Τρίτη Σειρά ασκήσεων
Ημερομηνία Παράδοσης: 1 Ιουνίου 2015, 11:59 μ.μ.

Η τρίτη σειρά ασκήσεων περιέχει δύο τύπους ασκήσεων. Τις βασικές ασκήσεις και μία bonus άσκηση. Οι βασικές ασκήσεις πρέπει να παραδοθούν μέχρι τις 1/6 11:59 μ.μ. Η bonus άσκηση έχει προθεσμία στις 7/6 11:59 μ.μ. Θα χρησιμοποιήσετε διαφορετικό turnin για τις διαφορετικές ασκήσεις.

Στην υλοποίηση των κλάσεων σας θα πρέπει να χρησιμοποιείτε μόνο private πεδία. Βαθμοί θα αφαιρεθούν για προγράμματα που δεν είναι καλά γραμμένα, δηλαδή δεν είναι σωστά στοιχισμένα ή δεν έχουν καλά επιλεγμένα ονόματα μεταβλητών ώστε να διαβάζονται εύκολα.

Άσκηση 1

Στην άσκηση αυτή θα κάνετε χρήση της κληρονομικότητας. Θα υλοποιήσετε ένα αγώνα μεταξύ οχημάτων διαφορετικού τύπου σε μία πίστα που αποτελείται από κομμάτια με οδόστρωμα διαφορετικού τύπου. Έχουμε τρεις τύπους οχημάτων: μηχανή (motorcycle), αυτοκίνητο (car) και 4x4 τζιπ (jeep). Επίσης έχουμε τρεις τύπους οδοστρώματος: κανονικό δρόμο με άσφαλτο (road), δρόμο με χώμα (dirt road) και ανώμαλο δρόμο με πέτρες (rocky road). Η πίστα αποτελείται από Ν τμήματα διαφορετικού μήκους από τους παραπάνω τύπους. Όλα τα οχήματα ξεκινάνε με την ίδια ποσότητα καυσίμων και στο τέλος κάθε τμήματος τους δίνεται επιπλέον καύσιμα. Το κάθε όχημα έχει διαφορετικό συντελεστή κατανάλωσης καυσίμων ο οποίος εξαρτάται από το όχημα και τον τύπο του δρόμου στον οποίο κινείται και τα επιπλέον χαρακτηριστικά του δρόμου. Η ταχύτητα του οχήματος εξαρτάται από την ποσότητα καυσίμου που θα αποφασίσει να χρησιμοποιήσει και τον συντελεστή κατανάλωσης του οχήματος για το κομμάτι του δρόμου. Στο τέλος κερδίζει το όχημα που έκανε τον μικρότερο χρόνο συνολικά.

Για τους διαφορετικούς τύπους οδοστρώματος:
Καταρχάς ορίστε μια κλάση **Road** η οποία υλοποιεί τον απλό δρόμο με άσφαλτο. Έχει πεδία τον αριθμό των χιλιομέτρων και μια Boolean μεταβλητή hasTurns η οποία μας λέει αν ο δρόμος έχει πολλές στροφές ή όχι. Ο constructor που παίρνει σαν όρισμα τον αριθμό των χιλιομέτρων και θέτει τυχαία την τιμή της μεταβλητής hasTurns σε true ή false. Ορίστε μια μέθοδο **toString** που δίνει όλες τις πληροφορίες για τον δρόμο, και όποιες μεθόδους πρόσβασης και μετάλλαξης χρειάζεστε. Επίσης ορίστε την μέθοδο **updateFactor** η οποία παίρνει σαν όρισμα τον συντελεστή κατανάλωσης ενός οχήματος και αν ο δρόμος έχει στροφές τον αυξάνει κατά 10%. Τέλος, ορίστε μια μέθοδο **getType** που επιστρέφει ένα ακέραιο που σηματοδοτεί τον τύπο του οδοστρώματος (0 για απλό δρόμο).

