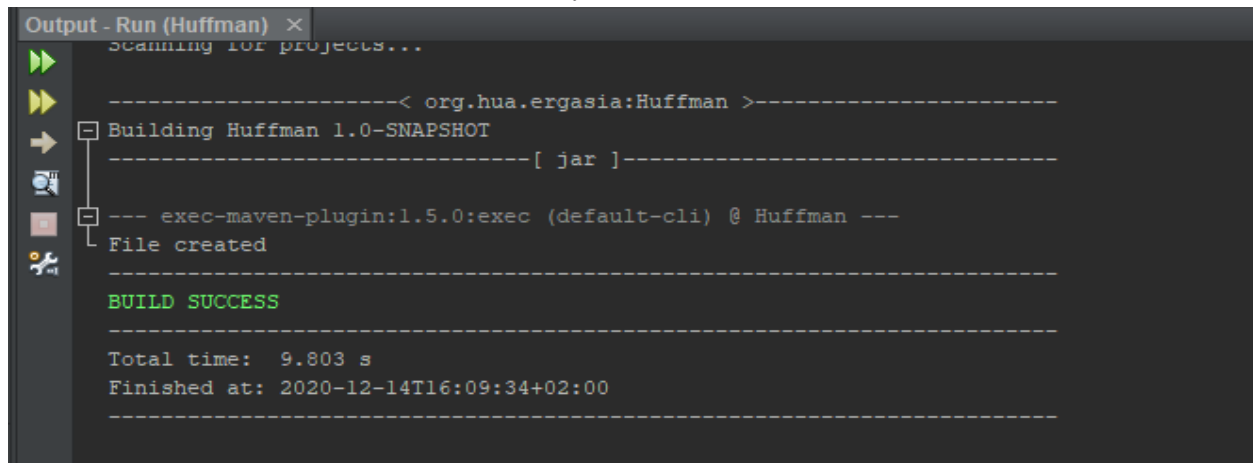


REPORT 1ου ΜΕΡΟΥΣ

ΤΡΟΠΟΣ ΣΥΓΓΡΑΦΗΣ ΕΡΓΑΣΙΑΣ: Η ομάδα δούλεψε συνεργατικά μέσω βιντεοκλήσης.

ΕΞΗΓΗΣΗ ΚΩΔΙΚΑ: Στο πρώτο μέρος υπάρχει η κλάση App η οποία περιέχει τη main(), η AsciiCounter που σκοπός της είναι η καταμέτρηση των χαρακτήρων ASCII και η κλάση FileCreator που είναι υπεύθυνη για τη δημιουργία του αρχείου(frequencies.dat). Αφού δημιουργηθούν τα αντικείμενα των παραπάνω κλάσεων (με τις ανάλογες παραμέτρους) μέσα στη main(), η AsciiCounter αρχίζει την διαδικασία καταμέτρησης των τριών αρχείων. Ορίζεται ο πίνακας που αποθηκεύει πόσες φορές εμφανίζεται κάθε χαρακτήρας στα δοσμένα αυτά αρχεία (η πρώτη θέση αντιστοιχεί στο πρώτο στοιχείο του πίνακα ASCII κλπ.) και ορίζουμε μια μεταβλητή που θα κρατά το πλήθος όλων των χαρακτήρων ascii. Στη συνέχεια, καλείται η FileCreator για να δημιουργήσει το αρχείο των συχνοτήτων (frequencies.dat). Έπειτα σε ένα πίνακα τύπου double εκχωρούμε τη συχνότητα διαιρώντας τον πίνακα των χαρακτήρων ascii με το συνολικό πλήθος τους αφού γίνει η στρογγυλοποίηση πρώτα του αριθμού μέσα από μια μέθοδο που καλείται τοπικά.

ΠΑΡΑΔΕΙΓΜΑ ΕΚΤΕΛΕΣΗΣ ΚΩΔΙΚΑ:



```
Output - Run (Huffman) X
Scanning for projects...
-----< org.hua.ergasia:Huffman >-----
Building Huffman 1.0-SNAPSHOT
-----[ jar ]-----
--- exec-maven-plugin:1.5.0:exec (default-cli) @ Huffman ---
File created
BUILD SUCCESS
Total time: 9.803 s
Finished at: 2020-12-14T16:09:34+02:00
```

Ενδεικτικά, κάποιες γραμμές του αρχείου frequencies.dat είναι οι εξής:

94 :	0.0
95 :	0.001
96 :	0.0
97 :	0.0582
98 :	0.0117
99 :	0.0174
100 :	0.0298
101 :	0.0927
102 :	0.0159
103 :	0.0153
104 :	0.0473
105 :	0.0489
106 :	8.0E-4
107 :	0.0057

Ascii Character:

Frequency:

REPORT 2ου ΜΕΡΟΥΣ

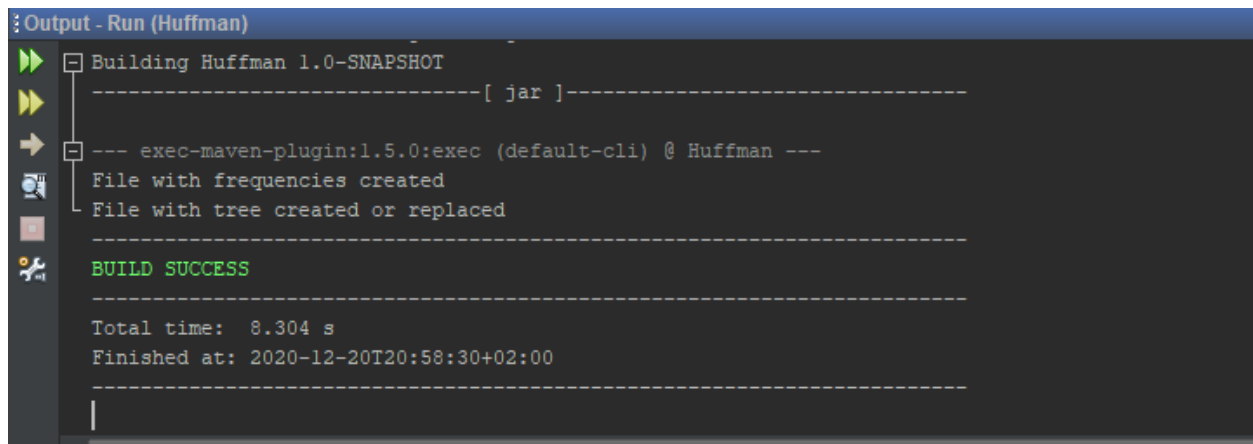
ΤΡΟΠΟΣ ΣΥΓΓΡΑΦΗΣ ΕΡΓΑΣΙΑΣ: Η ομάδα δούλεψε συνεργατικά μέσω βιντεοκλήσης.

ΕΞΗΓΗΣΗ ΚΩΔΙΚΑ: Στο δευτερο μέρος υλοποιήσαμε τις εξής κλάσεις:

Την FileRead η οποία διαβάζει το αρχείο frequencies.dat και τοποθετεί σε ένα πίνακα τύπου HuffmanNode αντικείμενα. Κάθε αντικείμενο της κλάσης HuffmanNode περιέχει τον χαρακτήρα ascii και τη συχνότητα του και τα δύο παιδιά τα οποία αρχικοποιούνται με null. Έπειτα αυτός ο πίνακας αποθηκεύεται σε ουρά προτεραιότητας (MinHeap) που φτιάξαμε στο εργαστήριο(χρησιμοποιήσαμε το interface MinHeap και την κλάση ArrayMinHeap). Έπειτα δημιουργήσαμε μια κλάση HuffmanTree η οποία υλοποιεί το δέντρο Huffman και με τη μέθοδο SaveTree αποθηκεύουμε το δέντρο ως serialized αντικείμενο στο αρχείο tree.dat

Η κλάση HuffmanNode υλοποιεί τις κλάσεις Serializable έτσι ώστε να μπορεί να αποθηκευτεί το δέντρο ως αντικείμενο και Comparable έτσι ώστε να μπορούν οι κομβοί να συγκριθούν μεταξύ τους

ΠΑΡΑΔΕΙΓΜΑ ΕΚΤΕΛΕΣΗΣ ΚΩΔΙΚΑ:



```
Output - Run (Huffman)
>> Building Huffman 1.0-SNAPSHOT
>> -----[ jar ]-----
> --- exec-maven-plugin:1.5.0:exec (default-cli) @ Huffman ---
> File with frequencies created
> File with tree created or replaced
> -----
> BUILD SUCCESS
> -----
> Total time: 8.304 s
> Finished at: 2020-12-20T20:58:30+02:00
> -----
> |
```

Παρατηρούμε ότι το πρόγραμμα αφού δημιουργήσει το αρχείο με τις συχνότητες δημιουργεί το αρχείο με το δέντρο Huffman

REPORT 3ου ΜΕΡΟΥΣ

ΤΡΟΠΟΣ ΣΥΓΓΡΑΦΗΣ ΕΡΓΑΣΙΑΣ: Η ομάδα δούλεψε συνεργατικά μέσω βιντεοκλήσης.

