Java Structured Programming

Mini projects

Πρόλογος

Τι λέει η Wikipedia για την SCRUM.

Within project management, **scrum**, sometimes written **Scrum** or **SCRUM**, is a framework for developing, delivering, and sustaining products in a complex environment,[1] with an initial emphasis on software development, although it has been used in other fields including research, sales, marketing and advanced technologies.[2] It is designed for teams of ten or fewer members, who break their work into goals that can be completed within time-boxed iterations, called *sprints*, no longer than one month and most commonly two weeks. The scrum team assess progress in time-boxed daily meetings of 15 minutes or less, called daily scrums (a form of stand-up meeting). At the end of the sprint, the team holds two further meetings: the sprint review which demonstrates the work done to stakeholders to elicit feedback, and sprint retrospective which enables the team to reflect and improve.

Name

The software development term *scrum* was first used in a 1986 paper titled "The New New Product Development Game" by Hirotaka Takeuchi and Ikujiro Nonaka.[4][5] The paper was published in the Jan 1986 issue of *Harvard Business Review*. The term is borrowed from rugby, where a scrum is a formation of players. The term *scrum* was chosen by the paper's authors because it emphasizes teamwork.[6]

Scrum is occasionally seen written in all-capitals, as *SCRUM*.[7] While the word itself is not an acronym, its capitalized styling likely comes from an early paper by Ken Schwaber[8] that capitalized *SCRUM* in its title.[9][10]

While the trademark on the term *scrum* itself has been allowed to lapse, it is now deemed as a generic trademark owned by the wider community rather than an individual.[11]

Many of the terms used in scrum literature are typically written with leading capitals (e.g., *Scrum Master*, *Daily Scrum*), but in many cases should not be capitalized if they are common nouns. To maintain correct grammar, this article uses normal sentence case for these terms (e.g., *scrum master*, *daily scrum*) – unless they are recognized marks (such as *Certified Scrum Master* and *Professional Scrum Master*).

Key ideas

Scrum is a lightweight, iterative and incremental framework for developing, delivering, and sustaining complex products.[12][13] The framework challenges assumptions of the traditional, sequential approach to product development, and enables teams to self-organize by encouraging physical co-location or close online collaboration of all team members, as well as daily face-to-face communication among all team members and disciplines involved.

Ο πρόλογος δεν σχετίζεται με τα ακόλουθα Projects στην Java, μιας και την μεθοδολογία Scrum την χρησιμοποιούμε για να τρέχουμε ομαδικά projects. Ωστόσο είναι χρήσιμη μία εισαγωγή στην Scrum, την οποία θα ξαναδούμε.

Project 01

Αναπτύξτε ένα πρόγραμμα σε Java που να διαβάζει από ένα αρχείο ακέραιους αριθμούς μέχρι να βρει την τιμή -1 (το αρχείο πρέπει να περιέχει περισσότερους από 6 αριθμούς και το πολύ 49 αριθμούς) με τιμές από 1 έως 49. Τους αριθμούς αυτούς τους εισάγει σε ένα πίνακα, τον οποίο ταξινομεί (π.χ. με την Arrays.sort()). Στη συνέχεια, το πρόγραμμα παράγει όλες τις δυνατές εξάδες (συνδυασμούς 6 αριθμών). Ταυτόχρονα και αμέσως μετά την παραγωγή κάθε εξάδας **'φιλτράρει' κάθε εξάδα** ώστε να πληροί τα παρακάτω κριτήρια: 1) Να περιέχει το πολύ 4 άρτιους, 2) να περιέχει το πολύ 4 περιττούς, 3) να περιέχει το πολύ 2 συνεχόμενους, 4) να περιέχει το πολύ 3 ίδιους λήγοντες, 5) να περιέχει το πολύ 3 αριθμούς στην ίδια δεκάδα.

Τέλος, εκτυπώνει τις τελικές εξάδες σε ένα αρχείο με όνομα της επιλογής σας και κατάληξη.txt.

Hint. Ακολουθήστε τη διαδικασία που είχαμε δει για την παραγωγή 4άδων. Κάθε παραγόμενη εξάδα μπορεί να αποθηκεύεται σε ένα πίνακα ο οποίος στη συνέχεια να ελέγχεται από κάθε μία από τις αναφερόμενες μεθόδους (φίλτρα). Αν για παράδειγμα μία εξάδα έχει αποθηκευτεί στον πίνακα arr, τότε για να 'περάσει' τα φίλτρα που είναι ταυτόχρονα περιορισμοί, θα πρέπει να ελεγχθεί. Π.χ. if (!isEven(arr)) && (!isOfdd(arr)) && (!isContiguous(arr)) && (!isSameEnding(arr)) && (!isSameTen), γράψε την εξάδα στο αρχείο εξόδου.