Ορίστε και τις κλάσεις **DirtRoad** και **RockyRoad** οι οποίες κληρονομούν από την Road.

Η **DirtRoad** έχει ένα επιπλέον double πεδίο που κρατάει μια τιμή μεταξύ 0 και 1 για το επίπεδο βροχής στον δρόμο. Η τιμή αυτή αρχικοποιείται τυχαία στον constructor. Η DirtRoad υπερβαίνει την μέθοδο **updateFactor** ώστε αν το επίπεδο της βροχής είναι μεταξύ 0.8 και 1, αυξάνει τον συντελεστή κατανάλωσης κατά ένα επιπλέον 20%. Αν το επίπεδο βροχής είναι μεταξύ 0.5 και 0.8 τον αυξάνει κατά ένα επιπλέον 10%. Αν είναι κάτω από 0.5 δεν έχει επίδραση. Προσέξτε ότι η αύξηση του συντελεστή είναι πάνω στην αύξηση που μπορεί να έχει κάνει ήδη η γονική κλάση. Υπερβείτε επίσης και την μέθοδο **toString** για να επιστρέφει και τις επιπλέον πληροφορίες για τον δρόμο. Τέλος υπερβείτε την μέθοδο **getType** ώστε να επιστρέφει την τιμή 1 για χωματόδρομο.

Η **RockyRoad** έχει ένα επιπλέον ακέραιο πεδίο το οποία παίρνει τιμές μεταξύ 0 και 2 που μετράει το πόσο ανώμαλο είναι το έδαφος. Η τιμή του αρχικοποιείται τυχαία στον constructor. Η RockyRoad υπερβαίνει την μέθοδο **updateFactor** ώστε αν η δυσκολία του δρόμου είναι στο επίπεδο 2, αυξάνει τον συντελεστή κατανάλωσης κατά ένα επιπλέον 20%. Αν είναι 1, τον αυξάνει κατά ένα επιπλέον 10%. Αν είναι 0 δεν έχει επίδραση. Προσέξτε ότι η αύξηση του συντελεστή είναι πάνω στην αύξηση που μπορεί να έχει κάνει ήδη η γονική κλάση. Υπερβείτε επίσης και την μέθοδο **toString** για να επιστρέφει και τις επιπλέον πληροφορίες για τον δρόμο. Τέλος υπερβείτε την μέθοδο **getType** ώστε να επιστρέφει την τιμή 2 ανώμαλο δρόμο.

Υπόδειξη: Μπορεί να σας βολέψει να ορίσετε στην Road ένα πίνακα που να μεταφράζει τον τύπο του δρόμου σε ένα String (π.χ. το 0 σε paved road).

Για τα οχήματα:
Ορίστε μια **αφηρημένη** κλάση **Vehicle** που κρατάει πληροφορίες για ένα όχημα, και τις παράγωγες **ενυπόστατες** κλάσεις **Motorcycle**, **Car** και **Jeep**. Η Vehicle έχει την **αφηρημένη** μέθοδο **computeFactor** η οποία παίρνει σαν όρισμα ένα αντικείμενο Road και υπολογίζει τον τελικό συντελεστή κατανάλωσης. Η μέθοδος υλοποιείται στις ενυπόστατες κλάσεις. Ο συντελεστής κατανάλωσης είναι η ποσότητα καυσίμων που καίει το αυτοκίνητο ανά χιλιόμετρο ώστε να έχει ταχύτητα 1 χλμ/λεπτό. Υποθέστε τις παρακάτω τιμές.

- Η μηχανή έχει συντελεστές 0.5, 2 και 2.5 για απλό δρόμο, δρόμο με χώμα και ανώμαλο δρόμο.
- Το αυτοκίνητο έχει συντελεστές 1, 1.5 και 2 για απλό δρόμο, δρόμο με χώμα και ανώμαλο δρόμο.
- Το τζιπ έχει συντελεστές 1.3, 1.5 και 1.8 για απλό δρόμο, δρόμο με χώμα και ανώμαλο δρόμο.

