ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΕΠΛ 233: Αντικειμενοστρεφής Προγραμματισμός

Χειμερινό Εξάμηνο 2012

ΑΣΚΗΣΗ 4
Υλοποίηση Εφαρμογής Εστιατορίου (take-away)

Διδάσκων Καθηγητής: Παναγιώτης Ανδρέου

Ημερομηνία Υποβολής: 29/10/2012
Ημερομηνία Παράδοσης: 15/11/2012

ΠΕΡΙΓΡΑΦΗ

Σε αυτή την άσκηση καλείστε να υλοποιήσετε ένα πρόγραμμα προσομοίωσης σουβλατζίδικου take-away. Σκοπός της άσκησης είναι η υλοποίηση ενός προγράμματος μικρού μήκους χρησιμοποιώντας με τις βασικές αρχές του αντικειμενοστρεφή προγραμματισμού (π.χ., κληρονομικότητα, πολυμορφισμός, κτλ.) καθώς και η χρήση προχωρημένου προγραμματισμού όπως διαχείριση εξαιρέσεων και διαπροσωπείες.

ΑΝΑΛΥΤΙΚΗ ΠΕΡΙΓΡΑΦΗ

Το σουβλατζίδικο δέχεται τηλεφωνικές παραγγελίες μεταξύ 18:00 και 23:00 καθημερινά (το σουβλατζίδικο ανοίγει στις 17:30 και μένει ανοιχτό μέχρι τις 24:00). Για την εργασία αυτή μπορείτε να δουλέψετε σε ομάδες των τριών ατόμων.

Το σουβλατζίδικο διαθέτει φουκού μήκους Μ εκατοστών και Ν τηγάνια για πατάτες, όπου κάθε τηγάνι χωράει C μερίδες πατάτες. Επίσης τα κάρβουνα της φουκούς χρειάζονται χρόνο Τ μέχρι να είναι έτοιμα προς χρήση (για να ανάψουν). Το σουβλατζίδικο διεκπεραιώνει παραγγελίες για:

- Πίττα σουβλάκι χοιρινό (περιλαμβάνει δύο σμίλες)
- Πίττα σουβλάκι κοτόπουλο (περιλαμβάνει δύο σμίλες)
- Πίττα σεφταλιά (περιλαμβάνει δύο σμίλες)
- Πίττα μίξ (περιλαμβάνει μία σμίλα χοιρινό και μία σμίλα σεφταλιά)
- Τηγανιτές πατάτες

Για την προετοιμασία των τροφών, απαιτείται:

- 1 σμίλα σουβλάκι χοιρινό: 20-25 λεπτά ψήσιμο
- 1 σμίλα σουβλάκι κοτόπουλο: 15-20 λεπτά ψήσιμο
- 1 σμίλα σεφταλιά: 25 λεπτά ψήσιμο
- 1 πίττα: 5 λεπτά ψήσιμο
- 1 μερίδα πατάτες: 20 λεπτά τηγάνισμα

Κάθε σμίλα σουβλάκι καταλαμβάνει *x* εκατοστά χώρου από την φουκού, κάθε σμίλα σεφταλιά *y* εκατοστά και κάθε πίτα *z* εκατοστά.

Κάθε παραγγελία που παραδίδεται τηλεφωνικά, έχει τα ακόλουθα χαρακτηριστικά:

- Μοναδικό αύξοντα αριθμό
- Ώρα παραγγελίας
- Επιθυμητή ώρα παραλαβής
- Αριθμό από πίττες και μερίδες πατάτες

Μια παραγγελία μπορεί να αναπαρασταθεί στο πρόγραμμά σας ως εξής:

<**num** $\mathbf{t_{order}}$, $\mathbf{t_{req}}$, $\mathbf{n_{pp}}$, $\mathbf{n_{pc}}$, $\mathbf{n_{ps}}$, $\mathbf{n_{pm}}$, $\mathbf{n_{pf}}$>
όπου:

num: ο αύξων αριθμός της παραγγελίας
$\mathbf{t_{order}}$: αντιστοιχεί στη χρονική στιγμή λήψης της παραγγελίας, εκπεφρασμένη σε **αριθμό λεπτών** από τις 18:00, οπότε ανοίγει το σουβλατζίδικο, μέχρι τις 23:00 (π.χ. μια παραγγελία που έγινε στις 19:45 θα έχει χρονική στιγμή λήψης $\mathbf{t_{order}}$ = 105).
$\mathbf{t_{req}}$: αντιστοιχεί στη χρονική στιγμή κατά την οποία ο πελάτης επιθυμεί να παραλάβει την παραγγελία του, εκπεφρασμένη σε αριθμό λεπτών από τις 18:00, οπότε ανοίγει το σουβλατζίδικο, μέχρι τις **24:00**.
$\mathbf{n_{pp}}$: αριθμός από πίττες σουβλάκια της παραγγελίας.
$\mathbf{n_{pc}}$: : αριθμός από πίττες κοτόπουλο της παραγγελίας.
$\mathbf{n_{ps}}$: αριθμός από πίττες σεφταλιά της παραγγελίας.
$\mathbf{n_{pm}}$: αριθμός από πίττες μίξ της παραγγελίας.
$\mathbf{n_{pf}}$: αριθμός από μερίδες πατάτες της παραγγελίας.

