοποθετεί ένα πλακίδιο στο ταμπλό συμβαίνουν τα εξής:

- Όσα πλακίδια εφάπτονται στο πλακίδιο που μόλις τοποθετήθηκε και έχουν το ίδιο χρώμα (ανήκουν στο παίκτη που παίζει) αυξάνουν το σκορ τους κατά ένα.
- Όσα πλακίδια εφάπτονται στο πλακίδιο που μόλις τοποθετήθηκε και ανήκουν στον αντίπαλο παίκτη αλλάζουν χρώμα, αν το σκορ τους είναι μικρότερο από το νέο πλακίδιο.

Το παιχνίδι τελειώνει μόλις συμπληρωθούν όλες οι θέσεις του ταμπλό. Νικητής ανακηρύσσεται ο παίκτης με το μεγαλύτερο σκορ, το οποίο υπολογίζεται ως το άθροισμα του επιμέρους σκορ των πλακιδίων. Σε περίπτωση ισοπαλίας, νικητής είναι ο παίκτης με τον μεγαλύτερο αριθμό πλακιδίων. Αν οι παίκτες ισοβαθμήσουν και σε αυτό το κριτήριο, τότε το παιχνίδι λήγει ισόπαλο.

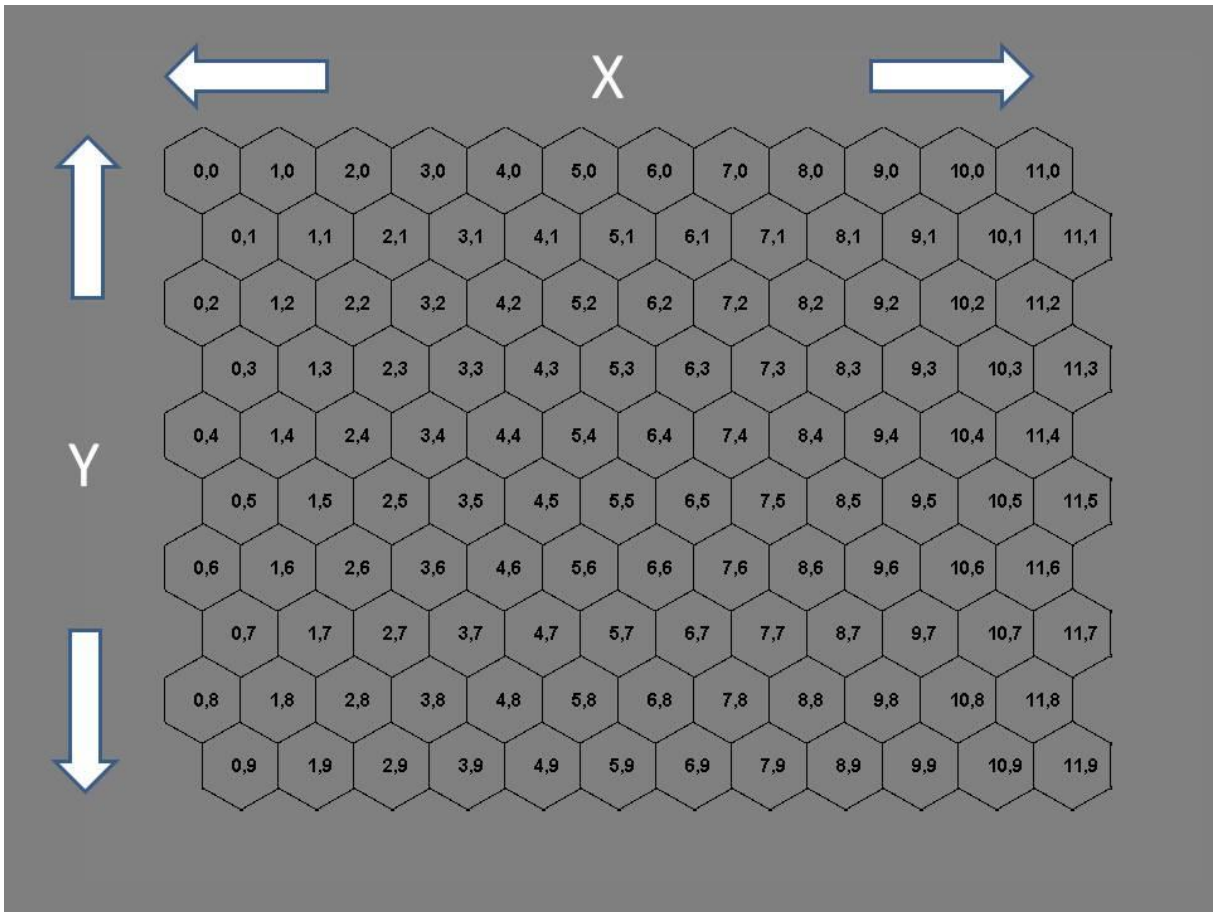
Για όσους ενδιαφέρονται, το παιχνίδι είναι διαθέσιμο σε μορφή web εδώ.



