Λειτουργικά Συστήματα

Τμήμα Πληροφορικής
Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης

2η Εργασία
Χειμερινό Εξάμηνο 2020

Στόχος της εργασίας είναι η εξοικείωσή σας με αλγόριθμους δρομολόγησης διεργασιών, καθώς και αλγόριθμους τοποθέτησης δυναμικής τμηματοποίησης κύριας μνήμης.

Για το σκοπό αυτό, σας ζητείται να αναπτύξετε κώδικα σε γλώσσα *Java*, ο οποίος θα υλοποιεί τους αντίστοιχους αλγόριθμους. Για τους αλγόριθμους δρομολόγησης αυτοί είναι οι FCFS, SJF (χωρίς προεκχώρηση) και Round Robin (για οποιαδήποτε τιμή κβάντου), ενώ για τους αλγόριθμους δυναμικής τμηματοποίησης μνήμης είναι οι Best Fit, Worst Fit, First Fit και Next Fit. Τα διεκπεραιωτικά κομμάτια του κώδικα έχουν υλοποιηθεί και σας παραδίδονται έτοιμα. Αυτά ορίζουν ένα απλοποιημένο μοντέλο λειτουργίας ενός υπολογιστικού συστήματος.

Σύμφωνα με αυτό το απλοποιημένο μοντέλο λειτουργίας, όταν δημιουργείται μια νέα διεργασία βρίσκεται στην κατάσταση *NEW*. Το σύστημα τότε προσπαθεί να τοποθετήσει τη διεργασία στη μνήμη RAM, σε ένα από τα διαθέσιμα τμήματα που υπάρχουν εκεί, τα οποία ενδεχομένως δεν είναι όλα του ίδιου μεγέθους. Αν η διεργασία τοποθετηθεί σε κάποιο από τα διαθέσιμα τμήματα, μεταβαίνει στην κατάσταση *READY* και είναι διαθέσιμη πια για να εκτελεστεί από τη CPU. Αν η διεργασία δεν χωρέσει σε κάποιο από τα διαθέσιμα τμήματα, θα πρέπει να περιμένει σε κατάσταση *NEW*, μέχρι να ελευθερωθεί χώρος στη μνήμη όταν τερματίσει κάποια άλλη, προηγούμενη, διεργασία. Όταν μια διεργασία εκτελείται από τη CPU, μεταβαίνει στην κατάσταση *RUNNING*. Αν η εκτέλεση της διεργασίας διακοπεί, μεταβαίνει πάλι στην κατάσταση *READY* (αφού στο απλοποιημένο αυτό μοντέλο δεν συμπεριλαμβάνονται λειτουργίες εισόδου/εξόδου). Όταν τελικά η εκτέλεση της διεργασίας ολοκληρωθεί από τη CPU, μεταβαίνει στην κατάσταση *TERMINATED*, ενώ ο χώρος που καταλάμβανε στη μνήμη RAM ελευθερώνεται. Η αλληλουχία των καταστάσεων παρουσιάζεται στο Σχήμα 1.

Στον κατάλογο *code* που συνοδεύει την εκφώνηση, θα βρείτε τα αρχεία γλώσσας προγραμματισμού Java στα οποία θα πρέπει να βασίζετε τα παραδοτέα σας. Τα αρχεία αυτά είναι τα:

- **Process.java**: Κλάση που αναφέρεται στις διεργασίες και τις ιδιότητες και μεθόδους που έχουν αυτές. Μια διεργασία έχει κατ'ελάχιστο ένα Process Control Block, ένα

Σχήμα 1: Η αλληλουχία καταστάσεων μιας διεργασίας στο απλοποιημένο μοντέλο λειτουργίας που ακολουθείται

χρόνο άφιξης, χρόνο CPU καταιγισμού και απαιτήσεις σε μνήμη.