Project 02

Δημιουργήστε μία εφαρμογή επαφών για ένα κινητό τηλέφωνο, η οποία μπορεί να περιέχει μέχρι 500 επαφές. Κάθε επαφή έχει Επώνυμο, Όνομα και Τηλέφωνο.

Για να αποθηκεύεται τις επαφές χρησιμοποιήστε ένα δυσδιάστατο πίνακα 500x3 όπου στην $1^η$ θέση κάθε επαφής θα αποθηκεύεται το Επώνυμο, στη $2^η$ θέση το Όνομα και στην $3^η$ θέση το τηλέφωνο, όλα ως String.

Υλοποιήστε τις βασικές **CRUD** πράξεις: Αναζήτηση Επαφής με βάση το τηλέφωνο, Εισαγωγή Επαφής (αν δεν υπάρχει ήδη), Ενημέρωση Επαφής (εάν υπάρχει), Διαγραφή Επαφής (εάν υπάρχει).

Η main() θα πρέπει να εμφανίζει ένα μενού στον χρήστη, οποίος θα επιλέγει την κατάλληλη πράξη ή Έξοδο από την εφαρμογή και με την κατάλληλη καθοδήγηση της εφαρμογής θα επιτελεί την επιλεγμένη πράξη.

Project 03

Αναπτύξτε μία εφαρμογή που διαβάζει έναν-έναν τους χαρακτήρες ενός αρχείου και τους εισάγει σε ένα πίνακα 256x2. Κάθε θέση του πίνακα είναι επομένως ένας πίνακας δύο θέσεων, όπου στην 1η θέση αποθηκεύεται ο χαρακτήρας που έχει διαβαστεί (αν δεν υπάρχει ήδη στον πίνακα) και στην 2η θέση αποθηκεύεται το πλήθος των φορών που έχει διαβαστεί (βρεθεί) κάθε χαρακτήρας. Χαρακτήρες θεωρούνται και τα κενά και οι αλλαγές γραμμής και γενικά οποιοσδήποτε UTF-8 χαρακτήρας.

Στο τέλος η main() παρουσιάζει στατιστικά στοιχεία για κάθε χαρακτήρα όπως η συχνότητα εμφάνισής του στο κείμενο ταξινομημένα ανά χαρακτήρα και ανά συχνότητα εμφάνισης.

Project 04

Έστω ένας δισδιάστατος πίνακας που περιέχει τα στοιχεία άφιξης και αναχώρησης αυτοκινήτων σε μορφή *arr[][] = {{1012, 1136}, {1317, 1417}, {1015, 1020}}* Αναπτύξτε μία εφαρμογή που διαβάζει και εκτυπώνει τον μέγιστο αριθμό αυτοκινήτων που είναι σταθμευμένα το ίδιο χρονικό διάστημα. Για παράδειγμα στον παραπάνω πίνακα το αποτέλεσμα θα πρέπει να είναι: 2. Το 1ο αυτοκίνητο αφίχθη στις 10:12 και αναχώρησε στις 11:36, το 3ο αυτοκίνητο αφίχθη στις 10:15 και αναχώρησε στις 10:20. Επομένως, το 1ο και το 3ο αυτοκίνητο ήταν παρόντα το ίδιο χρονικό διάστημα.

Hint. Με βάση τον αρχικό πίνακα, δημιουργήστε ένα δυσδιάστατο πίνακα που σε κάθε γραμμή θα περιέχει δύο πεδία int. Στο πρώτο πεδίο θα εισάγεται η ώρα άφιξης ή αναχώρησης από τον αρχικό πίνακα και στο 2ο πεδίο θα εισάγεται ο αριθμός 1 αν πρόκειται για άφιξη και 0 αν πρόκειται για αναχώρηση.

Ταξινομήστε τον πίνακα σε αύξουσα σειρά με βάση την ώρα. Στη συνέχεια υπολογίστε το μέγιστο αριθμό αυτοκινήτων που είναι σταθμευμένα το ίδιο χρονικό διάστημα με ένα πέρασμα του πίνακα.