Η μέθοδος computeFactor υπολογίζει τον συντελεστή κατανάλωσης και τον επιστρέφει, αφού πρώτα καλέσει την updateFactor του αντικειμένου Road που δίνεται ως παράμετρος.

Η κλάση **Vehicle** έχει πεδία για το όνομα του οχήματος, την ποσότητα καυσίμων που έχει το όχημα, και τον συνολικό χρόνο που έχει κάνει μέχρι στιγμής στον αγώνα. Ορίστε την μέθοδο **race** η οποία παίρνει σαν όρισμα ένα αντικείμενο Road (κομμάτι της πίστας) και μια (αποδεκτή) ποσότητα καυσίμων που θα καταναλώσει το όχημα για αυτό το κομμάτι και υπολογίζει τον χρόνο που θα κάνει το όχημα για αυτό το κομμάτι. (Υπόδειξη: Για να υπολογίσουμε την ταχύτητα του οχήματος υπολογίζουμε την πραγματική κατανάλωση ανά χιλιόμετρο του οχήματος και διαιρούμε με τον συντελεστή κατανάλωσης). Η μέθοδος ενημερώνει τα πεδία κατάλληλα. Υπερφορτώστε την μέθοδο race ώστε να παίρνει σαν όρισμα μόνο το αντικείμενο Road και να χρησιμοποιεί όλα τα διαθέσιμα καύσιμα. Ορίστε επίσης την μέθοδο **refuel** η οποία αυξάνει τα καύσιμα κατά 100 μονάδες, την μέθοδο **toString** που επιστρέφει τις βασικές πληροφορίες για το όχημα, και όποια άλλη μέθοδο χρειάζεστε.

Υπόδειξη: Τους συντελεστές κατανάλωσης μπορεί να σας βολεύει να τους κρατήσετε σε ένα πίνακα σε κάθε κλάση.

Για την πίστα:
Ορίστε μια κλάση **RaceTrack** η οποία κρατάει πληροφορία για την πίστα συνολικά. Έχει ένα πίνακα με τα διάφορα κομμάτια της πίστας, το μέγεθος του οποίου δίνεται σαν όρισμα στον constructor. Ο constructor επίσης αναθέτει τυχαία (με ίση πιθανότητα) ένα από τους τρείς τύπους οδοστρώματος σε κάθε κομμάτι, και μια τυχαία τιμή μεταξύ 50 και 100 για το μήκος του κάθε κομματιού της διαδρομής. Ορίστε μια μέθοδο **nextSegment** που σας δίνει το επόμενο κομμάτι της πίστας το οποίο δεν έχουν τρέξει ακόμη, και μια μέθοδο **reachedEnd** η οποία ελέγχει αν φτάσαμε στο τέλος. Τέλος ορίστε και μία μέθοδο **print** η οποία τυπώνει το υπόλοιπο της πίστας που έχουν ακόμη να τρέξουν οι διαγωνιζόμενοι.

Για τους συμμετέχοντες:
Ορίστε μια κλάση **RaceParticipants** η οποία θα κρατάει πληροφορία για τους συμμετέχοντες στον αγώνα. Η κλάση θα έχει ως πεδίο ένα πίνακα με τρία οχήματα που ελέγχονται από τον υπολογιστή, και ένα όχημα που ελέγχει ο χρήστης. Στον constructor δημιουργούμε τα τρία οχήματα του υπολογιστή ένα για καθένα από τους διαφορετικούς τύπους οχημάτων, ενώ ζητάμε από τον χρήστη να επιλέξει ποιον τύπο οχήματος θέλει να χρησιμοποιήσει. Η κλάση χειρίζεται τα πάντα που έχουν σχέση με τα οχήματα: να τα βάλει να τρέξουν ένα κομμάτι της πίστας, να τυπώσει την κατάσταση των οχημάτων, να τα γεμίσει με καύσιμα, και να αναδείξει τον τελικό νικητή. Όταν τρέχουν τα οχήματα του υπολογιστή χρησιμοποιούν πάντα όλα τα διαθέσιμα καύσιμα. Για τον παίχτη του χρήστη τον ρωτάμε πόσες μονάδες καυσίμων από τις διαθέσιμες θέλει να καταναλώσει.