- **ProcessControlBlock.java**: Κλάση που αναφέρεται στο Process Control Block της κάθε διεργασίας. Αυτό περιλαμβάνει το PID της διεργασίας και την κατάστασή της. Διατηρεί επίσης ιστορικό με τους χρόνους στους οποίους η διεργασία εκκινήθηκε αλλά και αυτούς που σταμάτησε να εκτελείται.
- **ProcessState.java**: Ένα enumeration που απλά καταγράφει τις διαφορετικές καταστάσεις που ενδέχεται να έχει μία διεργασία.
- **Scheduler.java**: Abstract κλάση που χρησιμοποιείται ως βάση για τους αλγόριθμους δρομολόγησης CPU. Περιλαμβάνει μια λίστα με τις διεργασίες που έχουν φορτωθεί στη RAM και είναι διαθέσιμες για εκτέλεση (δηλαδή είτε σε κατάσταση *READY* ή *RUNNING*). Περιλαμβάνονται επίσης μέθοδοι για την προσθήκη μιας διεργασίας στη λίστα αυτή, αλλά και για την αφαίρεσή τους.
- **FCFS.java**: Κλάση που υλοποιεί την abstract κλάση Scheduler για τον αλγόριθμο δρομολόγησης FCFS.
- **SJF.java**: Κλάση που υλοποιεί την abstract κλάση Scheduler για τον αλγόριθμο δρομολόγησης SJF.
- **RoundRobin.java**: Κλάση που υλοποιεί την abstract κλάση Scheduler για τον αλγόριθμο δρομολόγησης Round Robin.
- **MMU.java**: Κλάση που αναφέρεται στη μονάδα διαχείρισης μνήμης του συστήματος. Αυτή περιλαμβάνει ένα χάρτη της μνήμης του συστήματος, με τα μεγέθη των διαθέσιμων τμημάτων μνήμης σε αυτή (τα οποία είναι χαρακτηριστικό του συστήματος και δεν μεταβάλλονται). Επίσης περιλαμβάνει μια λίστα των τμημάτων μνήμης που είναι κατηλημμένα από διεργασίες (η οποία μεταβάλλεται με την εκτέλεση των διεργασιών). Η MMU χρησιμοποιεί ένα αλγόριθμο διάθεσης της μνήμης στις διεργασίες.
- **MemorySlot.java**: Βοηθητική κλάση που υλοποιεί τα τμήματα μνήμης, το διαθέσιμο χώρο τους και τον χώρο που είναι κατηλλημένος.
- **MemoryAllocationAlgorithm.java**: Abstract κλάση που χρησιμοποιείται ως βάση για τους αλγόριθμους δυναμικής τμηματοποίησης μνήμης. Περιλαμβάνει ένα χάρτη της μνήμης του συστήματος και των τμημάτων που είναι κατηλημμένα, όπως παρέχονται

από την MMU. Απαιτεί από τις υποκλάσεις που βασίζονται σε αυτή να υλοποιείται μέθοδος η οποία θα τοποθετεί μια διεργασία σε κάποιο διαθέσιμο τμήμα μνήμης.

