Αναφορά παράδοσης

Παπαθανασίου Αθανασία Μαρία : 3180147

Ερώτημα Α

Η ουρά και η στοίβα του ερωτήματος Α κατασκευάζονται με τη χρήση συνδεδεμένης λίστας μονής σύνδεσης. Τη λίστα μονής σύνδεσης αντιπροσωπεύει η κλάση List<T>. Για τη δημιουργία των κόμβων της λίστας χρησιμοποιούμε την κλάση ListNode<T> .Η κλάση ListNode<T> έχει δύο μεταβλητές στιγμιοτύπου : την data τύπου T , όπου περιέχει τα δεδομένα ενός κόμβου και την nextNode τύπου ListNode<T> που αποτελεί μια αναφορά στον επόμενο κόμβο. Η κλάση ListNode<T> έχει δύο κατασκευαστές : έναν που δημιουργεί ένα ListNode που αναφέρεται σε ένα αντικείμενο και έναν που αρχικοποιεί τις μεταβλητές στιγμιοτύπου. Οι μέθοδοι getData και getNextNode επιστρέφουν τις μεταβλητές data και nextNode αντίστοιχα. Η κλάση List<T> έχει 3 μεταβλητές στιγμιοτύπου : την firstNode που αναφέρεται στον πρώτο κόμβο της λίστας , την lastNode που αναφέρεται στον τελευταίο κόμβο της λίστας και την name που αναφέρεται στο όνομα της δομής. Η List<T> έχει δύο κατασκευαστές: έναν που δημιουργεί μια κενή λίστα με το όνομα list και έναν που αρχικοποιεί τις μεταβλητές στιγμιοτύπου. Η κλάση List<T> έχει τις ακόλουθες μεθόδους : την insertAtFront που δημιουργεί έναν νέο κόμβο και εισάγει σε αυτόν τα στοιχείο που μας δίνεται και εισάγει τον κόμβο στην αρχή της λίστας, την insertAtBack που δημιουργεί έναν νέο κόμβο και εισάγει σε αυτόν τα στοιχείο που μας δίνεται και εισάγει τον κόμβο στο τέλος της λίστας , την removeFromFront που αφαιρεί το πρώτο στοιχείο της λίστας και προκαλείται NoSuchElementException εάν η λίστα είναι άδεια, την firstItem που επιστρέφει τα δεδομένα του πρώτου κόμβου, την removeFromBack που αφαιρεί το τελευταίο στοιχείο της λίστας και προκαλείται NoSuchElementException εάν η λίστα είναι άδεια , την isEmpty που επιστρέφει true αν η λίστα είναι άδεια , την printList που τυπώνει τα στοιχεία της λίστας και την size που επιστρέφει το μέγεθος της λίστας. Η κλάση StringStackImpl<T> κληρονομεί την κλάση List<T> και υλοποιεί την διασύνδεση StringStack<T>. Η μέθοδος push αντιστοιχεί στην insertAtFront, η μέθοδος pop στην removeFromFront, η peek αντιστοιχεί στην firstItem και η printStack στην printList. Οι μέθοδοι isEmpty και size κληρονομούνται απευθείας από την κλάση List<T>. Με αυτόν τον τρόπο κατασκευάσαμε μια στοίβα με τη χρήση λίστας μονής σύνδεσης που παράλληλα υλοποιεί όλες τις μεθόδους της StringStack<T>. Αντίστοιχα υλοποιήσαμε την ουρά. Η κλάση StringQueueImpl<T> κληρονομεί την List<T> και υλοποιεί τη διασύνδεση StringQueue<T>. Η μέθοδος put αντιστοιχεί στην insertAtBack, η μέθοδος get στην removeFromFront, η μέθοδος peek αντιστοιχεί στην firstItem και η μέθοδος printQueue αντιστοιχεί στην printList. Οι μέθοδοι isEmpty και size κληρονομούνται απευθείας από την κλάση List<T>. Με αυτόν τον τρόπο κατασκευάσαμε μια ουρά με τη χρήση λίστας μονής σύνδεσης που παράλληλα υλοποιεί όλες τις μεθόδους της StringQueue<T>.