Παραδείγματα αποδεκτών παραγγελιών ακολουθούν:

- <14 75, 105, 1, 0, 0, 0, 1> : ο 14ος πελάτης τηλεφωνεί στις 7:15μμ και θέλει να παραλάβει μέχρι τις 7:45μμ μια πίττα σουβλάκια χοιρινό και μια πατάτες
- <15 65, 135, 10, 2, 0, 5, 10> : ο 15ος πελάτης τηλεφωνεί στις 7:05μμ και θέλει να παραλάβει μέχρι τις 8:15μμ δέκα πίττες σουβλάκια χοιρινό, δύο πίττες κοτόπουλο, πέντε σεφταλιά και δέκα πατάτες (σουβλάκι πάρτυ).

Υποθέστε ότι στατιστικές μετρήσεις των παραγγελιών στο σουβλατζίδικο έχουν δείξει ότι:

- Κάθε ημέρα, το 20% των εισερχόμενων παραγγελιών αφορά σε μια πίττα μόνο, το 35% των εισερχόμενων παραγγελιών αφορά σε δύο πίττες, το 10% των παραγγελιών αφορά σε τρείς πίττες, το 20% των παραγγελιών αφορά σε τέσσερεις πίττες, ενώ το υπόλοιπο ποσοστό των παραγγελιών αφορά σε 5-20 πίττες. Θεωρείστε ότι σε μια τυχαία παραγγελία, υπάρχουν οι ίδιες πιθανότητες μια τυχαία πίττα να ανήκει σε οποιαδήποτε είδος (σουβλάκι χοιρινό, κοτόπουλο, σεφταλιά ή μίξ).
- Επίσης, θεωρείστε ότι κάθε ημέρα, στο 60% των παραγγελλόμενων πιττών, καθεμιά πίττα συνοδεύεται από μια μερίδα πατάτες. Το 35% των παραγγελλόμενων πιττών δεν συνοδεύονται από πατάτες. Τέλος, ένα 5% των παραγγελλόμενων πιττών συνοδεύεται από δύο μερίδες πατάτες.
- Οι χρονικές στιγμές άφιξης των παραγγελιών είναι διεσπαρμένες χρονικά και ακολουθούν κανονική κατανομή (Gaussian) με μέση τιμή 180 (δηλ. 21:00) και διασπορά 60.
- Η κατανομή της επιθυμητής ώρας παραλαβής κυμαίνεται από τριάντα λεπτά μέχρι τρείς ώρες από την λήψη της παραγγελίας - σε παραγγελίες πάνω των 10 πιττών, ο επιθυμητός χρόνος παραλαβής δεν μπορεί να είναι λιγότερος της μιας ώρας.

ΖΗΤΗΜΑ 1ο:

Υλοποιήστε ένα πρόγραμμα γεννήτρια παραγγελιών **OrderGenerator**, η οποία να δημιουργεί τις παραγγελίες μιας ημέρας. Η γεννήτρια πρέπει να δέχεται σαν παραμέτρους εισόδου τους αριθμούς των παραγγελιών που θέλουμε να δημιουργήσουμε. Αν π.χ. θέλουμε να εξετάσουμε ένα σενάριο με 1000 παραγγελίες, θα πρέπει να εκτελέσουμε το πρόγραμμα ως εξής:

java OrderGenerator 1000

Το πρόγραμμα αυτό στην έξοδό του πρέπει να δημιουργεί ένα αρχείο *orders.txt*, το οποίο ξεκινάει με μια γραμμή η οποία περιέχει τον αριθμό των παραγγελιών του (π.χ. 1000) και μετά ακολουθούν οι παραγγελίες για το συγκεκριμένο σενάριο. Κάθε παραγγελία εμφανίζεται σε διαφορετική γραμμή του αρχείου και έχει την ακόλουθη δομή:

num t_{order} $t_{delreq;}$ n_{pp} n_{pc} n_{ps} n_{pm} n_{pf}

Οι παραγγελίες πρέπει να εμφανίζονται **ταξινομημένες με βάση τη χρονική σειρά της παράδοσής** τους στο σουβλατζίδικο. Οι παραγγελίες πρέπει να ακολουθούν τα στατιστικά χαρακτηριστικά που περιγράφηκαν πιο πάνω.
Το ζητούμενο είναι **η υλοποίηση** του προγράμματος.

