ΠΡΟΗΓΜΕΝΑ ΘΕΜΑΤΑ ΑΝΤΙΚΕΙΜΕΝΟΣΤΡΕΦΟΥΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

ΠΜΣ «Προηγμένα Συστήματα Πληροφορικής – Ανάπτυξη Λογισμικού και Τεχνητής Νοημοσύνης»

Απαλλακτική Εργασία Χειμερινού Εξαμήνου 2020-21.

Τίτλος Εργασίας: «Precedence Graph Threading using Blockchain»

15.03.2021

Μέλη ομάδας:

1. **Φιλιόπουλος Δημήτρης – Α.Μ: ΜΠΣΠ20054.**
2. **Ασημακόπουλος Γεώργιος – Α.Μ: ΜΠΣΠ20004.**

Προεργασία:

Για τη δημιουργία της εφαρμογής χρησιμοποιήθηκε το πρόγραμμα IntelliJ IDEA. Πριν την υλοποίηση του κώδικα έγιναν τα παρακάτω βήματα προεργασίας:

1. Μέσα στο φάκελο της εργασίας, δημιουργήσαμε ένα φάκελο input, όπου εκεί αποθηκεύσαμε τα δύο αρχεία κειμένου που θα δοθούν ως είσοδο στην εφαρμογή μας, έτσι ώστε να ληφθούν τα δεδομένα των νημάτων.

2. Η εργασία μας υλοποιήθηκε σε πρότζεκτ τύπου Maven, στο αρχείο pom.xml που δημιουργείται αυτόματα από το IntelliJ εισάγαμε τα dependencies για να λειτουργούν ορθά τα αρχεία JSON για την εκτύπωση του τελικού Block Chain, αλλά και για τη SQLite βάση δεδομένων (διάφορες λειτουργίες σε αυτήν).

Τα dependencies βρέθηκαν στις παρακάτω σελίδες:

- Gson: https://mvnrepository.com/artifact/com.google.code.gson/gson/2.8.6
- SQLite JDBC: https://mvnrepository.com/artifact/org.xerial/sqlite-jdbc/3.34.0

Παρουσίαση της εφαρμογής:

Η εφαρμογή εκτελεί μία προσομοίωση η οποία βασίζεται σε νήματα. Τα δεδομένα των νημάτων διαβάζονται από δύο αρχεία κειμένου(p_precedence.txt και p_timings.txt) τα οποία περιέχουν τα εν λόγω δεδομένα. Αρχικά, κατά την ανάγνωση των αρχείων δημιουργούνται αντικείμενα τύπου MyThread τα οποία κληρονομούν τη Thread και οι ιδιότητές τους είναι το όνομα, οι εξαρτήσεις και ο χρόνος εκτέλεσής τους και τοποθετούνται σε μία λίστα. Έπειτα, κάθε νήμα αυτής της λίστας εκτελείται με βάση τους χρόνους και τις εξαρτήσεις που έχει. Στη συνέχεια δημιουργούνται τα μπλοκ κάθε νήματος με τη σειρά που εκτελέστηκαν, στα οποία κάθε φορά εκχωρούνται τα δεδομένα των νημάτων. Αφού υπολογιστούν, δημιουργείται η συνολική αλυσίδα. Η αλυσίδα αυτή ελέγχεται αν είναι έγκυρη και μετά εκτυπώνεται στη κονσόλα μας ως JSON αρχείο. Τέλος, η αλυσίδα αποθηκεύεται στη τοπική βάση με όλα τα δεδομένα της, όπως αυτά αναγράφονται στην εκφώνηση της εργασίας.

Επεξήγηση κλάσεων & κώδικα εργασίας:

Θα δώσουμε κάποιες λεπτομέρειες για κάθε κλάση (αρχείο .java) της εργασίας μας και το πώς αυτή συμβάλλει στο τελικό αποτέλεσμα:

`Block.java`

Η κλάση αυτή κρατά πληροφορίες για ένα μοναδικό block στο ολικό block chain που θα δημιουργηθεί αφού εκτελεστεί το πρόγραμμα. Η κλάση κρατά δεδομένα για το ίδιο το block και τον ορθό υπολογισμό του αλλά και δεδομένα για μία προσομοίωση, όπως το όνομά της, το νήμα εκτέλεσης, τη διάρκειά του, τις εξαρτήσεις του από τα άλλα νήματα και τέλος τη χρονοσφραγίδα εκτέλεσης. Υλοποιείται με builder design pattern.