Εικόνα 1: Το περιβάλλον του παιχνιδιού DS-Proximity για το μάθημα των Δομών Δεδομένων 2015-2016

Εξάγωνο Πλέγμα

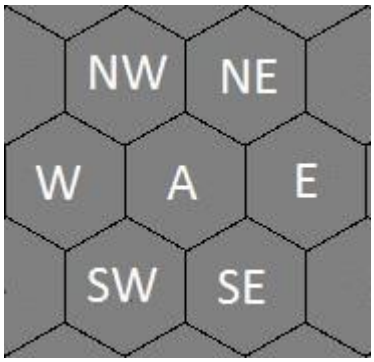
Το παιχνίδι μας παίζεται σε ένα ταμπλό που αποτελείται από 12x10 εξάγωνα κελιά. Η αρίθμηση των κελιών αρχίζει από το κελί πάνω αριστερά, όπως φαίνεται στην **Εικόνα 2**. Η συντεταγμένες των κελιών δίνονται στην μορφή [x,y].



Εικόνα 2: Συντεταγμένες του εξάγωνου πλαισίου του παιχνιδιού

Γειτονικά Κελιά

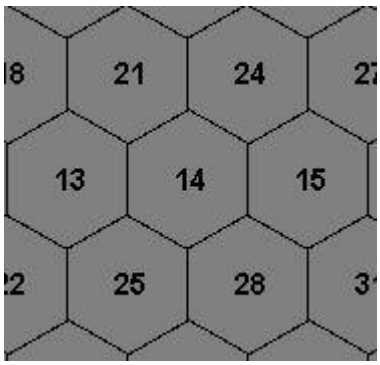
Για κάθε κελί, πλην αυτών που βρίσκονται στην άκρη του ταμπλό, **ορίζονται 6 κελιά ως γειτονικά**. Έτσι λοιπόν, ακολουθώντας ωρολογιακή φόρα, ορίζεται το **ανατολικό** γειτονικό κελί (East), το **νοτιοανατολικό** γειτονικό κελί (SouthEast), το **νοτιοδυτικό** γειτονικό κελί (SouthWest), το **δυτικό** γειτονικό κελί (West), τον το **βορειοδυτικό** γειτονικό κελί (NorthWest) και, τέλος, το **βορειοανατολικό** γειτονικό κελί (NorthEast) (**Εικόνα 3**).



Εικόνα 3: Ονομασία των γειτόνων του κελιού Α

Αναπαράσταση εξάγωνου πλέγματος σε δισδιάστατο πίνακα

Έστω η περίπτωση του εξαγωνικού πλέγματος της **Εικόνα 4**. Υπάρχουν πολλοί τρόποι για να αναπαραστήσουμε το συγκεκριμένο πλέγμα σε έναν δισδιάστατο πίνακα.



