ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

Εξεταστική Περίοδος Φεβρουαρίου
2015 - 2016

Εργασία Ανάπτυξης Λογισμικού για Δίκτυα και Τηλεπικοινωνίες

ΟΝΟΜΑ: Γαλούνη Κωνσταντίνα Α.Μ: 1115201000034
ΟΝΟΜΑ: Γιαννακέλος Κωνσταντίνος Α.Μ: 1115201000029

ΔΙΔΑΣΚΩΝ: Αλωνιστιώτη Νάνσυ

ΑΘΗΝΑ – 2015

Πίνακας περιεχομένων

Περιγραφή Εκτέλεσης

Για την εργασία δημιουργήθηκε ένα νέο Maven Project. Το Maven είναι ένα build εργαλείο που χρησιμοποιείται σε Java Projects και επιτρέπει τη χρήση εξωτερικών jar αρχείων. Στην παρόν παραδοτέο αρκεί ένα απλό Java Project, ωστόσο προτιμήθηκα το Maven, δεδομένης της ευρείας χρήσής του και των λειτουργιών του που πιθανώς να μας είναι χρήσιμα στη συνέχεια.

Μεταγλώττιση-Εκτέλεση μέσω eclipse

Μεταγλώττιση: Κάνοντας δεξί κλικ στο Project, βλέπουμε την εντολή Run As. Έπειτα, επιλέγοντας την 5η εντολή "**Maven clean**" καθαρίζουμε τα αντικείμενα που δημιουργήθηκαν από πιθανές προηγούμενες μεταγλωττίσεις/εκτελέσεις. Τέλος, η 7η εντολή "**Maven install**" είναι υπεύθυνη για τη μεταλώττιση και εγκατάσταση του πακέτου στο τοπικό repository, για χρήση ως εξάρτηση σε άλλα projects τοπικά.

Εκτέλεση: Έπειτα από τα παραπάνω βήματα, με την απλή εντολή Run, ξεκινά η εκτέλεση του project.

Σημειώνεται πως δεδομένου ότι η εντολή "nmap" εκτελείται με δικαιώματα διαχειριστή, το eclipse πρέπει να ανοίξει μέσω terminal με την εντολή "sudo ./eclipse" στον αντίστοιχο φάκελο που είναι εγκατεστημένο.

Μεταγλώττιση-Εκτέλεση μέσω terminal

Μεταγλώττιση: Για τη μεταγλώττιση χρησιμοποιούνται οι ίδιες εντολές με παραπάνω, ωστόσο απαιτείται και μία προσθήκη στο αρχείο "pom.xml", προκειμένου να χρησιμοποιηθεί η σωστή έκδοση Java (1.7) και να είναι δυνατή η χρήση generics. Συγκεκριμένα προσθέτουμε στο αρχείο αυτό τα εξής:

```
<properties>
    <maven.compiler.source>1.7</maven.compiler.source>
    <maven.compiler.target>1.7</maven.compiler.target>
</properties>
```

Στη συνέχεια, μπαίνοντας στο φάκελο του project, εκτελούμε τις εντολές:

1. **mvn clean**
2. **mvn install**

οι οποίες συμπεριφέρονται ακριβώς με τον ίδιο τρόπο όπως παραπάνω.

Εκτέλεση: Για να εκτελεστεί το πρόγραμμα χρειάζεται να αλλάξουμε φάκελο, δηλαδή να μεταβούμε στο φάκελο "target" και στη συνέχεια στο φάκελο "classes", όπου βρίσκονται τα .class αρχεία. Έτσι, δίνοντας την εντολή "**sudo java Part1**", ξεκινά η εκτέλεση που τερματίζεται όταν δοθεί το σήμα Ctrl + C.