Για τον αγώνα:
Ορίστε την κλάση **RacingGame** η οποία έχει την **main** η οποία υλοποιεί τον αγώνα (τεστάρετε το με μια πίστα με 10 κομμάτια). Το πρόγραμμα όσο δεν έχουμε φτάσει στο τέλος της πίστας κάνει τα εξής βήματα: (1) τυπώνει το υπόλοιπο της πίστας; (2) γεμίζει τα αυτοκίνητα με καύσιμα; (3) βάζει τα οχήματα να τρέξουν το επόμενο κομμάτι; (4) τυπώνει την κατάσταση των οχημάτων. Όταν τελειώσει ο αγώνας αναδεικνύει τον νικητή.

Παραδώστε τον κώδικα για όλες τις κλάσεις που θα δημιουργήσετε.

Άσκηση 2

Στην άσκηση αυτή θα εξασκηθείτε με την χρήση των δομών. Έχουμε ένα κοινωνικό δίκτυο όπως στην προηγούμενη σειρά με τις φιλίες μεταξύ των χρηστών. Σε αυτό το κοινωνικό δίκτυο θέλουμε να βρούμε το συντομότερο μονοπάτι μεταξύ δύο χρηστών. Ένα μονοπάτι μεταξύ δύο χρηστών Α (αρχικός) και Τ (τελικός) είναι μια ακολουθία από χρήστες $X_1, X_2, .., X_K$, όπου X_1 = Α και X_K = Τ, και για κάθε ζευγάρι (X_i, X_{i+1}) οι χρήστες (X_i, X_{i+1}) είναι φίλοι. Το μήκος του μονοπατιού είναι Κ-1 (ο αριθμός τω σχέσεων που πρέπει να διατρέξουμε για να φτάσουμε από το Α στο Τ). Σε ένα κοινωνικό δίκτυο μπορεί να υπάρχουν πολλαπλά μονοπάτια μεταξύ του Α και του Τ. Θέλουμε να βρούμε ένα με το μικρότερο δυνατό μήκος. Επίσης αν δεν υπάρχει κανένα μονοπάτι μεταξύ του Α και Τα θέλουμε να το αναφέρουμε.

Για να βρούμε το συντομότερο μονοπάτι ξεκινάμε από τον χρήστη Α και επισκεπτόμαστε τους χρήστες με αύξουσα απόσταση από τον Α: πρώτα τους φίλους του Α (χρήστες σε απόσταση 1), μετά τους φίλους των φίλων του Α (χρήστες σε απόσταση 2), κ.ο.κ. μέχρι είτε να βρούμε τον Τ, είτε να εξαντλήσουμε όλους τους χρήστες που μπορούμε να επισκεφτούμε ξεκινώντας από τον Α. Ο αλγόριθμος κρατάει ένα σύνολο με τους χρήστες που έχει ήδη επισκεφτεί, και μια ουρά με τους φίλους των χρηστών που έχει επισκεφτεί τους οποίους θα επισκεφτεί στο μέλλον. Αρχικά η ουρά περιέχει μόνο τον Α. Ο αλγόριθμός προχωράει επαναληπτικά. Σε κάθε επανάληψη αφαιρεί τον πρώτο χρήστη Χ από την κορυφή της ουράς και τον επισκέπτεται: (1) τοποθετεί τον Χ στο σύνολο των χρηστών που έχει επισκεφτεί, (2) διατρέχει τους φίλους του Χ και όσους δεν τους έχει επισκεφτεί τους τοποθετεί στο τέλος της ουράς. Οι επαναλήψεις σταματάνε όταν δεν υπάρχει άλλος χρήστης στην ουρά (άρα δεν βρήκαμε τον Τ), ή όταν βρεθεί ο Τ. Για κάθε χρήστη Χ (εκτός του Α), τον προσθέτουμε στην ουρά ως αποτέλεσμα της επίσκεψης σε κάποιον προηγούμενο χρήστη Υ. Λέμε ότι ο Υ είναι ο γονέας του Χ. Για κάθε χρήστη κρατάμε την πληροφορία για τον γονέα του. Έτσι όταν βρούμε τον κόμβο Τ, μπορούμε να κατασκευάσουμε το μονοπάτι μεταξύ Α και Τ ακολουθώντας προς τα πίσω τις σχέσεις γονέα-παιδιού.