Εικόνα 4: Εξάγωνο πλέγμα σε πίνακα

Ο πιο απλός τρόπος είναι να χρησιμοποιήσουμε κενές θέσεις στον πίνακα, όπως φαίνεται **Πίνακα 1**.

	21		24	
13		14		15
	25		28	

Πίνακας 1: Εξάγωνο πλέγμα σε πίνακα με χρήση κενών

Εύκολα γίνεται αντιληπτό πως αυτός ο τρόπος είναι αναποτελεσματικός, καθώς απαιτεί περίπου διπλάσιο χώρο σε σχέση με το μέγεθος του πλέγματος.

Για να εξοικονομήσουμε χώρο μπορούμε **να εξαλείψουμε τα κενά**, όπως φαίνεται στον **Πίνακα 2**. **Σε αυτή την περίπτωση δεν χρησιμοποιούμε περισσότερο χώρο από όσο πραγματικά χρειαζόμαστε, αλλά δεν μπορούμε να διαχωρίσουμε του γείτονες ενός κελιού με εύκολο τρόπο.** Για να μπορέσουμε να εντοπίσουμε με σωστό τρόπο τους γείτονες του κελιού θα πρέπει να γνωρίζουμε αν το στοιχείο που αναφερόμαστε βρίσκεται σε μονή ή σε ζυγή σειρά στον πίνακα που έχουμε ορίσει.

Παραδείγματος χάριν, έστω ότι το στοιχείο 14 της Εικόνας 4 βρίσκεται σε **μόνη σειρά**. Σε αυτή την περίπτωση οι γείτονές του θα είναι αποθηκεμένοι στον πίνακα ως εξής:

...	21 (NW)	24 (NE)
13 (W)	14	15 (E)
...	25 (SW)	28 (SE)

Πίνακας 2: Εξάγωνο πλέγμα σε πίνακα χωρίς κενά, γείτονες κελιού σε μονή σειρά.

Αντίστοιχα, αν το στοιχείο 14 είναι σε **ζυγή σειρά**, τότε οι γειτονές του θα είναι αποθηκεμένοι στον πίνακα ως εξής:

21 (NW)	24 (NE)	
13 (W)	14	15 (E)
25 (SW)	28 (SE)	

Πίνακας 2: Εξάγωνο πλέγμα σε πίνακα χωρίς κενά, γείτονες κελιού σε μονή σειρά.

Για περισσότερη ανάλυση των εξάγωνων πλεγμάτων μπορείτε να δείτε εδώ.

Εργασία Α' – Random Movement (0,75 βαθμοί)

Στην παρούσα εργασία υπάρχουν δύο ζητούμενα.

1. Να υλοποιήσετε μια συνάρτηση που θα επιστρέφει τις συντεταγμένες των γειτονικών πλακιδίων δεδομένου ενός πλακιδίου με συντεταγμένες [x,y].
2. Να υλοποιήσετε όλες τις αναγκαίες συναρτήσεις ώστε να δημιουργηθεί ένας παίκτης που θα μπορεί να επιλέξει τυχαία μια θέση [x,y] πάνω στο ταμπλό. Στην συνέχεια, θα πρέπει να ελέγξει αν αυτή η θέση είναι άδεια, ώστε να μπορεί να τοποθετήσει εκεί το πλακίδιο στην επόμενη κίνησή του. Σε αντίθετη περίπτωση θα πρέπει να διαλέξει μια άλλη τυχαία θέση.

Στη συνέχεια παρουσιάζονται οι κύριες κλάσεις που αποτελούν τον κορμό του παιχνιδιού με τις συναρτήσεις τους (οι κλάσεις Tile και Board), καθώς και η κλάση RandomPlayer που καλείστε να συμπληρώσετε.