Περιγραφή Υλοποίησης

Package - Default

Part1: Η κλάση Part1 είναι η βασική κλάση του προγράμματος η οποία περιλαμβάνει τη συνάρτηση main ,από όπου και ξεκινά η εκτέλεση του προγράμματος. Η δημιουργία του Threadpool γίνεται στη συνάρτηση main, και την απαραίτητη πληροφορία τη βλέπει η κλάση Producer και τα υπόλοιπα νήματα μέσω ορισμάτων στους constructors τους.
Επίσης, η συνάρτηση main είναι υπεύθυνη να διαβάσει από το αρχείο "property_file" τυχόν ακέραιο αριθμό που αντιστοιχεί στον αριθμό των OneTimeJob νημάτων που δημιουργεί η κλάση Threadpool, καθώς και να ελέγξει εάν δίνεται όρισμα κατά την εκκίνηση, το οποίο είναι ο αριθμός των jobs που θα διαβάζονται περιοδικά από το αρχείο "input.dot". Στην περίπτωση που δε δοθεί κάποιος από αυτούς τους αριθμούς υπολογίζεται ψευδοτυχαία.

Package - software_agent

Threadpool: Η κλάση αυτή είναι υπεύθυνη για τη δημιουργία των OneTimeJob νημάτων, καθώς και του Sender νήματος, των διαμοιραζόμενων ουρών, και της λίστας όλων των νημάτων. Επίσης, είναι αυτή που "πιάνει" το σήμα που προέρχεται από το Ctrl + C και φροντίζει για τον ομαλό τερματισμό της εφαρμογής.

Η υλοποίηση του threadpool αποτελείται από :

1. μία διαμοιραζόμενη ουρά (**queue**) για την αποθήκευση των **NmapJob** από το producer νήμα και την εξαγωγή τους από τα consumer νήματα (**OnetimeJob**).

2. μία διαμοιραζόμενη ουρά (**results)** για την αποθήκευση των αποτελεσμάτων των **NmapJob** ύστερα από την εκτέλεση τους.

3. μία λίστα από **Thread** για να επιτυγχάνεται ο ομαλός τερματισμός των νημάτων (**cleanup**).

BlockingQueue: Για τις διαμοιραζόμενες ουρές έχει γίνει χρήση των πακέτων :

- **java.util.concurrent.BlockingQueue;**
- **java.util.concurrent.LinkedBlockingQueue;**

Η συγκεκριμένη δομή υποστηρίζει το μοντέλο **producer-consumer** σε ένα πολυνηματικό περιβάλλον. Παρέχει εσωτερικά locks για το συγχρονισμό των νημάτων και εντολές που είναι **thread-safe**. Πιό συγκεκριμένα :

put(): Η μέθοδος προσθέτει στοιχεία στην ουρά και μπλοκάρει το νήμα εαν συναντήσει τη δομή γεμάτη. Το νήμα ξεμπλοκάρει και συνεχίζει κανονικά όταν υπάρξει διαθέσιμος χώρος ξανά στην ουρά.

take(): Η μέθοδος αφαιρεί το πρώτο στοιχείο από την ουρά (εφόσον αυτή έχει στοιχεία) και **μπλοκάρει** το νήμα που την καλεί εαν η ουρα είναι άδεια. Επίσης η μέθοδος είναι **thread-safe**, δηλαδή ένα νήμα που χρησιμοποιεί τη μέθοδο αυτή, **μπλοκάρει** όλα τα υπόλοιπα νήματα που προσπαθούν να την καλέσουν για τη διάρκεια εκτέλεσης της. Με αυτόν τον τρόπο αποφεύγονται τυχόν **conflicts** και **race conditions** μεταξύ των νημάτων.

Για περισσότερες πληροφορίες μπορείτε να μεταβείτε στον εξής σύνδεσμο:

https://docs.orlace.com/javase/7/docs/api/java/util/concurrent/BlockingQueue.html

ShutdownHook: Για το τερματισμό των νημάτων με χρήση του ctrl + C , το threadpool περιέχει ενα shutdownhook το οποίο "πιάνει" το σήμα και καλεί τη μέθοδο **cleanup** για τον ομαλό τερματισμό όλων των νημάτων (Onetime & Periodic) μέσω **interrupt** και **join.**

Producer: Η κλάση Producer είναι υπεύθυνη να διαβάσει από το αρχείο το οποίο της δόθηκε ως όρισμα τα jobs και να τα κατανείμει. Συγκεκριμένα, εάν το job που διάβασε είναι περιοδικό, εκκινεί ένα νέο περιοδικό νήμα στο οποίο αναθέτει το job αυτό, ενώ στην περίπτωση που το αντίστοιχο flag είναι false (μη περιοδικό job) εισάγει το job αυτό στη διαμοιραζόμενη ουρά του threadpool, από την οποία εξάγουν τα OneTimeJob νήματα.
Δεδομένου ότι τα αποτελέσματα πρέπει να είναι σε μορφή XML, εξασφαλίζεται ότι η εντολή "nmap" περιλαμβάνει το flag "-oX -".