Project 05

Έστω ένας **ταξινομημένος** πίνακας με επαναλαμβανόμενα στοιχεία. Γράψτε μία μέθοδο *int[] getLowAndHighIndexOf(int[] arr, int key)* που να υπολογίζει και να επιστρέφει τα low και high index ενός πίνακα arr, για ένα ακέραιο key που λαμβάνει ως παράμετρο.

Γράψτε και μία main() που να βρίσκει το low και high index για τον πίνακα {0, 1, 4, 4, 4, 6, 7, 8, 8, 8, 8, 8}. Για παράδειγμα, αν δώσουμε ως τιμή το 8, θα πρέπει να επιστρέφει {7, 11}.

Hint. Ελέγξτε αν το key περιέχεται στον πίνακα και σε ποια θέση. Αν ναι, τότε από τη θέση αυτή μετρήστε τα στοιχεία όσο υπάρχουν στοιχεία με ίδια τιμή και μέχρι να βρείτε το τέλος του πίνακα.

Project 06

Έστω ένας πίνακας n ακεραίων. Τότε ο maximum sum subarray ο είναι ο συνεχόμενος υποπίνακας (contiguous subarray - δυνητικά κενό) με το μεγαλύτερο άθροισμα. Σχεδιάστε έναν γραμμικό αλγόριθμο (με πολυπλοκότητα O(n)) για να επιλύσετε τα παραπάνω πρόβλημα. Για παράδειγμα, αν έχουμε τον πίνακα {−2, 1, −3, 4, −1, 2, 1, −5, 4} τότε ο συνεχόμενος υποπίνακας με το μέγιστο άθροισμα είναι ο {4, −1, 2, 1}, του οποίου το άθροισμα είναι 6.

Δώστε μια λύση τριών μερών της ακόλουθης μορφής:

(α) Περιγράψτε (με λόγια και σχήματα) ξεκάθαρα τον αλγόριθμό σας.

(β) Γράψτε τον κώδικα σε Java .

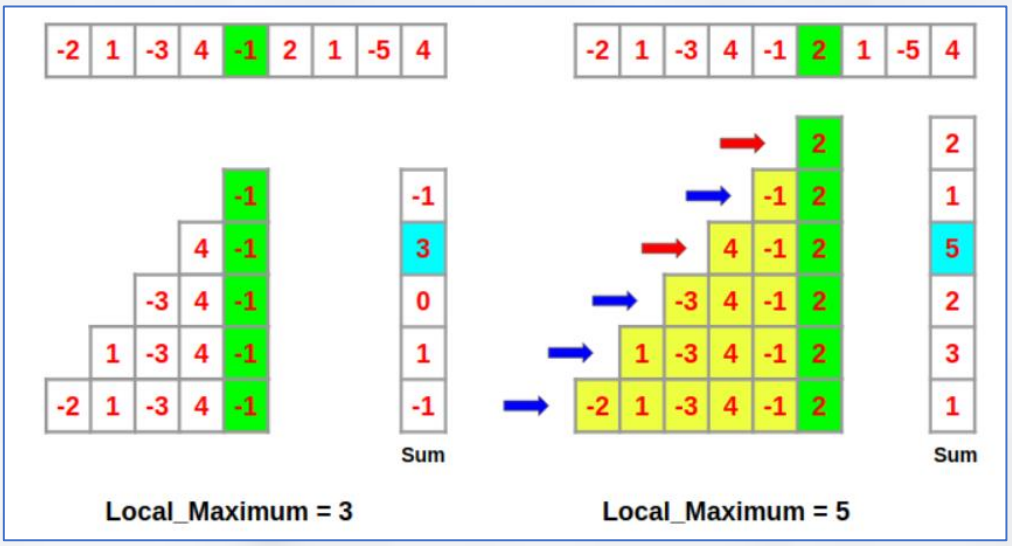
(γ) Δείξτε ότι η πολυπλοκότητα χρόνου είναι O(n)

Hint. Χρησιμοποιήστε δυναμικό προγραμματισμό (βασική αρχή στον δυναμικό προγραμματισμό είναι όταν υπολογίζουμε κάτι να το αποθηκεύουμε, ώστε αν το ξαναχρειαστούμε να μην το ξαναυπολογίζουμε). Μην υπολογίζετε ξανά και ξανά το άθροισμα για όλους τους δυνατούς υποπίνακες. Αν βρείτε ένα τοπικό μέγιστο (το μέγιστο από τη θέση 0 μέχρι μία θέση i του πίνακα arr) τότε για τη θέση i + 1 το μέγιστο θα είναι το max(τοπικό μέγιστο (i – 1) + arr[i], arr[i]).

