1η Εργασία Λειτουργικά Συστήματα

Δημητρίου Δημήτρης Π18036

Ο κάθε φιλόσοφος υλοποιείται ως ένα ξεχωριστό νήμα (η κλάση Philosopher κληρονομεί από την τάξη Thread).Η κλάση αποτελείται από τα εξής πεδία: τον ακέραιο Id(ο αριθμός του φιλοσόφου),την ακέραια priority(η προτεραιότητα που θα χρειαστούμε αργότερα στον χρονοπρογραμματισμό),τα πεδία που εκφράζουν τον χρόνο εκτέλεσης (int) δηλαδή το runtime και το previousRuntime,την ακέραια μεταβλητή secondsToEat που υποδηλώνουν τα δευτερόλεπτα που θα δοθούν στον φιλόσοφο για να φάει καθώς και την ακέραια totalSeconds που εκφράζει τα συνολικά δευτερόλεπτα που θα δοθούν στον φιλόσοφο για να γευματίσει(δηλαδή 20 συνολικά όπως ορίζει η άσκηση).

Ακόμα δημιουργούμε ένα αντικείμενο rand της κλάσης Random ,το αντικείμενο f της κλάσης DateFormatter το οποίο ακολουθεί το πρότυπο HH::mm::ss καθώς και ορίζουμε μία ArrayList waitingTime τύπου Long η οποία χρησιμοποιείται όπως θα δούμε για τον μέσο χρόνο αναμονής για την απόκτηση των πιρουνιών.

Η κλάση Philosopher αποτελείται από τις εξής μεθόδους:

Ο Constructor Philosopher(int Id);.Εφαρμόζεται για την αρχικοποίηση των instance variables του Philosopher.Επίσης χρησιμοποιούμε το this(αναφορά στο τρέχον αντικείμενο) για να αποκτήσουμε πρόσβαση στα μέλη (members) της «τρέχουσας» τάξης. Η μεταβλητή Id τίθεται ίση με το όρισμα του κατασκευαστή, τα previousRuntime,secondsToEat,totalSeconds ίσα με το 0 ενώ το runtime μέσα από την βοήθεια του αντικειμένου random που δημιουργήσαμε ίσα με ένα αριθμό ανάμεσα στο 1-3 καθώς και ο χρόνος εκτέλεσης runtime θα είναι ίσος με την προτεραιότητα επί το κβάντο χρόνου(είναι 1 θα το ορίσουμε στην τάξη Main).

void changePriorities():Όπως ήδη αναφέραμε πιο πάνω η μεταβλητή previousRuntime εκφράζει τον προηγούμενο συνολικό χρόνο που δόθηκε στο νήμα Philosopher.Έτσι κάθε φόρα που θα καλείται αυτή η μέθοδος θα ανανεώνεται η μεταβλητή αυξάνοντας το runtime που είχε προηγουμένως. Η προτεραιότητα υπολογίζεται ξανά χρησιμοποιώντας την random για να πάρει τιμές από 1-3,ενώ χρησιμοποιώντας την μέθοδο java.lang.Thread.currentThread() εκχωρούμαι στο τρέχον εκτελούμενο νήμα αντικείμενο philosopher εκ νέου μία προτεραιότητα.To runtime ανανεώνεται όπως είπαμε πιο πάνω και ακολουθεί και μήνυμα όσον αναφορά προς τις αλλαγές. Η μέθοδος αυτή δημιουργήθηκε με σκοπό να υπάρχει πρόβλεψη ώστε να μην περιμένουν διαρκώς οι διεργασίες με χαμηλή προτεραιότητα αλλά να ανανεώνονται οι προτεραιότητες δυναμικά ως προς τον χρόνο.

Η μέθοδος run():

Όσο τα συνολικά δευτερόλεπτα που χρησιμοποιούνται για να φάει ένας φιλόσοφος δεν έχουν ξεπεράσει τα 20 ορίζουμε την τοπική μεταβλητή τύπου long startTime που εκφράζει τον τρέχοντα χρόνο σε millisecond και την ενεργοποιούμε για να τρέξει .Αμέσως μετά χρησιμοποιώντας ένα monitor (του οποίου η χρήση θα εξηγήσουμε σε λίγο) καλούμε την take\_forks(Id) έτσι ώστε ό φιλόσοφος να αποκτήσει τα πιρούνια. Αφού τα αποκτήσει μέσα από την long μεταβλητή endTime δεχόμαστε τον τρέχοντα χρόνο σε milliseconds και την διαφορά των δύο μεταβλητών την προσθέτουμε στην λίστα waitingTime.Στην συνέχεια καλούμε την Eat() για να φάει ο φιλόσοφος και μετά καλούμε την μέθοδο που έχει οριστεί στο monitor put\_forks(Id) έτσι ώστε να προσπαθήσει μήπως φάει κάποιος άλλος. Όλα αυτά εκτελούνται μέσα στο while όταν η συνθήκη είναι αληθής και μέσα σε try catch block έτσι ώστε να μην προκύψει exception .Η μέθοδος printStackTrace() αν προκύψει exception εκτυπώνει το throwable μαζί με άλλες πληροφορίες όπως το όνομα της κλάσης και την γραμμή που προέκυψε το σφάλμα.