- **BestFit.java**: Κλάση που υλοποιεί την abstract κλάση MemoryAllocationAlgorithm για τον αλγόριθμο δυναμικής τμηματοποίησης μνήμης Best Fit.
- **WorstFit.java**: Κλάση που υλοποιεί την abstract κλάση MemoryAllocationAlgorithm για τον αλγόριθμο δυναμικής τμηματοποίησης μνήμης Worst Fit.
- **FirstFit.java**: Κλάση που υλοποιεί την abstract κλάση MemoryAllocationAlgorithm για τον αλγόριθμο δυναμικής τμηματοποίησης μνήμης First Fit.
- **NextFit.java**: Κλάση που υλοποιεί την abstract κλάση MemoryAllocationAlgorithm για τον αλγόριθμο δυναμικής τμηματοποίησης μνήμης Next Fit.
- **CPU.java**: Η κεντρική κλάση του συστήματος που υλοποιεί τη CPU. Μία CPU διαθέτει ένα αλγόριθμο δρομολόγησης διεργασιών και μία μονάδα διαχείρισης μνήμης. Ταυτόχρονα. γνωρίζει τις διεργασίες που έρχονται στο σύστημα και προφανώς θα πρέπει να γνωρίζει και τη διεργασία που εκτελείται σε κάθε χρονική στιγμή. Η λειτουργία της CPU επαναλαμβάνεται σε κάθε κύκλο του ρολογιού του συστήματος. Σε αυτό, η CPU, χρησιμοποιώντας την MMU, αποφασίζει αν μία διεργασία μπορεί να φορτωθεί στη μνήμη RAM, ενώ χρησιμοποιώντας τον αλγόριθμο δρομολόγησης, αποφασίζει ποια από τις διαθέσιμες διεργασίες που έχουν καταφτάσει, θα πρέπει να εκτελεστεί από αυτή.
- **PC.java**: Βοηθητικό «πρόγραμμα» που περιέχει τη μέθοδο *main* και μπορεί να χρησιμοποιηθεί ως παράδειγμα ενός συστήματος και των διεργασιών που καταφθάνουν σε αυτό.

Εσείς, θα πρέπει να συμπληρώσετε κώδικα μόνο στις μεθόδους αυτές που σημειώνονται με το σχόλιο

```
/* TODO: you need to add some code here */
```

Υπάρχουν επίσης κάποιες μέθοδοι οι οποίες συνοδεύονται από το σχόλιο:

```
/* TODO: you _may_ need to add some code here */
```

Σε αυτές, ενδέχεται να χρειαστεί να προσθέσετε κάποιο κώδικα, αλλά ενδέχεται και όχι, ανάλογα με τον τρόπο που θα επιλέξετε να τις υλοποιήσετε.

- Μπορείτε ελεύθερα να προσθέσετε όσες επιπλέον `private` ή `protected` μεθόδους ή ιδιότητες κρίνετε απαραίτητο. Το ίδιο ισχύει και για τους αντίστοιχους getters και setters.

- Δεν επιτρέπεται να προσθέσετε καμία άλλη `public` ιδιότητα ή μέθοδο σε οποιαδήποτε κλάση.
- Δεν επιτρέπεται να αλλάξετε τον τύπο των ιδιοτήτων που υπάρχουν ήδη.
- Δεν επιτρέπεται να αλλάξετε τον τύπο ή την υπογραφή των μεθόδων που υπάρχουν ήδη.
- Δεν επιτρέπεται να μεταβάλλετε με οποιοδήποτε τρόπο μεθόδους οι οποίες δεν περιλαμβάνουν το σχόλιο `TODO`.
- Δεν επιτρέπεται να προσθέσετε επιπλέον αρχεία/κλάσεις.
- Δεν επιτρέπετει να κάνετε `import` καμία επιπρόσθετη δομή από τη βιβλιοθήκη της Java, πέρα από την `java.util.ArrayList`, η οποία ήδη περιλαμβάνεται σε κάποια αρχεία.
- Σε κάποια αρχεία ενδεχομένως να μη χρειαστεί καμία προσθήκη/αλλαγή.
- Θα πρέπει να υλοποιήσετε όλους τους αλγόριθμους δρομολόγησης CPU που ζητούνται: FCFS, SJF και Round Robin.
- Θα πρέπει να υλοποιήσετε όλους τους αλγόριθμους δυναμικής τμηματοποίησης μνήμης που ζητούνται: Best Fit, Worst Fit, First Fit, Next Fit.
- Μπορείτε ελεύθερα να προσθέσετε εντολές `println` σε οποιοδήποτε σημείο του κώδικα, ώστε να παρακολουθείτε την εκτέλεσή του.
- Αλλάξτε τη μέθοδο *main* στο αρχείο **PC.java** ώστε να δοκιμάσετε διαφορετικές συνθήκες λειτουργίας (διεργασίες και διαθέσιμης μνήμης).
- Κάντε τις δοκιμές σας με (υποθετικά) συστήματα περιορισμένων πόρων. Πχ, ελέξτε τι συμβαίνει αν μία διεργασία δεν χωράει σε κανένα τμήμα μνήμης.
- Προσπαθήστε να σκεφτείτε όσο το δυνατό περισσότερες «ακραίες» συνθήκες και να τις καλύψετε με την υλοποίησή σας. Πχ, τι θα γίνει αν καμία διεργασία δεν χωράει στη μνήμη, αν όλες οι διεργασίες έρχονται ταυτόχρονα ή αν υπάρχουν «κενά» στην εκτέλεση διεργασιών από τη CPU.
- Παρατηρήστε τη συμπεριφορά του συστήματος όταν δημιουργούνται φαινόμενα όπως αυτά της φάλαγγας ή παρατεταμένης στέρησης.
- Για τον καλύτερο έλεγχο της υλοποίησής σας προτείνεται η συγγραφή unit tests που θα εξετάζουν διαφορετικά πιθανά προβλήματα της υλοποίησης. Σε αυτή την περίπτωση και μόνο σε αυτή, μπορείτε να προσθέσετε επιπλέον αρχεία *.java* στο παραδοτέο σας.