Παρατηρήστε στον παρακάτω πίνακα ότι αν έχουμε υπολογίσει το τοπικό μέγιστο για τη θέση arr[4] όπου το τοπικό μέγιστο είναι το 3, τότε για την επόμενη θέση του πίνακα, την θέση arr[5], το τοπικό μέγιστο είναι max(local-max(i – 1) + arr[i], i), δηλαδή max(3 + 2, 2) = 5.

Ο πιο εύκολος τρόπος να λυθεί το πρόβλημα είναι ο **επαναληπτικός** με μία for.

Για την αναδρομική λύση μπορούμε ξεκινώντας από το τελευταίο στοιχείο (n-1) να υπολογίσουμε **αναδρομικά** τα local maxima. Το local maximum του arr[0] είναι η τιμή του arr[0]. Στο παράδειγμα είναι arr[0] = -2. Μπορούμε επίσης να έχουμε μία μεταβλητή static int σε επίπεδο κλάσης που να είναι το globalMaximum και να ενημερώνεται από όλες τις μεθόδους στο βάθος της αναδρομής, όταν το localMaximum είναι μεγαλύτερο από το globalMaximum.



Project 07

Γράψτε δύο μεθόδους που αφορούν την αντιγραφή δυσδιάστατων πινάκων. Μία μέθοδο int[][] shallowCopy(int[][] arr) που αντιγράφει ένα δυσδιάστατο πίνακα αλλά μόνο τις τιμές του βασικού πίνακα που είναι αναφορές στους πίνακες που είναι στοιχεία του βασικού πίνακα. Και μία μέθοδο int[][] deepCopy(int[][] arr).

Γράψτε μία main που να δείχνετε γιατί το shallow copy δεν δουλεύει όπως θα θέλαμε, αφού αλλάζοντας ένα στοιχείο ενός πίνακα από δύο για παράδειγμα copies, αλλάζει το στοιχείο και στον άλλο πίνακα, αφού κατά βάση πρόκειται για ένα και μόνο κοινό στοιχείο (αφού έχει γίνει shallow copy).

Δείξτε και την περίπτωση του deep copy. Δείξτε ότι δουλεύει όπως θα θέλαμε. Δηλαδή δεν επηρεάζουν οι αλλαγές στοιχείων το κάθε copy, το οποίο τώρα είναι ανεξάρτητο.

Hint. Ένας δυσδιάστατος πίνακας είναι ένας βασικός πίνακας που έχει ως στοιχεία πίνακες. Η shallowCopy() αντιγράφει ένα δυσδιάστατο πίνακα χρησιμοποιώντας κάποια μέθοδο όπως Arrays.copyOf() ή System.arraycopy(). Όμως αυτές οι μέθοδοι κάνουν shallow copies δηλαδή αντιγράφουν τις αναφορές των υποπινάκων του βασικού πίνακα. Αν δηλαδή θεωρήσουμε ότι ένας δυσδιάστατος πίνακας αποτελείται από μία κάθετη στήλη που είναι ο βασικός πίνακας και οριζόντιες γραμμές που αντιστοιχούν στους πίνακες που είναι στοιχεία ου βασικού πίνακα (οπότε τελικά έχουμε ένα δυσδιάστατο πίνακα), τότε το shallow copy αντιγράφει μόνο τα στοιχεία του βασικού πίνακα που είναι όμως οι αναφορές προς τους οριζόντιους πίνακες. Αυτό σημαίνει πως ο πίνακας που έχει αντιγραφεί δεν είναι ανεξάρτητο copy αλλά αν αλλάξει κάτι στα στοιχεία του αντιγραμμένου πίνακα, αλλάζουν ταυτόχρονα και τα στοιχεία του αρχικού πίνακα, αφού μόνο οι αναφορές έχουν αντιγραφεί και όχι και τα στοιχεία των πινάκων στην οριζόντια διάσταση. Αφού οι πίνακες είναι mutable (και όχι immutable όπως τα Strings) οι αλλαγές που γίνονται στο ένα copy αφορούν έμμεσα και το άλλο copy, κάτι το οποίο δεν είναι σωστό για mutable στοιχεία πινάκων (όπως πίνακες και κλάσεις ή οποιοδήποτε άλλο mutable στοιχείο).