Ερώτημα Γ

Στο ερώτημα Γ κατασκευάσαμε την ουρά χρησιμοποιώντας ένα δείκτη, τον lastNode που είναι ο δείκτης του τελευταίου κόμβου. Για να γίνει αυτό χρησιμοποιήσαμε κυκλική λίστα αντί για λίστα μονής σύνδεσης. Η διαφορά είναι ότι ο τελευταίος κόμβος δείχνει στον πρώτο κόμβο και έτσι υλοποιείται η κυκλική λίστα. Η κλάση CircularList<T> έχει τρεις μεταβλητές στιγμιοτύπου : ListNode<T> lastNode, String name και size. Στον κατασκευαστή αρχικοποιούμε τον τελευταίο κόμβο της λίστας σε null και το μέγεθος της λίστας σε 0. Με τη μέθοδο insert εισάγουμε έναν κόμβο που περιέχει το στοιχείο insertItem στο τέλος της λίστας. Αν ο τελευταίος κόμβος είναι null τότε στη θέση του βάζουμε το νέο στοιχείο και ο τελευταίος κόμβος δείχνει στον εαυτό του. Σε άλλη περίπτωση ο καινούριος κόμβος δείχνει στον πρώτο κόμβο, ο τελευταίος κόμβος δείχνει στον καινούριο και ο νέος κόμβος γίνεται τελευταίος. Η μέθοδος remove αφαιρεί το στοιχείο από την αρχή της λίστας και προκαλεί NoSuchElementException εάν η λίστα είναι άδεια. Εάν η λίστα περιέχει ένα στοιχείο ο τελευταίος κόμβος γίνεται null ενώ σε κάθε άλλη περίπτωση ο τελευταίος κόμβος δείχνει στον δεύτερο κόμβο που πλέον έχει γίνει πρώτος αφού αφαιρούμε τον πρώτο κόμβο από τη λίστα. Στο τέλος επιστρέφουμε τα δεδομένα του πρώτου κόμβου, που είναι ο κόμβος ο οποίος αφαιρέθηκε από τη λίστα και μικραίνουμε κατά 1 το μέγεθος της λίστας. Η μέθοδος firstItem επιστρέφει τα δεδομένα του πρώτου κόμβου αν η λίστα δεν είναι άδεια αλλιώς προκαλείται NoSuchElementException. Η μέθοδος isEmpty επιστρέφει true αν ο τελευταίος κόμβος είναι null αλλιώς false. Η μέθοδος printList εκτυπώνει τα στοιχεία της λίστας. Αν η λίστα είναι άδεια εκτυπώνει κατάλληλο μήνυμα, αλλιώς ξεκινάμε από τον πρώτο κόμβο και εκτυπώνουμε τα δεδομένα κάθε κόμβου μέχρι να φτάσουμε στον τελευταίο κόμβο. Τέλος η size επιστρέφει το μέγεθος της λίστας. Χρησιμοποιώντας την συγκεκριμένη κυκλική λίστα κατασκευάζουμε την ουρά που χρησιμοποιεί έναν δείκτη, τον τελευταίο. Η κλάση StringQueueWithOnePointer<T> κληρονομεί την κλάση CircularList<T> και υλοποιεί τη διεπαφή StringQueue<T>. Οι μέθοδοι isEmpty και size κληρονομούνται απευθείας από την υπερκλάση. Η μέθοδος put αντιστοιχεί στην insert , αφού εισάγουμε στο τέλος της ουράς το στοιχείο που θέλουμε, η μέθοδος get αντιστοιχεί στη remove αφού αφαιρούμε το παλιότερο στοιχείο της ουράς, η μέθοδος peek αντιστοιχεί στην firstItem , αφού επιστρέφει το πρώτο στοιχείο της ουράς και η printQueue αντιστοιχεί στην printList αφού εκτυπώνει τα στοιχεία της ουράς ξεκινώντας από το πρώτο. Η StringQueueWithOnePointer<T> υλοποιεί όλες της μεθόδους της διεπαφής StringQueue<T> χρησιμοποιώντας έναν μόνο δείκτη.