ΖΗΤΗΜΑ 2ο:

Πρέπει να υλοποιήσετε ένα πρόγραμμα για την εκτέλεση των παραγγελιών με τρείς διαφορετικούς αλγόριθμους χρονοπρογραμματισμού. Το πρόγραμμα σας **OrderDelivery** δέχεται σαν παραμέτρους την χωρητικότητα της φουκούς (M) και το χρόνο προετοιμασίας των κάρβουνων (T), τον αριθμό των τηγανιών (N) και της χωρητικότητας τους (C) καθώς και το χώρο που καταλαμβάνουν οι σμίλες (*x,y*) και οι πίττες (*z*). Εν συνεχεία, διαβάζει το αρχείο *orders.txt* και υπολογίζει τον χρόνο ολοκλήρωσης της κάθε παραγγελίας βάση του αλγόριθμου που του ζητήθηκε, την απόκλιση από την επιθυμητή ώρα παράδοσης της και τον αριθμό των πελατών που μένουν ευχαριστημένοι (ευχαριστημένος είναι ο πελάτης του οποίου η παραγγελία είναι έτοιμη όχι αργότερα από τον επιθυμητό χρόνο παραλαβής του). Π.χ. η εντολή

java OrderDelivery 400 45 10 4 3 6 12 2

Θα ξεκινήσει το πρόγραμμα με μια φουκού μήκους 400 εκατοστών που χρειάζεται 45 λεπτά για να ανάψει, 10 τηγάνια που χωρούν 4 μερίδες, σμίλες σουβλάκι των 3 εκατοστών, σμίλες σεφταλιά των 6 εκατοστών, πίττες των 12 εκατοστών και κάνοντας χρήση του αλγόριθμού 2.

Η έξοδος του προγράμματος είναι ένα αρχείο *deliveries.txt*. Το αρχείου ξεκινάει με μια γραμμή η οποία περιέχει τον αριθμό των παραγγελιών που παραλαμβάνονται, τον μέσο όρο της απόκλισης από τις επιθυμητές ώρες παράδοσης των παραγγελιών και τον αριθμό των ευχαριστημένων πελατών, (π.χ. 1000 22,54 170), και μετά ακολουθούν πληροφορίες για την διεκπεραίωση των παραγγελιών. Για κάθε παραγγελία, εμφανίζονται σε διαφορετική γραμμή του αρχείου οι ακόλουθες πληροφορίες:

num t_{order} t_{del} $(t_{del} - t_{delreq})$ $(n_{pp}+n_{pc}+n_{ps}+n_{pm})$ n_{pf}

π.χ. η γραμμή: 10 150 195 15 5 1 σημαίνει ότι η παραγγελία 10 έγινε στις 20:30, παραδόθηκε στις 21:15, είχε 15 λεπτά καθυστέρηση και περιείχε 5 πίττες και μια μερίδα πατάτες.

Αλγόριθμος 1:

Υποθέστε ότι ο σουβλιστής εκτελεί τις παραγγελίες με την σειρά που αυτές παραλαμβάνονται (δηλ., FCFS - first come first served).

Αλγόριθμος 2:

Υποθέστε ότι ο σουβλιστής εκτελεί τις παραγγελίες διαλέγοντάς τη μικρότερη παραγγελία (μικρότερο χρόνο εκτέλεσης) που είναι διαθέσιμη και είναι δυνατό (χωρά στη φουκού) να εκτελεστεί τη δεδομένη στιγμή.

Αλγόριθμος 3:

Υποθέστε ότι ο σουβλιστής για να εκτελέσει τις παραγγελίες διαλέγει τη παραγγελία με το μεγαλύτερο βάρος που είναι διαθέσιμη και είναι δυνατό να εκτελεστεί τη δεδομένη στιγμή. Το βάρος της κάθε παραγγελίας υπολογίζεται ως εξής:

$$w = \frac{t_{exec} - \left(t_{delreq} - t_{now}\right)}{t_{exec}}$$, όπου:

t_{now}: η τρέχουσα χρονική στιγμή εκφρασμένη σε αριθμό λεπτών από τις 18:00
t_{exec}: ο χρόνος που απαιτείτε για την εκτέλεση της παραγγελία σε λεπτά.

Προφανώς το πρόγραμμα προσομοίωσης κατά τους υπολογισμούς του πρέπει ανά πάσα στιγμή να λαμβάνει υπόψη μόνο τις παραγγελίες που έχουν ήδη δοθεί. Για αυτό το σκοπό μπορείτε να χρησιμοποιήσετε διακριτό χρόνο.

Τέλος, εκτός από την δημιουργία του αρχείου εξόδου, το πρόγραμμα πρέπει να αναπαριστά γραφικά (κάνοντας χρήση των βιβλιοθηκών που σας δόθηκαν) τη διεκπεραίωση των παραγγελιών.