Κλάση Tile (package: gr.auth.dsproject.proximity.board)

Η κλάση αυτή δημιουργεί τα πλακίδια του ταμπλό του παιχνιδιού. Έχει τις εξής μεταβλητές:

- **`int id`**: είναι μια μεταβλητή που δείχνει το μοναδικό κωδικό ενός πλακιδίου.
- **`int x`**: η μεταβλητή αυτή δείχνει την θέση του πλακιδίου στον άξονα x'x.
- **`int y`**: η μεταβλητή αυτή δείχνει την θέση του πλακιδίου στον άξονα y'y.
- **`int color`**: η μεταβλητή αυτή δείχνει το χρώμα του που πλακιδίου. Συγκεκριμένα έχουμε τα εξής χρώματα:
  - **`0`** → **Gray** (Υποδηλώνει κενή θέση)
  - **`1`** → **Blue** (Υποδηλώνει θέση όπου έχει τοποθετήσει πλακίδιο ο μπλε παίκτης)
  - **`2`** → **Red** (Υποδηλώνει θέση όπου έχει τοποθετήσει πλακίδιο ο κόκκινος παίκτης)
- **`int PlayerId`**: η μεταβλητή αυτή δείχνει αν στην θέση του συγκεκριμένου πλακιδίου έχει παίξει κάποιος από τους 2 παίκτες και αν ναι, ορίζει τον παίκτη στον οποίο ανήκει το συγκεκριμένο πλακίδιο. Συγκεκριμένα έχουμε:
  - **`0`** → **Κενό πλακίδιο**
  - **`1`** → **Πλακίδιο του παίκτη Α**
  - **`2`** → **Πλακίδιο του παίκτη Β**

ΠΡΟΣΟΧΗ!!! **Αν θέλετε να διαβάσετε τις τιμές των μεταβλητών ενός αντικειμένου τύπου Tile θα πρέπει να χρησιμοποιήσετε τους αντίστοιχους getters.**

Κλάση Board (package:gr.auth.dsproject.proximity.board)

Η κλάση αυτή δημιουργεί και αποτυπώνει στην οθόνη το αντικείμενο τύπου Board. Οι συναρτήσεις που σας ενδιαφέρουν και θα σας βοηθήσουν στην υλοποίηση των εργασιών σας είναι οι εξής:

`Tile getTile(int x, int y)`: Επιστρέφει το αντικείμενο τύπου Tile που βρίσκεται στην θέση [x,y].

`Tile[] getNeighbors (int x, int y)`: Η συνάρτηση αυτή παίρνει σαν όρισμα 2 ακεραίους (x,y) που αναπαριστούν τις συντεταγμένες ενός πλακιδίου. Με βάση τις συντεταγμένες αυτές επιστρέφει τα γειτονικά Tiles στην μορφή ενός πίνακα μεγέθους 6x1, όπου σε κάθε θέση θα περιέχει το αντικείμενο τύπου Tile που αναπαριστά τον αντίστοιχο γείτονα. ***Η συγκεκριμένη συνάρτηση θα λειτουργεί σωστά μόνο αφού υλοποιήσετε σωστά την συνάρτηση `getNeighborsCoordinates` της κλασης RandomPlayer (βλέπετε παρακάτω).***

`boolean isInsideBoard(int x, int y)`: Επιστρέφει true αν η θέση [x,y] είναι εντός των ορίων του ταμπλό. Σε διαφορετική περίπτωση επιστρέφει false.

Κλάση RandomPlayer (package:gr.auth.dsproject.proximity.defplayers)

Η κλάση αυτή **(την οποία καλείστε να συμπληρώσετε)** είναι υπεύθυνη για την δημιουργία και την ομαλή λειτουργία των παικτών της πλατφόρμας. Θα πρέπει να έχει τρεις μεταβλητές:

1. **`int id`**: είναι μια μεταβλητή που παίρνει την τιμή 1 ή 2 ανάλογα με το αν ο παίκτης είναι ο μπλε ή ο κόκκινος. Ο παίκτης σας αλλάζει από παιχνίδι σε παιχνίδι. Για τον λόγο αυτό, αν θέλετε να ελέγξετε αν ένα πλακίδιο είναι δικό σας θα πρέπει να συγκρίνεται την τιμή id του παίκτης σας με την τιμή της μεταβλητής playerId του πλακιδίου.