Ερώτημα Β

Για να διαβάσουμε το αρχείο που περιέχει το λαβύρινθο αποθηκεύουμε το path σε μία μεταβλητή τύπου Path με τη βοήθεια της συνάρτησης get της κλάσης Paths. Το path που χρειαζόμαστε δίνεται στη γραμμή εντολών οπότε ως όρισμα της get χρησιμοποιούμε το args[0] . Από αυτό το path για να πάρουμε το όνομα του αρχείου, χρησιμοποιούμε την getFileName της κλάσης Path και αποθηκεύουμε το αποτέλεσμα στην μεταβλητή fileName. Έπειτα, δημιουργούμε ένα αντικείμενο τύπου Scanner για να μπορούμε να διαβάσουμε το αρχείο. Σ ε ένα πίνακα String[]

dimensions αποθηκεύουμε την πρώτη γραμμή του αρχείου που μας δίνει πληροφορίες για τις διαστάσεις του πίνακα. Στη μεταβλητή rows αποθηκεύουμε το πρώτο στοιχείο του πίνακα dimensions που αντιπροσωπεύει τον αριθμό γραμμών, ενώ στη μεταβλητή columns αποθηκεύουμε το δεύτερο στοιχείο του πίνακα dimensions που αντιπροσωπεύει τον αριθμό στηλών. Στη συνέχεια φτιάχνουμε έναν δισδιάστατο πίνακα χαρακτήρων τον myArray που έχει μέγεθος rows x columns. Δημιουργούμε έναν πίνακα String [] entrance και αποθηκεύουμε σε αυτόν τη δεύτερη γραμμή του αρχείου που μας δίνει πληροφορίες για τις συντεταγμένες της εισόδου (E) στον πίνακα. Αποθηκεύουμε αυτές τις συντεταγμένες στις μεταβλητές entrance_x και entrance_y. Προκειμένου να ελέγξουμε εάν τα δεδομένα εισόδου είναι έγκυρα δηλαδή αν ο πίνακας έχει τον αριθμό στηλών και γραμμών που αναγράφονται στην πρώτη γραμμή και η είσοδος βρίσκεται στις συντεταγμένες που μας δίνονται στη δεύτερη γραμμή ορίζουμε τις μεταβλητές lines και numberOfColumns που μετράνε πόσες γραμμές και στήλες έχει ο πίνακας που μας δόθηκε. Σε ένα πίνακα String[] Line αποθηκεύουμε όλα τα στοιχεία μιας γραμμής του πίνακα και αυξάνουμε το lines κατά 1 όσο διαβάζουμε μία καινούρια γραμμή, ενώ ο αριθμός των στηλών θα ισούται με το μέγεθος του πίνακα line. Έπειτα αποθηκεύουμε στον myArray κάθε στοιχείο του πίνακα line. Για να ελέγξουμε εάν τα δεδομένα εισόδου είναι σωστά χρησιμοποιούμε την ακόλουθη συνθήκη: if (lines==rows && numberOfColumns==columns && myArray[entrance_x][entrance_y]=='E'). Εάν τα δεδομένα που δόθηκαν ήταν σωστά συνεχίζουμε για να βρούμε την έξοδο του λαβυρίνθου εάν υπάρχει αλλιώς εκτυπώνεται κατάλληλο μήνυμα. Για να βρούμε τις πιθανές εξόδους δημιουργούμε έναν πίνακα int[][] exits μεγέθους rows*2+columns*2-5 x 2 αφού μετράμε κάθε άκρη μία φορά και αφαιρούμε την είσοδο. Γεμίζουμε τον πίνακα με τις πιθανές εξόδους δηλαδή τις συντεταγμένες των στοιχείων της πρώτης γραμμής ή στήλης ή της τελευταίας γραμμής ή στήλης που έχουν την τιμή 0. Στη συνέχεια δημιουργούμε τον πίνακα boolean [][]visited που τον αρχικοποιούμε σε false και θα γίνεται true όποτε επισκεπτόμαστε ένα σημείο του λαβυρίνθου. Χρησιμοποιούμε τη στοίβα που κατασκευάσαμε στο μέρος Α προκειμένου να αποθηκεύσουμε το μονοπάτι μέχρι την έξοδο του λαβυρίνθου. Η στοίβα που κατασκευάσαμε είναι τύπου Node , όπου η συγκεκριμένη κλάση αντιπροσωπεύει έναν κόμβο του λαβυρίνθου με 3 μεταβλητές στιγμιοτύπου τις x,y και direction. Στο Node current εισάγουμε έναν νέο κόμβο με συντεταγμένες τις συντεταγμένες εισόδου. Με τη μέθοδο push εισάγουμε στη στοίβα τον τωρινό κόμβο και ορίζουμε μία boolean exit_found η οποία θα γίνει true εάν ο λαβύρινθος έχει κάποια έξοδο. Όσο η στοίβα μας δεν είναι άδεια και δεν έχουμε βρει κάποια έξοδο κάνουμε τα ακόλουθα βήματα : η κορυφή της στοίβας (stack.peek()) γίνεται ο τωρινός κόμβος, αρχικοποιούμε τα direction, x, y με τη βοήθεια των get που έχουμε ορίσει στην κλάση Node. Ορίζουμε μία μεταβλητή found που γίνεται true εάν βρούμε κάποιο σημείο στον πίνακα πάνω, δεξιά, κάτω ή αριστερά που να είναι μηδέν. Στη συνέχεια αφαιρούμε με τη μέθοδο pop το πρώτο στοιχείο της στοίβας και εισάγουμε με την push το current στη στοίβα. Εάν το x και το y του current αντιστοιχούν σε κάποιες από τις πιθανές συντεταγμένες κάποιας εξόδου τότε εκτυπώνουμε ότι βρήκαμε την έξοδο , βάζουμε σε κάποιο Node n την