Οι μέθοδοί της κλάσης είναι:

- `String mineBlock()`: Η μέθοδος κάνει mine ένα block, επιστρέφει το hash του σε μορφή String, το οποίο είναι πεδίο της κλάσης αυτής.
- `String calculateBlockHash()`: Υπολογίζει το hash ενός block το οποίο αποθηκεύει δεδομένα από το αρχείο εισόδου των νημάτων.
- Η κλάση επίσης έχει getters μεθόδους για όλα της τα πεδία.

BlockChainController.java

Η κλάση χειρίζεται τη συνολική αλυσίδα από blocks που δημιουργείται κάθε φορά που τρέχουμε μία ακολουθία νημάτων.

Οι μέθοδοί της κλάσης είναι:

- `void createBlock(MyThread)`: Η μέθοδος δημιουργεί ένα μοναδικό block, το οποίο τελικά το κάνει mine και το προσθέτει στη συνολική αλυσίδα. Δέχεται ως όρισμα το νήμα, όπου λαμβάνει και τα δεδομένα του.
- `Boolean isChainValid(List blockChain)`: Η μέθοδος ελέγχει αν η συνολική αλυσίδα από block που δημιουργήθηκε, είναι έγκυρη. Κάνει τους απαραίτητους ελέγχους και επιστρέφει true ή false ανάλογα. Δέχεται ως όρισμα την αλυσίδα-λίστα από τα blocks.

DBConnection.java:

Η κλάση αυτή δημιουργεί τη σύνδεση με τη τοπική βάση δεδομένων που έχουμε για την αποθήκευση των δεδομένων της αλυσίδας από blocks. Ο κώδικας μας για να «επικοινωνήσει» με τη βάση χρειάζεται και ένα αρχείο τύπου SQLite το οποίο είναι αποθηκευμένο στο φάκελο της εργασίας μας, με όνομα: prohgmena_themata.db. Το αρχείο έχει δημιουργηθεί μέσω του DB Browser for SQLite και κρατά το πίνακα με τα πεδία του block όπως περιγράφηκαν παραπάνω.

Οι μέθοδοί της κλάσης είναι:

- `Connection connect()`: Κάνει τη σύνδεση με τη τοπική βάση, επιστρέφει το αντικείμενο της σύνδεσης αυτής.
- `void addBlockChain(List blockchain)`: Δέχεται ως όρισμα ολόκληρη την αλυσίδα των blocks. Εισάγει όλα τα δεδομένα της αλυσίδας στη τοπική βάση, υπάρχει βρόγχος για κάθε block. Υπάρχει έλεγχος για το αν η εισαγωγή των δεδομένων έγινε σωστά.
- `String getLastHash()`: Επιστρέφει το hash ως String, από το τελευταίο μπλοκ της βάσης μας.
- `String getSimulationName()`: Επιστρέφει ως String, το τυχαίο όνομα της προσομοίωσης η οποία τρέχει.
- `String calculateSimulationName(List simulationNames)`: Η μέθοδος αυτή παράγει κάθε φορά ένα νέο τυχαίο όνομα προσομοίωσης. Δέχεται ως όρισμα μία λίστα με όλα τα ονόματα προσομοιώσεων που υπάρχουν στη βάση και ελέγχει αν το νέο όνομα υπάρχει ήδη.

MyThread.java

Η κλάση αυτή υλοποίει και τρέχει το κάθε νήμα, με βάση τις ιδιότητες που του παρέχουμε κάθε φορά (όνομα, χρόνος αναμονής και εξαρτήσεις). Κρατά όλα αυτά τα δεδομένα στα private πεδία της, η κλάση έχει υλοποιηθεί με builder design pattern. Αφού δημιουργεί και τρέχει νήματα υποχρεωτικά πρέπει να κληρονομεί τα γνωρίσματα και τις μεθόδους της Thread κλάσης με extends.