2. **`String name`**: η μεταβλητή αυτή δίνει το όνομα που επιθυμούμε στον παίκτη.

3. **`int score`**: η μεταβλητή αυτή αποθηκεύει το άθροισμα των τιμών των πλακιδίων του παίκτη.

4. **`int numOfTiles`**: η μεταβλητή αυτή αποθηκεύει τον αριθμό πλακιδίων του παίκτη που βρίσκονται στο ταμπλό κάθε δεδομένη στιγμή.

Οι συναρτήσεις που πρέπει να υλοποιήσετε είναι οι εξής:

a. **Constructor της κλάσης**: ο constructor θα πρέπει να παίρνει ένα όρισμα (`Integer pid`), το οποίο αντιστοιχεί στην μεταβλητή id του παίκτη. Προσοχή!!!! **Το όρισμα μέσα στην παρένθεση είναι τύπου *Integer* και όχι *int*, σε αυτή τη συγκεκριμένη συνάρτηση μόνο.**

b. **Constructor της κλάσης (2):** αποτελεί τον δεύτερο constructor της κλάσης και παίρνει 4 ορίσματα τα οποία θα αρχικοποιούν όλες τις μεταβλητές της κλάσης. Θα πρέπει να ορίσετε σωστά τον τύπο των ορισμάτων και να τα αντιστοιχίσετε στις κατάλληλες μεταβλητές της κλάσης.

   Προσοχή!!!! Τα ορίσματα – μεταβλητές που θα έχει ο constructor θα πρέπει να είναι ορισμένα με την σειρά που δίνονται παραπάνω. **Επίσης και εδώ το όρισμα μέσα στη παρένθεση θα πρέπει να είναι τύπου *Integer* και όχι int**.

c. Όλες οι συναρτήσεις ***get*** και ***set*** για τις μεταβλητές της κλάσης.

***d*. Συνάρτηση `int[] getNextMove (Board board)`:** Η συγκεκριμένη συνάρτηση παίρνει σαν όρισμα την μεταβλητή **`board`** η οποία είναι αντικείμενο της κλάσης **`Board`** και επιστρέφει έναν **μονοδιάστατο πίνακα ακεραίων, μεγέθους δύο (2)** ο οποίος θα περιέχει την θέση `[x,y]` στην οποία θα τοποθετηθεί το επόμενο πλακίδιο. Η θέση (x,y) επιλέγετε με **τυχαίο** τρόπο και να είναι εντός των ορίων του ταμπλό. Θα πρέπει επίσης να ελέγξετε αν η θέση `[x,y]` είναι ήδη κατειλημμένη και αν ναι, θα πρέπει να επιλέξετε μια νέα θέση `[x1,y1]`, με επίσης τυχαίο τρόπο.