Εξετάστε ποιος αλγόριθμος δίνει τη μικρότερη συνολική απόκλιση και πόσοι πελάτες μένουν ευχαριστημένοι. Δώστε μια μικρή αναφορά (με γραφικές παραστάσεις) που δείχνει πως αλλάζει η μέση απόκλιση και ο αριθμός των ικανοποιημένων πελατών για κάθε αλγόριθμο για 5 διαφορετικά σενάρια παραγγελιών – 200, 400, 600, 800 και 1000 παραγγελίες. Εκτός από την αναφορά πρέπει να παραδώσετε και τα αρχεία που παράγονται (20 συνολικά – για κάθε σενάριο ένα αρχείο orders.txt και τρία αρχεία deliveries.txt).

Για τον υπολογισμό τυχαίου αριθμού που ακολουθεί την κανονική (Gaussian) κατανομή μπορείτε να χρησιμοποιήσετε την μέθοδο nextGaussian της κλάσης Random του πακέτου java.util. Η nextGaussian επιστρέφει τυχαίους αριθμούς που ακολουθούν την κανονική κατανομή με μέση τιμή 0 και διασπορά 1. Για την μετατροπή σε κανονική μορφή με διαφορετική μέση τιμή (μ) και διασπορά (σ^2) μπορείτε να χρησιμοποιήσετε τον ακόλουθο τύπο:

$$\sigma * nextGaussian + \mu$$

Προσέξτε ότι ο τύπος χρησιμοποιεί την τυπική απόκλιση και όχι τη διασπορά (σ και όχι σ^2).

ΑΞΙΟΛΟΓΗΣΗ

Για την αξιολόγηση του προγράμματος σας, θα ληφθούν οι πιο κάτω παράμετροι:

- **Ορθότητα**: ο πρόγραμμα σας πρέπει τρέχει ορθά για οποιαδήποτε είσοδο
- **Κατανοητά σχόλια**: Γράφετε κατανοητά σχόλια που να εξηγούν την λειτουργία της κάθε κλάσης/πεδίου/μεθόδου.
- **Αποδοτικότητα**: θα πρέπει να γίνει επαναχρησιμοποίηση αντικειμένων όπου αρμόζει
- **Χαμηλή σύζευξη μεταξύ των αντικειμένων**: κάθε αντικείμενο πρέπει να υποστηρίζει λειτουργίες που είναι άμεσα συσχετιζόμενες με αυτό
- **Δώστε ιδιαίτερη προσοχή στη χρήση των βασικών αρχών του αντικειμενοστραφή προγραμματισμού (π.χ., κληρονομικότητα, αφαιρετικές κλάσεις, διαπροσωπείες, πολυμορφισμός, εξαιρέσεις).**

ΓΕΝΙΚΕΣ ΟΔΗΓΙΕΣ

- Η κάθε κλάση θα πρέπει να υλοποιηθεί σε ξεχωριστό αρχείο JAVA.
- Στο πάνω μέρος της κάθε κλάσης να γράψετε με σχόλια το ονοματεπώνυμο και τον αριθμό ταυτότητας σας.
- Βεβαιωθείτε ότι τα προγράμματα σας είναι ορθά και τρέχουν.
- Η άσκηση σας θα πρέπει να παραδοθεί στον υπεύθυνο βοηθό σαν ένα zip αρχείο (που να περιλαμβάνει όλες τις κλάσεις που θα υλοποιήσετε και το έγγραφο τεκμηρίωσης) σε ηλεκτρονική μορφή και σε έντυπη μορφή την ημερομηνία παράδοσης.
- Το zip αρχείο θα είναι πρέπει να γίνει μέσω export από το eclipse σαν archive file.
- Το zip αρχείο θα ονομάζεται με τον εξής τρόπο:
  epl233.ex4.<ταυτότητα>.zip
- Μη τήρηση των ημερομηνιών παράδοσης των εργασιών συνεπάγεται τις ανάλογες βαθμολογικές επιπτώσεις (μέχρι τον μηδενισμό της εργασίας).
- Οι προγραμματιστικές ασκήσεις θα ελέγχονται από ειδικό πρόγραμμα για την ανίχνευση των αντιγραφών. Οι αντιγραμμένες εργασίες θα μηδενίζονται και για τους αντιγραφείς θα εφαρμόζονται οι κανόνες τού Πανεπιστημίου. Αποφύγετε λοιπόν την αντιγραφή προγραμμάτων από άλλους συναδέλφους σας, διότι έτσι εκτίθετε και αυτούς και τον εαυτό σας στον κίνδυνο μηδενισμού και πειθαρχικής δίωξης.

ΚΑΛΗ ΕΠΙΤΥΧΙΑ!!!