Οι μέθοδοί της κλάσης είναι:

- `void run()`: Η μέθοδος αυτή γίνεται override αφού έχουμε κληρονομικότητα ως προς τη βασική κλάση Thread. Λαμβάνει τα δεδομένα του νήματος που τη καλεί κάθε φορά, ελέγχει αν το νήμα εξαρτάται από άλλα και αν υπάρχει χρόνος αναμονής για αυτό και το εκτελεί. Θέτει τη χρονοσφραγίδα του νήματος και αφού τελειώσει το αποθηκεύει σε μία public λίστα της κλάσης όπου κρατά τα ολοκληρωμένα νήματα.
- Η κλάση επίσης περιέχει μεθόδους getters & setters για τα πεδία της, τα οποία είναι private και πρέπει αναγκαστικά να προσπελαστούν από άλλες κλάσεις μέσω των μεθόδων αυτών.

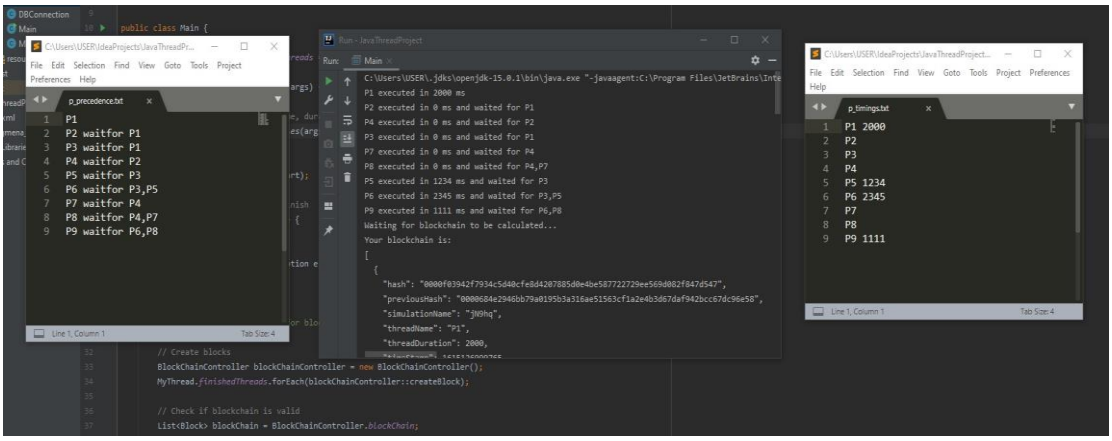
Main.java

Η κεντρική κλάση του προγράμματός μας, από την οποία ξεκινά η κάθε προσομοίωση. Οι μέθοδοί της κλάσης είναι οι:

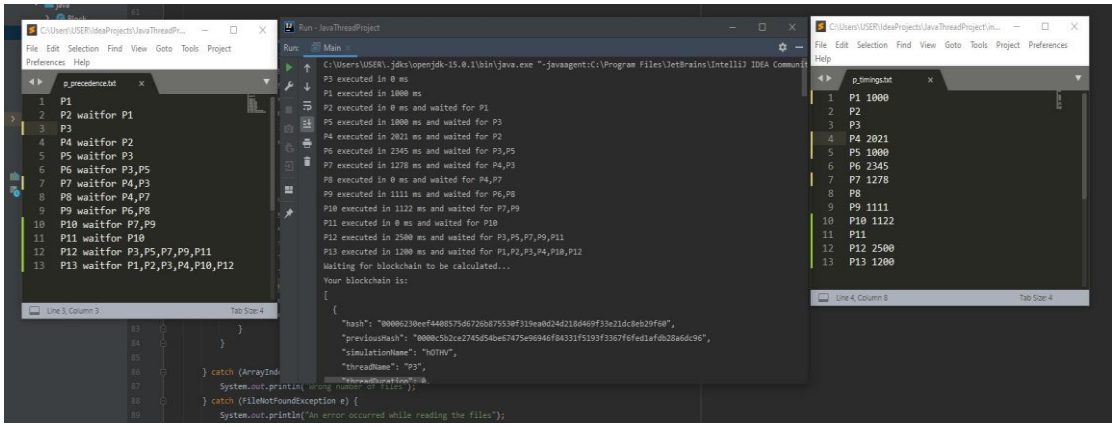
- `List getThreadsFromFiles(String[] args)`: Δέχεται ως όρισμα ένα πίνακα που περιέχει τα ονόματα των αρχείων κειμένου από τα οποία διαβάζουμε τα δεδομένα των νημάτων. Διαβάζει τα αρχεία αυτά γραμμή προς γραμμή και δημιουργεί το κάθε νήμα αναλόγως. Στο τέλος επιστρέφει μία λίστα με όλα τα νήματα που διάβασε και πρόσθεσε σε αυτήν.
- `void main(String[] args)`: Εδώ ξεκινά η εφαρμογή μας. Καλείται η παραπάνω μέθοδος, η οποία επιστρέφει μία λίστα νημάτων με τα δεδομένα τους. Εκτελεί τα νήματα και αφού τελειώσουν, δημιουργεί τα μπλοκ, ελέγχει αν η αλυσίδα με όλα τα μπλοκ είναι έγκυρη μέσα από μία σειρά ελέγχων σε αυτήν, τυπώνει στη κονσόλα της εφαρμογής την αλυσίδα σε μορφή JSON και τελικά την αποθηκεύει στη τοπική βάση.