e*. static int[][] getNeighborsCoordinates(Board board,int x, int y)**: Η συνάρτηση αυτή παίρνει σαν όρισμα την μεταβλητή board και 2 ακεραίους (x,y) που αναπαριστούν τις συντεταγμένες ενός πλακιδίου πάνω στο ταμπλό. Με βάση τις συντεταγμένες αυτές θα πρέπει να υπολογίζει τις συντεταγμένες των γειτονικών πλακιδίων (σύμφωνα με την ανάλυση που παρουσιάστηκε στην αντίστοιχη παράγραφο) και να τις επιστρέφει στην μορφή ενός πίνακα μεγέθους 6x2. Η πρώτη γραμμή του πίνακα που θα επιστρέφεται θα περιέχει τις συντεταγμένες [x0.y0] του ανατολικού γείτονα, η δεύτερη γραμμή τις συντεταγμένες του νότιο-ανατολικού γείτονα και ούτω καθ' εξής, μέχρι την 6η σειρά που θα περιέχει τις συντεταγμένες του βόρειο-ανατολικού γείτονα. Σε πρίπτωση που κάποιο κελί δεν έχει όλους τους γείτονες γιατί είναι στην άκρη του ταμπλό, ορίστε ως συντεταγμένες τις [-1,-1]. ***Η συγκεκριμένη συνάρτηση χρησιμοποιείται για να λειτουργεί σωστά η συνάρτηση **`getNeighbors`** ***της κλάσης Board (βλέπε παραπάνω)*.**

Σημείωση: Οι συναρτήσεις της κλάσης **RandomPlayer** που θα δημιουργήσετε, θα πρέπει να είναι ορατές από τον υπόλοιπο κώδικα του παιχνιδιού ο οποίος βρίσκετε σε **διαφορετικό πακέτο**. Θα πρέπει λοιπόν να χρησιμοποιήσετε τους **κατάλληλους τροποποιητές**. Επίσης, θα πρέπει τα ονόματα των συναρτήσεων και των μεταβλητών να είναι **ακριβώς** ίδια με τα ονόματα που αναφέρονται στην περιγραφή τους.

Βοηθητικές Κλάσεις - Συναρτήσεις

Για την υλοποίηση των παραπάνω συναρτήσεων χρειάζεται να χρησιμοποιήσετε κάποιες μεταβλητές / συναρτήσεις που υπάρχουν ήδη υλοποιημένες στην πλατφόρμα.

- **Στατικές Μεταβλητές της κλάσης *ProximityUtilities***

  Αριθμός Γραμμών και Στηλών:
  NUMBER_OF_ROWS = 10; NUMBER_OF_COLUMNS = 12;
  Οι συγκεκριμένες μεταβλητές μπορούν να χρησιμοποιηθούν για την σάρωση του ταμπλό. Αν **δεν** χρησιμοποιήσετε τις μεταβλητές αλλά γράψετε στον κώδικα απευθείας τον αριθμό 12 ή 10, ίσως αντιμετωπίσετε πρόβλημα στο μέλλον –π.χ. αν αποφασίσουμε να μεγαλώσουμε το ταμπλό, θα πρέπει να κάνετε αλλαγές στον κώδικά σας.

  Για να καλέσετε αυτές τις μεταβλητές αρκεί να γράψετε στον κώδικά σας ProximityUtilities.NUMBER_OF_ROWS ή ProximityUtilities.NUMBER_OF_COLUMNS αντίστοιχα.

- **double Math.random():** Η συνάρτηση αυτή σας επιστρέφει έναν τυχαίο αριθμό τύπου **double** στο διάστημα [0,1). Θα σας χρειαστεί ώστε να επιλέξετε την επόμενή σας κίνηση με τυχαίο τρόπο.

Εγκατάσταση

Για να εγκαταστήσετε το project στον Eclipse, αρκεί να κάνετε unzip το αρχείο Proximity Part A.zip μέσα στον φάκελο workspace που έχετε ορίσει, και στη συνέχεια από το περιβάλλον του Eclipse να κάνετε:

`File → Import → General → Existing Projects into Workspace`

Για να τρέξετε το project κάνετε δεξί κλικ πάνω στο project και στη συνέχεια επιλέγετε:

`Run As → Java Application`

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.

Είναι δική σας ευθύνη η απόδειξη καλής λειτουργίας του προγράμματος.

Παραδοτέα για κάθε μέρος της εργασίας

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_PartA.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 30 Νοεμβρίου, 23:59** (ηλεκτρονικά)
Δε θα υπάρξει καμία παρέκκλιση από την παραπάνω προθεσμία.