Project 08

Αναπτύξτε ένα παιχνίδι Τρίλιζα, όπου δύο παίκτες παίζουν X και O (ή 1 και 2 αν θέλετε να υλοποιήσετε με πίνακα ακεραίων και όχι με πίνακα char) και κερδίζει ο παίκτης που έχει συμπληρώσει τρία ίδια σύμβολα ή αριθμούς σε οποιαδήποτε διάσταση του πίνακα, οριζόντια, κάθετα ή διαγώνια.

Η main() μπορεί να ελέγχει τη ροή του παιχνιδιού, όπως ποιος παίκτης παίζει κάθε φορά (εναλλαγή μεταξύ των δύο παικτών), να διαβάζει από το stdin το σύμβολο που δίνει ο κάθε παίκτης και να εμφανίζει με γραφικό τρόπο (όπως είχαμε δει σε αντίστοιχο παράδειγμα στην τάξη) την τρίλιζα μετά από κάθε κίνηση κάθε παίκτη.

Ενώ, μπορείτε να δημιουργήσετε και μία μέθοδο που να ελέγχει (μετά από κάθε κίνηση) αν ο παίκτης που έκανε την κίνηση έκανε τρίλιζα.

Το πρόγραμμα θα πρέπει να λαμβάνει υπόψη την περίπτωση ισοπαλίας όπως και να μην επιτρέπει ένας παίκτης να παίξει σε θέση που είναι ήδη κατειλημμένη.

Project 09

Μία από τις πρώτες εφαρμογές των Η/Υ ήταν η κρυπτογράφηση. Ένας απλός τρόπος κρυπτογράφησης είναι η κωδικοποίηση κάθε χαρακτήρα με ένα ακέραιο με βάση ένα **κλειδί κρυπτογράφησης.** Μία τέτοια **μέθοδος κρυπτογράφησης** περιγράφεται στη συνέχεια.

Κωδικοποίησε **τον 1ο χαρακτήρα του μηνύματος με την ακέραια τιμή που αντιστοιχεί σε αυτόν** (από τον κώδικα ASCII). Κωδικοποίησε του επόμενους χαρακτήρες: (α) **προσθέτοντας** την ακέραια ASCII τιμή του καθένα από αυτούς με τον κωδικό του προηγούμενού του, (β) παίρνοντας **το υπόλοιπο της διαίρεσης του αθροίσματος** αυτού διά μία σταθερά. Η σταθερά αυτή ονομάζεται **κλειδί (key)** κρυπτογράφησης και (υποτίθεται πως) είναι μυστική.

Υποθέτουμε πως τα μηνύματα τελειώνουν με τον χαρακτήρα #.

Γράψτε ένα πρόγραμμα java που να υλοποιεί τον αλγόριθμο κρυπτογράφησης έτσι ώστε το κωδικοποιημένο μήνυμα που προκύπτει να είναι μία ακολουθία ακεραίων που τελειώνει με -1.

Γράψτε και τον αλγόριθμο αποκρυπτογράφησης που λαμβάνει ως είσοδο μία ακολουθία ακεραίων που τελειώνει με -1 και υπολογίζει το αρχικό μήνυμα.

Project 10

Έστω ένα θέατρο που έχει θέσεις όπου η κάθε θέση περιγράφεται με ένα χαρακτήρα που είναι η στήλη και ένα αριθμό που είναι η σειρά. Για παράδειγμα η θέση C2 βρίσκεται στην $2^{η}$ σειρά και $3^{η}$ στήλη.

Αναπτύξτε ένα πρόγραμμα διαχείρισης θεάτρου με 30 σειρές και 12 στήλες. Πιο συγκεκριμένα γράψτε μία μέθοδο void book(char column, int row) που να κάνει book μία θέση αν δεν είναι ήδη booked και μία μέθοδο void cancel(char column, int row) που να ακυρώνει την κράτηση μία θέσης αν είναι ήδη booked.

Hint. Υποθέστε ότι ο δυσδιάστατος πίνακας που απεικονίζει το θέατρο είναι ένα πίνακας από boolean, όπου το true σημαίνει ότι η θέση είναι booked και false ότι δεν είναι booked. Αρχικά όλες οι θέσεις πρέπει να είναι non-booked.