Εικόνες από την εφαρμογή:

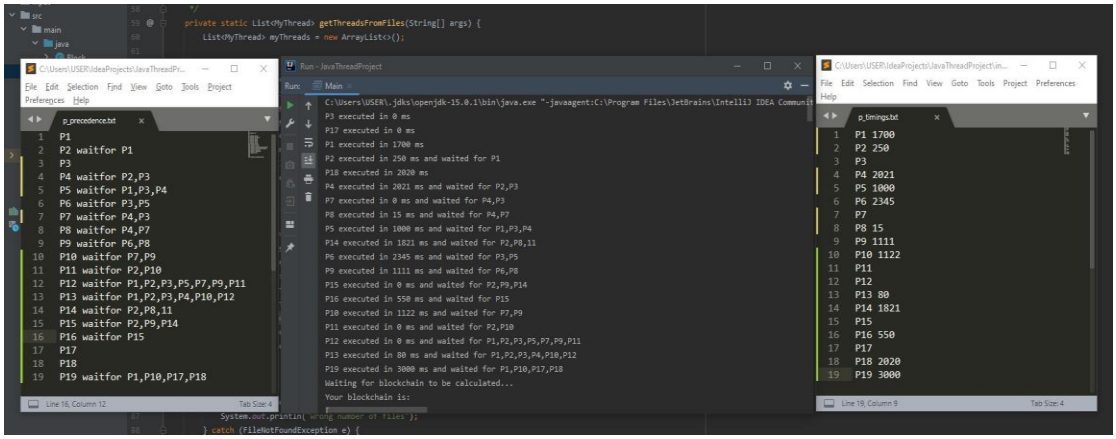
Παρακάτω θα παρουσιάσουμε κάποιες εικόνες εκτέλεσης από την εφαρμογή μας, κάθε εικόνα αποτελείται από τρία παράθυρα, το κεντρικό είναι το αποτέλεσμα εκτέλεσης από τη κονσόλα του IntelliJ, αριστερά και δεξιά υπάρχουν τα δύο αρχεία κειμένου όπου μας δείχνουν τα δεδομένα των νημάτων και της προσομοίωσης.



Εικόνα 1: Εκτέλεση σύμφωνα με τα δεδομένα που δίνονται στην εκφώνηση.



Εικόνα 2: Εκτέλεση με 13 νήματα με αντίστοιχες εξαρτήσεις & χρόνους αναμονής.



Εικόνα 3: Εκτέλεση με 19 νήματα με αντίστοιχες εξαρτήσεις & χρόνους αναμονής.