Η υλοποίηση σας θα πρέπει να έχει τις κλάσεις **User**, **Node**, **SocialNetwork** και **ShortestPaths**.

Η κλάση **User** θα κρατάει πληροφορία για ένα χρήστη του κοινωνικού δικτύου. Θα έχει το όνομα του (το οποίο είναι μοναδικό για κάθε χρήστη) την ηλικία και την τοποθεσία του χρήστη. Επίσης θα κρατάει και ένα ArrayList με τους φίλους του χρήστη. Ο constructor παίρνει σαν ορίσματα το όνομα ηλικία και τοποθεσία. Προσθέσετε την μέθοδο **toString** που επιστρέφει ένα String με τις βασικές πληροφορίες (όνομα, ηλικία, τοποθεσία), και όποια άλλη μέθοδο χρειάζεστε.

Η κλάση **Node** η οποία μας βοηθάει να κατασκευάσουμε το μονοπάτι και κρατάει πληροφορία για ένα κόμβο του μονοπατιού. Η κλάση έχει ένα πεδίο τύπου User που αναφέρεται στον χρήστη, και ένα πεδίο τύπου Node που κρατάει πληροφορία για τον πατέρα του κόμβου στο μονοπάτι. Ο constructor αρχικοποιεί και τα δύο πεδία. Ορίστε και τις μεθόδους πρόσβασης.

Η κλάση **SocialNetwork** θα κάνει την περισσότερη δουλειά. Η κλάση κρατάει ένα πεδίο HashMap το οποίο αντιστοιχεί το όνομα του χρήστη με το αντίστοιχο User αντικείμενο. Η κλάση έχει τις εξής μεθόδους:

1. Η μέθοδος **readUsers** παίρνει σαν όρισμα ένα όνομα αρχείου και από αυτό διαβάζει και αποθηκεύει (στο HashMap) τις πληροφορίες για τους χρήστες. Κάθε γραμμή του αρχείου περιέχει τις πληροφορίες για ένα χρήστη (το όνομα, την ηλικία, την τοποθεσία) χωρισμένα με κενό.
2. Η μέθοδος **readNetwork** παίρνει σαν όρισμα ένα όνομα αρχείου και από αυτό διαβάζει και αποθηκεύει τις σχέσεις φιλίας μεταξύ των χρηστών. Κάθε γραμμή του αρχείου περιέχει ένα ζευγάρι με ονόματα χωρισμένα με κενό.
3. Η μέθοδος **pathToString** είναι μια private μέθοδος που παίρνει σαν όρισμα ένα αντικείμενο τύπου Node το οποίο αντιστοιχεί στον τελευταίο χρήστη στο μονοπάτι. Χρησιμοποιώντας την πληροφορία για τον γονέα, ακολουθεί το μονοπάτι προς τα πίσω μέχρι την αρχή, και δημιουργεί ένα String με την πληροφορία του κάθε κόμβου στο μονοπάτι (καλώντας την toString για κάθε χρήστη), από την αρχή προς το τέλος. Επιστρέφει το String με το μονοπάτι. (Σημείωση: Η υλοποίηση της pathToString μπορεί να γίνει και με χρήση αναδρομής).
4. Η μέθοδος **computeShortestPath** παίρνει σαν όρισμα δύο ονόματα και υπολογίζει το συντομότερο μονοπάτι μεταξύ τους (αν υπάρχει). Υλοποιεί την βασική λειτουργία του αλγορίθμου που περιγράψαμε παραπάνω. Χρησιμοποιείστε το HashSet για να κρατάτε το σύνολο των χρηστών που έχετε επισκεφτεί, και

