ΠΑΡΑΛΛΗΛΟΣ ΚΑΙ ΚΑΤΑΝΕΜΗΜΕΝΟΣ ΥΠΟΛΟΓΙΣΜΟΣ

ΚΑΘΗΓΗΤΗΣ: ΜΑΡΓΑΡΙΤΗΣ ΚΩΝΣΤΑΝΤΙΝΟΣ

Αρχοντής-Εμμανουήλ Κωστής | ics21044

ΕΡΓΑΣΙΑ 1

1. Τι συμβαίνει αν στο πρόγραμμα `HelloThread.java` αφαιρέσουμε το loop με το `join()`?

Αν αφαιρέσουμε αυτή την for τότε η μέθοδος main δεν θα περιμένει να ολοκληρωθούν τα νήματα πριν προχωρήσει στην εκτέλεση της εντολής System.out.println("In main: threads all done"). Μετά την εκκίνηση όλων των νημάτων, η μέθοδος main θα εκτυπώσει αμέσως «In main: threads all done» και στη συνέχεια θα τερματίσει χωρίς να περιμένει τα 20 νήματα να ολοκληρώσουν την εκτέλεσή τους.
Στην πράξη η main Thread δεν θα περιμένει να τελειώσουν την εκτέλεση τους τα νήματα παιδιά που δημιουργεί.

2. Τι συμβαίνει αν στο ίδιο πρόγραμμα βάλουμε στο ίδιο loop με το `start()`, αμέσως μετά το `start()` το `join()`?

Αν βάλουμε το join μέσα σε αυτή την for αμέσως μετά το start τότε:
το κύριο νήμα (δηλ. η μέθοδος main) θα περιμένει να τελειώσει την εκτέλεσή του κάθε μεμονωμένο νήμα πριν ξεκινήσει το επόμενο. Αυτό ουσιαστικά αλλάζει το μοντέλο ταυτόχρονης εκτέλεσης από παράλληλη σε διαδοχική καθώς τα νήματα δεν εκτελούνται πλέον παράλληλα.
Αυτο είναι εμφανές και από το output στην κονσόλα. Το να το κάνουμε αυτό είναι αντίθετο με τον σκοπό της χρήσης των νημάτων και αναιρεί τους λόγους και τα οφέλη της χρήσης των νημάτων και της παράλληλης εκτέλεσης (π.χ. ταχύτητα, αποδοτικότητα).

3. Δείτε τη σειρά εκτυπώσεων στο πρόγραμμα `HelloThreadsArgs.java`. Τι παρατηρείτε?

Η έξοδος του προγράμματος δείχνει ότι τα νήματα δημιουργούνται και εκκινούνται με διαδοχική σειρά από το 0 έως το 19, αυτό φαίνεται από ακόλουθες γραμμές:

```
In main: create and start thread 0
In main: create and start thread 1
...
In main: create and start thread 19
```

Τα μηνύματα «Hello from Thread» δεν ακολουθούν την ίδια σειρά με τη δημιουργία του νήματος και μοιάζουν μπερδεμένα. Αυτό γίνεται γιατί τα νήματα εκτελούνται με μη ντετερμινιστική σειρά, κάτι το οποίο είναι συνηθισμένο στην ταυτόχρονη εκτέλεση. Για παράδειγμα:

```
Hello from thread 4 out of 20
Hello from thread 14 out of 20
Hello from thread 9 out of 20
...
Hello from thread 0 out of 20
```

Παρόμοια με τα μηνύματα «Hello from thread», τα μηνύματα «Thread X exits» εμφανίζονται επίσης με μη ντετερμινιστική σειρά.

```
Thread 9 exits
Thread 2 exits
Thread 4 exits
...
Thread 0 exits
```

Το τελικό μήνυμα «In main: threads all done» εκτυπώνεται μετά την ολοκλήρωση της εκτέλεσης όλων των νημάτων, υποδεικνύοντας ότι η μέθοδος join() κατάφερε να διασφαλίσει ότι το κύριο νήμα περιμένει την ολοκλήρωση όλων των νημάτων.

```
In main: threads all done
```

Παρατηρούμε επίσης ότι τα νήματα είναι αριθμημένα. Η ύπαρξη ορίσματων κλάσης στην κλάση MyThread επιτρέπει σε κάθε thread να αρχικοποιηθεί με συγκεκριμένα δεδομένα που μπορούν να χρησιμοποιηθούν κατά την εκτέλεσή του.

4. Στο πρόγραμμα `HelloThreadsArgs.java` μπορούμε να μην χρησιμοποιήσουμε δομή δεδομένων για τα νήματα; Τι αποτέλεσμα θα είχε στην εκτέλεση του προγράμματος;

Με την αποθήκευση των νημάτων σε μια δομή δεδομένων (όπως π.χ. έναν πίνακα), μπορούμε να καλέσουμε τη μέθοδο join() κάθε νήματος αργότερα στο πρόγραμμα μας, εξασφαλίζοντας πως η κύρια ροή εκτέλεσης (δηλ. η main thread) θα περιμένει μέχρι να τελειώσουν όλα τα νήματα την δουλειά τους πριν προχωρήσει στην εκτύπωσ. Αυτό σημαίνει ότι η κύρια ροή εκτέλεσης μπορεί να ολοκληρωθεί πριν τα νήματα έχουν τελειώσει τη δουλειά τους, οδηγώντας σε απροσδόκητα αποτελέσματα. Αφού δεν μπορούμε να βάλουμε την join() σε ξεχωριστή for τότε πρέπει να μπει στην ίδια for που φτιάχνει τα νήματα, όπως είδαμε όμως και στο ερώτημα 2, κάθε νήμα θα ξεκινά και θα το πρόγραμμα θα περιμένει να ολοκληρωθει πριν ξεκινήσει το επόμενο. Κάτι που ακυρώνει τα οφέλη της παράλληλης εκτέλεσης.
Επίσης αν δεν αποθηκεύαμε τα νήματα, τότε δεν θα είχαμε κάποιο τρόπο να τα διαχειριστούμε μετά τη δημιουργία τους κάτι που θα περιόριζε την ικανότητά μας να διαχειριστούμε την εκτέλεση του προγράμματος αποτελεσματικά. Ακόμη, η αποθήκευση των νημάτων μας επιτρέπει να ελέγξουμε την κατάσταση κάθε νήματος αργότερα. Για παράδειγμα για να δούμε αν ένα νήμα είναι ακόμα σε εκτέλεση, αν έχει ολοκληρωθεί, ή αν έχει πεθάνει, π.χ.:

```
for (int i = 0; i < numThreads; ++i) {
      if (threads[i].isAlive()) {
      System.out.println("Thread " + i + " is still running.");
```

```
        } else {
        System.out.println("Thread " + i + " has finished.");
        }
}
```

Τέλος, Η χρήση δομών δεδομένων για την αποθήκευση των νημάτων καθιστά τον κώδικα πιο επεκτάσιμο και ευκολότερο στη συντήρηση.