hash	previous_hash	simulation_name	thread_name	thread_duration	thread_dependencies	timestamp	nonce
0000063deaba1e82e0050bd783bb1f18796ad9dfc87d9e5b8afcb7ef8e9...	000059c0dbbd5286396eceeb5b75c689dc3db5ecdfd60f0d0a8beadac344635b5	x5j1l	P5	1000	[P1, P3, P4]	1615128398304	179044
00001ffee310d77ef94b0be52be7860355d8bbb74c6dbf4eb0013e9bab...	0000063deaba1e82e0050bd783bb1f18796ad9dfc87d9e5b8afcb7ef8e957e1d	x5j1l	P14	1821	[P2, P8, 11]	1615128399141	44175
000077e169f8e29736a9074c8f6e2e24b3e8fed95b2922f5b75cc04cdaf...	00001ffee310d77af94b0be52be7860355d8bbb74c6dbf4eb0013e9bab162406	x5j1l	P6	2345	[P3, P5]	1615128400650	72644
0000865715ca51f5bd5f2e814348ceeba5e3b1bf70d86e65383f596b1e6...	000077e169f8e29736a9074c8f6e2e24b3e8fed95b2922f5b75cc04cdaf582c9	x5j1l	P9	1111	[P6, P8]	1615128401761	138434
00004bde794e11317bfda63119d3abd2ef98405fe4203251542f49cd4cf...	0000865715ca51f5bd5f2e814348ceeba5e3b1bf70d86e65383f596b1e6ad333	x5j1l	P15	0	[P2, P9, P14]	1615128401761	51213
00000614e3314ea05422c1420c798f61915bff829aa270675f2a9d947ae8...	00004bde794e11317bfda63119d3abd2ef98405fe4203251542f49cd4cf51ad6	x5j1l	P16	550	[P15]	1615128402312	1918
000078981b8c4ff289a4a12007d77b7e9577b7d27f3e34e457e61976e7...	00000614e3314ea05422c1420c798f61915bff829aa270675f2a9d947ae87fd59	x5j1l	P10	1122	[P7, P9]	1615128402884	28407
00004ff167b13ea8693f8839dabd07531bf6908aa05db11c21c3f67ac03...	000078981b8c4ff289a4a12007d77b7e9577b7d27f3e34e457e61976e7b98e5f	x5j1l	P11	0	[P2, P10]	1615128402884	109075
00002a2c1bd92c57df6eb3e117eb298f02515cb5ace2cbcf7ec6e1ae998...	00004ff167b13ea8693f8839dabd07531bf6908aa05db11c21c3f67ac035dfac	x5j1l	P12	0	[P1, P2, P3, P5, P7, P9, P11]	1615128402884	12680
00007a4e010a84d59475750dd3123456d1f2fc3167067acf3d5c4a261b...	00002a2c1bd92c57df6eb3e117eb298f02515cb5ace2cbcf7ec6e1ae998885eb	x5j1l	P13	80	[P1, P2, P3, P4, P10, P12]	1615128402965	109114
00003ac9dafaeed2982148632101c77ccabcc0823262b618872c4a7dda...	00007a4e010a84d59475750dd3123456d1f2fc3167067acf3d5c4a261b8f3586	x5j1l	P19	3000	[P1, P10, P17, P18]	1615128405885	74483
00001f0c727f13baa94e6d63e8f4aa22c840e7aff0f74a584185e714382...	00003ac9dafaeed2982148632101c77ccabcc0823262b618872c4a7dda5c58e6	cet0i	P3	0	null	1615128450718	45988
00003f48e612e13beecd01e7825ceff2a40d9a0b42b5fb7c41d2446d927...	00001f0c727f13baa94e6d63e8f4aa22c840e7aff0f74a584185e714382d67a7	cet0i	P17	0	null	1615128450724	33711
000020f1485fb42c02d88dbfbf37c1b20ea8f078c23bcb08731e7964ddf1...	00003f48e612e13beecd01e7825ceff2a40d9a0b42b5fb7c41d2446d927493b3	cet0i	P1	1700	null	1615128452418	126383
0000733cbedafcb9f2584a88b8f70d24b56bdb92a0bf0ba07d2f937e96e...	000020f1485fb42c02d88dbfbf37c1b20ea8f078c23bcb08731e7964ddf16f57	cet0i	P2	250	[P1]	1615128452668	29379
0000915c8607fd96a3a71bbab04b5f15476eefef5caaeb768066fc2dfdea...	0000733cbedafcb9f2584a88b8f70d24b56bdb92a0bf0ba07d2f937e96e6da7a	cet0i	P18	2020	null	1615128452748	25160