κορυφή της στοίβας , εκτυπώνουμε τις συντεταγμένες της εξόδου και το exit_found γίνεται true. Στη συνέχεια κάνουμε 4 διαφορετικά if για κάθε κατεύθυνση. Σε κάθε ένα από αυτά ελέγχουμε την τιμή της κατεύθυνσης και την τιμή του found. Μέσα σε κάθε if αυξάνουμε την τιμή της κατεύθυνσης κατά 1 για να δούμε εάν στην επόμενη τιμή της κατεύθυνσης υπάρχει κάποιο σημείο 0. Έπειτα ελέγχουμε εάν το x ή y έχουν τιμές στα όρια του πίνακα και εάν οι τιμές του myArray στα x-1, αν η κατεύθυνση είναι 0 δηλαδή κινούμαστε προς τα επάνω, y-1 αν η κατεύθυνση είναι 1 δηλαδή κινούμαστε προς τα δεξιά, x+1, αν η κατεύθυνση είναι 2 δηλαδή κινούμαστε προς τα κάτω και y+1 αν η κατεύθυνση είναι 3 δηλαδή κινούμαστε προς τα αριστερά και εάν ο πίνακας visited είναι false για τις συγκεκριμένες τιμές. Αν όλη αυτή η συνθήκη είναι αληθής τότε βάζουμε σε ένα Node temp τις συντεταγμένες που προαναφέρθηκαν, κάνουμε το visited στις συντεταγμένες αυτές true και με την push εισάγουμε στη στοίβα τον κόμβο temp και κάνουμε το found true αφού βρήκαμε κάποιο σημείο του λαβυρίνθου που μπορεί να μας οδηγήσει στην έξοδο. Εάν το found παραμείνει false πηγαίνουμε πίσω , δηλαδή κάνουμε το visited με τις συντεταγμένες του τωρινού κόμβου true και αφαιρούμε με την pop το πρώτο στοιχείο της στοίβας. Σε κάθε περίπτωση η κατεύθυνση ξαναγίνεται 0 με τη βοήθεια της set που ορίζεται στην κλάση Node. Έξω από αυτό το while loop εάν η στοίβα είναι άδεια σημαίνει ότι δεν βρέθηκε έξοδος οπότε εκτυπώνεται κατάλληλο μήνυμα. Σε περίπτωση που το αρχείο δε βρεθεί προκαλείται FileNotFoundException και εκτυπώνεται κατάλληλο μήνυμα.

Σημείωση: έτρεξα το πρόγραμμα του ερωτήματος Β στη γραμμή εντολών δίνοντας ολόκληρο το path όπως αναφέρεται στις οδηγίες ακολουθούμενο από /όνομα_αρχείου οπότε το path αποθηκεύεται στο args[0].