Η μέθοδος Eat():

Τα δευτερόλεπτα που θα φάει υπολογίζονται τυχαία με πιθανές τιμές από 1-20(δηλαδή μπορεί να φάει μονομιάς ή να φάει λίγο και μετά ξανά).Αν τα δευτερόλεπτα για να φάει ο φιλόσοφος μαζί με τα δευτερόλεπτα που είχε φάει πιο πριν ξεπερνάνε τα 20 τότε προσαρμόζονται τα δευτερόλεπτα που θα φάει ,έτσι ώστε να μην φάει συνολικά πάνω από 20.Τα συνολικά δευτερόλεπτα κάθε φορά ανανεώνονται ,ενώ εμφανίζονται μηνύματα σχετικά για πόσα δευτερόλεπτα θα φάει ,τί ώρα τρώει. Γίνεται προσομοίωση του δείπνου θεωρώντας ότι για τα δευτερόλεπτα που του αναλογούν να κοιμηθεί(sleep) δηλαδή αναστέλλεται. Πάλι γίνεται πρόβλεψη για την αντιμετώπιση του exception.

Η κλάση ourMonitor

Για να μην προκύψει ποτέ αδιέξοδο αποφασίσαμε για την συγγραφή της άσκησης να χρησιμοποιηθεί ένας ελεγκτής. Με τον ελεγκτή μόνο ένα νήμα είναι ενεργό σε αυτόν .Άλλο ένα πλεονέκτημα από την χρήση του είναι ότι ο μεταγλωττιστής θα ασχοληθεί με τον αμοιβαίο αποκλεισμό όποτε είναι απίθανο να προκύψει τέτοιο λάθος.

Η κλάση υλοποίει το interface State που θα εξηγήσουμε παρακάτω. Αρχικά ορίζουμε αντικείμενο f της κλάσης DateFormatter το οποίο ακολουθεί το πρότυπο HH::mm::ss,ένα πίνακα ακεραίου state[],έναν πίνακα self[] τύπου Condition και ένα lock entlock της ReentrantLock,την οποία προτιμήσαμε εξαιτίας της ευελιξίας έναντι σηματοφόρων που χρησιμοποιούνται σε ελεγκτή .Μια  ReentrantLock lock ανήκει στο thread που την έχει κλειδώσει αλλά δεν το έχει ξεκλειδώσει ακόμα. Βασικό πλεονέκτημα είναι ότι το νήμα μπορεί να αποκτήσει την κλειδαριά παραπάνω από μια φορά.

Constructor public ourMonitor(int num)

Δημιουργούμε το αντικείμενο entlock(αποκτά reference),τον πίνακα state,self ενώ μέσω ενός βρόγχου δίνονται αρχικές τιμές στο self μέσω του lock με ένα αντικείμενο Condition(new Condition) και το state με την κατάσταση Thinking που όπως θα δούμε είναι 0.

Μέθοδος void take\_forks(int i)

Κλειδώνει η κλειδαριά από το νήμα καθώς μπαίνουμε σε κρίσιμη περιοχή, το state του φιλοσόφου της παραμέτρου i γίνεται Hungry,εκτυπώνονται μηνύματα για την κατάσταση φιλοσόφου και lock.Και καλείται η μέθοδος test(i).Αργότερα αν μετά την κλήση της test, ο φιλόσοφος δεν είναι σε κατάσταση Eating τότε μέσω των μεταβλητών συνθήκης το νήμα μπλοκάρεται (εκτελεί await()) και επιτρέπει σε οποιαδήποτε άλλο νήμα που είχε απορριφθεί από τον ελεγκτή να χρησιμοποιηθεί. Ξεκλειδώνεται η κλειδαριά και έπειτα εκτυπώνεται μήνυμα για την κατάστασης της.

Μέθοδος void test(i)

Αν ο φιλόσοφος που ο αριθμός δίνεται από την παράμετρο βρίσκεται σε κατάσταση Hungry και οι διπλανοί φιλόσοφοι δεν τρώνε τότε μπορεί να πάρει τα πιρούνια και κάνει την κατάσταση του σε Eating.Επίσης ξυπνάει αν έχει μπλοκαριστεί προηγουμένως το νήμα μέσω της signal().Αν είναι όμως ψευδής η συνθήκη τότε θα πρέπει αιτιολογηθεί γιατί δεν μπόρεσε να φάει ο φιλόσοφος εξετάζοντας την περιπτώσεις να έτρωγε ο αριστερός φιλόσοφος , ο δεξιός είτε απλά δεν πείναγε. Εκτυπώνονται ανάλογα μηνύματα σε σχέση με τι ισχύει.