0000e90621c54614c53f6f158b61b076b5ceab26513d670eee07441cd3...	0000915c8607fd96a3a71bbab04b5f15476eefef5caaeb768066fc2dfdea7cde	cet0i	P4	2021	[P2, P3]	1615128454690	18681
0000208f9a40fb53bc2abf2b610cb2402cf958bd169f06e93eff781880227e73	0000e90621c54614c53f6f158b61b076b5ceab26513d670eee07441cd3a4fa4d	cet0i	P7	0	[P4, P3]	1615128454690	138628
000073135aa0259a20c04831acfb2dd69e551740577a7a17657383925ec...	0000208f9a40fb53bc2abf2b610cb2402cf958bd169f06e93eff781880227e73	cet0i	P8	15	[P4, P7]	1615128454706	7564
0000574e90bb71c77923416fba0382f44ffdcc9405e31357449fa62760c...	000073135aa0259a20c04831acfb2dd69e551740577a7a17657383925ec649d345	cet0i	P5	1000	[P1, P3, P4]	1615128455691	51106
0000503c679d87dd8940b18a2e95f23c5872a7e49543037b91f7c8eac9...	0000574e90bb71c77923416fba0382f44ffdcc9405e31357449fa62760c4f521	cet0i	P14	1821	[P2, P8, 11]	1615128456529	56702
00007e5aab99f6a95651a0046c988c5353bb6fd5c427764e0d5c1fcc396...	0000503c679d87dd8940b18a2e95f23c5872a7e49543037b91f7c8eac918c234	cet0i	P6	2345	[P3, P5]	1615128458037	66008
000088b0c5ab48eb999f65d58ba1ac75153741eb9ceca5683990b0379d...	00007e5aab99f6a95651a0046c988c5353bb6fd5c427764e0d5c1fcc396f670e	cet0i	P9	1111	[P6, P8]	1615128459148	111218
0000cbff8fe52c509a2f159229cc2e7f2bd6a5f3ae093baf0cf8b09e23ad9...	000088b0c5ab48eb999f65d58ba1ac75153741eb9ceca5683990b0379d8a3abb	cet0i	P15	0	[P2, P9, P14]	1615128459148	129211
00009843e692cca0c95a69b7b4015b26fd8599502112 69a8d0dd42a7a...	0000cbff8fe52c509a2f159229cc2e7f2bd6a5f3ae093baf0cf8b09e23ad9773	cet0i	P16	550	[P15]	1615128459699	37593
0000e34137deda5ecda759caaef1dbeab681e8abba802277116cd5aae...	00009843e692cca0c95a69b7b4015b26fd85995021126 9a8d0dd42a7a4b448ee	cet0i	P10	1122	[P7, P9]	1615128460271	152384
00001ae0cf5ca39be94d693c2b24a6208518d3145cb5ad463b86788049a...	0000e34137deda5ecda759caaef1dbeab681e8abba802277116cd5aae897365c	cet0i	P11	0	[P2, P10]	1615128460271	49644
0000d8b29ea16d3310247e4ffbcd37cf6695adc7b3719d62b389daf2104...	00001ae0cf5ca39be94d693c2b24a6208518d3145cb5ad463b86788049a1a0dc	cet0i	P12	0	[P1, P2, P3, P5, P7, P9, P11]	1615128460271	72171
00004ef0de0cced120c240b05f1ba219a71cc024da2e4d8f554ee890ae7...	0000d8b29ea16d3310247e4ffbcd37cf6695adc7b3719d62b389daf2104b5637	cet0i	P13	80	[P1, P2, P3, P4, P10, P12]	1615128460352	3770
0000385b77abdc229c691b99a331cff17c6eaabfc01d676b10d9e771a48...	00004ef0de0cced120c240b05f1ba219a71cc024da2e4d8f554ee890ae76f876	cet0i	P19	3000	[P1, P10, P17, P18]	1615128463272	11296

Εικόνα 4: Τα καταγεγραμμένα στη βάση δεδομένα για κάθε προσομοίωση. Έγινε χρήση του προγράμματος DB Browser for SQLite.

Βίντεο εκτέλεσης:

Ο παρακάτω σύνδεσμος περιέχει το βίντεο εκτέλεσης της εφαρμογής μας, για δεδομένα εισόδου και ο τρόπος που τυπώνει τα δεδομένα στη κονσόλα:

https://drive.google.com/file/d/1Zz3zBKlaNt48KXsk400LWMlOlDu5LC-X/view?usp=sharing