NmapJob: Η κλάση NmapJob κρατάει και διαχειρίζεται τις πληροφορίες των jobs που διαβάζονται από το αρχείο στην κλάση Producer.

OneTimeJob: Η κλάση OneTimeJob είναι η υλοποίηση των OneTimeJob νημάτων, καθένα από τα οποία διαβάζει από την ουρά ένα job, δημιουργεί την εντολή "nmap" ολοκληρωμένα, και την εκτελεί με τη χρήση της κλάσης Runtime που διαθέτει η Java. Προτού εισάγει το αποτέλεσμα στην ουρά των αποτελεσμάτων, από την οποία αντλεί η κλάση Sender, περιμένει να ολοκληρωθεί η εκτέλεση της εντολής.

Ως προς την κλάση Runtime, ισχύει πως κάθε εφαρμογή Java έχει ένα μόνο στιγμιότυπο της κλάσης Runtime που επιτρέπει στην εφαρμογή τη διεπαφή με το περιβάλλον στο οποίο η εφαρμογή εκτελείται. Το τρέχον runtime περιβάλλον μπορεί να αποκτηθεί μέσω της μεθόδου getRuntime. Η μέθοδος exec, εκτελεί σε ξεχωριστή διεργασία την εντολή που δέχεται ως όρισμα και επιστρέφει μία νέα διεργασία.

PeriodicJob: Η κλάση PeriodicJob είναι η υλοποίηση των περιοδικών νημάτων, καθένα από τα οποία αναλαμβάνει να εκτελεί περιοδικά το job που του ανατέθηκε. Συνεπώς, όπως και τα OneTimeJob νήματα, εκτελεί την εντολή "nmap" με τη χρήση της κλάσης Runtime που διαθέτει η Java. Ομοίως, προτού εισάγει το αποτέλεσμα στην ουρά των αποτελεσμάτων, από την οποία αντλεί η κλάση Sender, περιμένει να ολοκληρωθεί η εκτέλεση της εντολής. Έπειτα εισέρχεται σε non-runnable κατάσταση (μέσω sleep) για όσα δευτερόλεπτα αναφέρονται στο αρχείο και μόλις "ξυπνήσει" επαναλαμβάνει την ίδια διαδικασία.

Sender: Η κλάση Sender είναι η υλοποίηση του Sender νήματος, το οποίο διαβάζει περιοδικά από την ουρά αποτελεσμάτων τα διαθέσιμα ολοκληρωμένα αποτελέσματα και τα εκτυπώνει στην προεπιλεγμένη έξοδο. Όταν δεν υπάρχουν άλλα αποτελέσματα να εκτυπώσει, εισέρχεται σε non-runnable κατάσταση (μέσω sleep) για ψευδοτυχαία δευτερόλεπτα και μόλις "ξυπνήσει" επαναλαμβάνει την ίδια διαδικασία.

Όλα τα νήματα συνεχίζουν την εκτέλεσή τους με βάση τη συνθήκη:

```
while (!Thread.currentThread().isInterrupted())
```

η οποία επιτρέπει στα νήματα να εκτελούνται συνεχώς μέχρις ότου να δεχθούν ένα σήμα που να τα διακόπτει.
Το σήμα αυτό το λαμβάνουν μέσω της συνάρτησης cleanup, που διακόπτει όλα τα νήματα και το διαχειρίζονται με τη βοήθεια του catch InterruptedException.
Με αυτόν τον τρόπο, εξασφαλίζεται ο ομαλός τερματισμός όλων των νημάτων, ενώ θέτοντας τις δομές σε null, και καλώντας τον garbage collector, αποδεσμεύονται όλα άμεσα.