την βιβλιοθήκη LinkedList για να κρατάτε την ουρά με τους κόμβους (Node αντικείμενα) για να επισκεφτείτε στο μέλλον. Η LinkedList έχει τις μεθόδους addLast και removeFirst για να προσθέτετε και να αφαιρείτε στοιχεία. Μπορείτε να διαβάσετε περισσότερα για την LinkedList online στις σελίδες της Oracle. Αν θέλετε μπορείτε να χρησιμοποιήσετε την δικιά σας υλοποίηση της Queue από την δεύτερη σειρά ασκήσεων (τροποποιημένη ώστε να αποθηκεύει αντικείμενα Node). Η μέθοδος θα πρέπει να εξετάζει και οριακές καταστάσεις όπου κάποιο από τα δύο ονόματα δεν ανήκει στο κοινωνικό δίκτυο, ή και τα δύο ονόματα είναι τα ίδια. Θα τυπώνει το μονοπάτι εφόσον υπάρχει το μονοπάτι, ή θα τυπώνει ένα μήνυμα ότι δεν υπάρχει το μονοπάτι.

Η κλάση **ShortestPaths** περιέχει την **main**. Παίρνει ορίσματα εκτέλεσης (command line arguments) τα ονόματα των δύο αρχείων για την κατασκευή του δικτύου. Κατασκευάζει το δίκτυο και μέσα σε ένα βρόγχο, όσο ο χρήστης θέλει να συνεχίσει, ζητάει από τον χρήστη δύο ονόματα για να υπολογίσει το συντομότερο μονοπάτι τους και τυπώνει την απάντηση.

Παραδώστε τον κώδικα για όλες τις κλάσεις που θα δημιουργήσετε.

Bonus Άσκηση

Στις προηγούμενες σειρές ασκήσεων δώσατε δύο διαφορετικές υλοποιήσεις για τον Αφηρημένο Τύπο Δεδομένων Ουρά (Queue), που αποθήκευαν συγκεκριμένους τύπους δεδομένων. Σε αυτή την άσκηση θα υλοποιήσετε μια γενικευμένη κλάση Queue. Η άσκηση έχει δύο κομμάτια.

1. Τροποποιήσετε την υλοποίηση σας είτε από την πρώτη σειρά ασκήσεων είτε από την δεύτερη, ώστε να ορίσετε μία γενικευμένη κλάση η οποία παίρνει σαν παράμετρο τον τύπο των δεδομένων T που αποθηκεύει η ουρά.
2. Ορίστε ένα interface **Prioritazable** το οποίο έχει την μέθοδο **hasPriority** η οποία δεν παίρνει ορίσματα και επιστρέφει μία boolean τιμή. Ορίστε μια γενικευμένη κλάση **PoliteQueue** η οποία παίρνει σαν παράμετρο ένα τύπο δεδομένων που υλοποιεί το interface Prioritazable. Η διαφορά της PoliteQueue με την Queue είναι ότι στην insert η PoliteQueue ελέγχει αν το στοιχείο που θέλουμε να προσθέσουμε έχει προτεραιότητα και αν ναι, το τοποθετεί στην κορυφή της ουράς. Ορίστε μια κλάση **Person** που υλοποιεί το interface Prioritizable. Η Person έχει πεδία για το όνομα και την ηλικία και η hasPriority επιστρέφει true αν η ηλικία είναι πάνω από 65. Δημιουργείστε μια **PoliteQueueTest** με την οποία θα προσθέτετε και θα αφαιρείτε αντικείμενα Person σε μία PoliteQueue για να δείξετε ότι η κλάση σας δουλεύει σωστά.