Void put\_forks():

Κλειδώνει η κλειδαριά. Η κατάσταση του I φιλοσόφου γίνεται Thinking,καλεί την μέθοδο Thinking() (δηλαδή αφήνει τα πιρούνια).Έπειτα εξετάζει αν μπορεί να φάει είτε ο αριστερά είτε ο δεξιά καλώντας την test.Γίνεται unlock στην entlock.

Private void Thinking():

Παραμένει σε κατάσταση Thinking ένα τυχαίο διάστημα (δηλαδή blocked)

Το interface State

Αποτελείται από τις τιμές Thinking=0,Hungry=1,Eating=2 που εκφράζουν τις καταστάσεις ενός φιλοσόφου.

Η κλάση Main

Αποτελείται από τα στατικά μεταβλητές num(εκφράζει τον αριθμό των φιλοσόφων που θα δοθεί),τις στατικές μεθόδους Left(i),Right(i) που βρίσκουν τον αριστερά και δεξιά του i,την μεταβλητή ακέραια max που θα εκφράζει την μέγιστη προτεραιότητα, την ακέραια position τον αριθμό του φιλοσόφου, την ακέραια sum άθροισμα και count3 μετρητή.

Στην main αρχικά μέσω ενός scanner και της Boolean ok γίνεται έλεγχος ότι δόθηκε ακέραιος αριθμός από 3-10 που εκφράζει τον αριθμό των φιλοσόφων. Στην συνέχεια δημιουργείται πίνακας από φιλοσόφους P[] με βάση τον αριθμό που δόθηκε καθώς και monitor(instances). Έπειτα μέσω βρόγχου εκχωρείται για τον κάθε φιλόσοφο μια προτεραιότητα μέσω της setPriority της java.Εμφανίζονται αντίστοιχα μηνύματα.

Για κάθε στοιχείο του πίνακα δημιουργείται ένα thread και εκτελείται η run του αφού άλλωστε καλούμε την start().Μέσα από την τοπική Boolean itsok όσο είναι false μηδενίζουμε τον counter,αλλάζουμε τις προτεραιότητες για το κάθε στοιχείο πίνακα φιλόσοφο. Τώρα θα εξηγήσουμε τον αλγόριθμο χρονοπρογραμματισμού .Αν ο φιλόσοφος δεν έχει τελειώσει το φαγητό του δηλαδή δεν έχει φάει για 20 seconds,ανάμεσα σε αυτούς τους φιλοσόφους βρίσκουμε ποιος έχει την μεγαλύτερη προτεραιότητα. Σε περίπτωση ισοβαθμία επιλέγεται αυτός που είχε πάρει τον λιγότερο χρόνο εκτέλεσης .Άρα ο max έχει την μέγιστη προτεραιότητα και το position το id του φιλοσόφου. Αν ο φιλόσοφος δεν έχει τελειώσει τον φαγητό και δεν είναι αυτός με την μέγιστη προτεραιότητα τότε το thread καλεί την yield δηλαδή ενημερώνει τον scheduler ότι δεν κάνει κάτι σημαντικό και λέει ότι καλό είναι να σταματήσει .Ο scheduler μπορεί να αγνοήσει αυτό το σήμα για αυτό έπειτα ο φιλόσοφος θα κοιμηθεί όσο runtime(seconds) δικαιούται αυτός που έχει την μέγιστη προτεραιότητα. Έτσι γίνεται χρονοπρογραμματισμός των νημάτων.

Επιπλέον αναζητούμε αν έχουν τελειώσει όλοι οι φιλόσοφοι την λειτουργία τους .Αν ναι ο counter θα πρέπει να είναι 0 αλλιώς θα πραγματοποιηθεί ξανά ο βρόγχος.

Τέλος μόλις τελειώσει το while loop , με την βοήθεια του μετρητή count3 υπολογίζουμε μέσα από την λίστα waitingTime του κάθε φιλοσόφου τον χρόνο αναμονής σε δευτερόλεπτα (στοιχεία της λίστας δια μέγεθος της λίστας του κάθε φιλοσόφου) καθώς και τον συνολικό χρόνο αναμονής(Το άθροισμα sum είναι το άθροισμα των χρόνων αναμονής του κάθε φιλοσόφου δια του πλήθους του num που είναι ο αριθμός των φιλοσόφων).Εκτυπώνονται αντίστοιχα μηνύματα.

Η εργασία υλοποιήθηκε σε περιβάλλον Intellij Idea χρησιμοποιώντας έννοιες αντικειμενοστραφούς προγραμματισμού ,ενώ το .exe που θα περιέχει ο φάκελος τρέχει σε Windows.

Τα screenshots έχουν τραβηχτεί όταν εκτελούνταν το πρόγραμμα σε περιβάλλον Intellij Idea της JetBrains.