- Φροντίστε να σχολιάσετε επαρκώς τον κώδικά σας.

Η εργασία είναι **ομαδική** και θα οργανωθείτε σε ομάδες των **τεσσάρων ατόμων**. Στο .zip αρχείο που θα καταθέσετε, θα πρέπει να συμπεριλάβετε κι ένα αρχείο κειμένου, στο οποίο θα αναγράφονται τα **ονοματεπώνυμα** και **ΑΕΜ** όλων των μελών της ομάδας.

- Με τη συγκρότηση της ομάδας σας, θα πρέπει να αποστείλετε email με τη σύνθεσή της στο `gvlahavas@csd.auth.gr`, από όπου θα λάβετε και επιβεβαίωση.
- Η συγκρότηση των ομάδων θα πρέπει να έχει πραγματοποιηθεί και να έχει ολοκληρωθεί μέχρι τις **17/12/2020**. Δεν επιτρέπεται να δηλωθεί καμία νεά ομάδα ή να υπάρξουν αλλαγές στη σύσταση των ομάδων μετά από αυτή την ημερομηνία.
- Εργασίες θα γίνουν δεκτές μόνο από ομάδες που έχουν δηλωθεί μέχρι την παραπάνω ημερομηνία.
- Η εργασία θα πρέπει να υποβληθεί έγκαιρα, και μόνο από **ένα** μέλος της ομάδας.

Για την παράδοση της εργασίας, θα δημιουργήσετε ένα συμπιεσμένο αρχείο με όνομα *opsys2020-assignment2-1234-5678-1357-2468.zip*, αντικαθιστώντας τους τετραψήφιους αριθμούς με τους αριθμούς μητρώου των φοιτητών της ομάδας σας. Το αρχείο αυτό, θα ανεβάσετε στην πλατφόρμα elearning. Σε αυτό θα πρέπει να περιέχονται μόνο τα αρχεία πηγαίου κώδικα *.java*, με συμπληρωμένα τα κομμάτια κώδικα που σας ζητούνται σε κάθε περίπτωση. *Δεν* θα πρέπει να ανεβάσετε μαζί οποιαδήποτε άλλα αρχεία, όπως project files του IDE που χρησιμοποιήσατε κλπ. *Δεν* θα πρέπει να συνοδεύετε τον κώδικά σας από κάποιο εξωτερικό κείμενο, ο σχολιασμός θα πρέπει να γίνει αποκλειστικά και μόνο πάνω στον κώδικα.

Αν έχετε απορίες σχετικά με την εργασία, μπορείτε να τις υποβάλετε μέσω της πλατφόρμας elearning στο αντίστοιχο forum συζητήσεων.