ΕΞΗΓΗΣΗ ΚΩΔΙΚΑ: Στο τρίτο μέρος υλοποιήσαμε την κλάση `TreeReader` η οποία διαβάζει το αρχείο `tree.dat` που περιέχει το δέντρο Huffman αποθηκευμένο ως `serialized` αντικείμενο. Χρησιμοποιώντας λοιπόν την ρίζα του δέντρου μέσα από την μέθοδο `CreateCodesFile()` δημιουργούμε το αρχείο με την κωδικοποίηση του δέντρου μας (`codes.dat`). Με σκοπο να διασχίσουμε το δέντρο χρησιμοποιούμε μια αναδρομική μέθοδο (`CodeGenerator`) η οποία διασχίζει το δέντρο και με την βοήθεια μια στοίβας(`ArrayDeque`) κρατάμε το μονοπάτι κάθε φύλλου που συναντάμε και το καταγράφουμε στο αρχείο "`codes.dat`".

ΠΑΡΑΔΕΙΓΜΑ ΕΚΤΕΛΕΣΗΣ ΚΩΔΙΚΑ:

```
--- exec-maven-plugin:1.5.0:exec (default-cli) @ Huffman ---
File with frequencies created
File with tree created or replaced
File with codes created
-----
BUILD SUCCESS
-----
Total time: 8.450 s
Finished at: 2021-01-02T17:46:31+02:00
-----
```

Παρατηρούμε ότι δημιουργείται το αρχείο "`codes.dat`" αφού διαβαστεί το αρχείο "`tree.dat`"

Ενδεικτικά, κάποιες γραμμές του αρχείου `codes.dat` είναι οι εξής:

```
Ascii character :      [Code-Path]
101:[0, 0, 0]
115:[0, 1, 0, 0]
104:[1, 1, 0, 0]
105:[0, 0, 1, 0]
110:[1, 0, 1, 0]
111:[0, 1, 1, 0]
63:[0, 0, 0, 0, 0, 0, 1, 1, 1, 0]
106:[1, 0, 0, 0, 0, 0, 1, 1, 1, 0]
113:[0, 1, 0, 0, 0, 0, 1, 1, 1, 0]
80:[1, 1, 0, 0, 0, 0, 1, 1, 1, 0]
53:[0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0]
86:[1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0]
```

REPORT 4ου ΜΕΡΟΥΣ

ΤΡΟΠΟΣ ΣΥΓΓΡΑΦΗΣ ΕΡΓΑΣΙΑΣ: Η ομάδα δούλεψε συνεργατικά μέσω βιντεοκλήσης.

ΕΞΗΓΗΣΗ ΚΩΔΙΚΑ: Στο τέταρτο μέρος υλοποιήσαμε την κλάση CodesRead() η οποία διαβάζει το αρχείο codes.dat που περιέχει την κωδικοποίηση Huffman για κάθε ascii character. Επίσης υλοποιήσαμε την FileCoder() η οποία παίρνει ως παραμέτρους τον πίνακα με την κωδικοποίηση των χαρακτήρων και το αρχείο που θέλει ο χρήστης να κωδικοποιήσει (πρώτη παράμετρος της main). Αφού γίνει η κωδικοποίηση σε bits τα οποία αποθηκεύονται με την βοήθεια της κλάσης BitSet() σε πίνακα από bytes, γίνεται αποθήκευση του πίνακα αυτού σε αρχείο που δημιουργείται με το όνομα που έβαλε ο χρήστης ως 2^η παράμετρο στη main. Χρειάστηκε επίσης να ρυθμίσουμε την main να δέχεται τις δυο παραμέτρους για input και output των αρχείων έτσι ώστε να γίνουν οι ανάλογες δοκιμές. Στην αρχή του αρχείου εξόδου επίσης αποθηκεύουμε το σύνολο των bits έτσι ώστε να μην διαβαστούν στο μέρος της αποκωδικοποίησης λάθος bits.

ΕΚΤΑΚΤΗ ΑΛΛΑΓΗ : Επεκτείναμε επίσης τους χαρακτήρες που διαβάζονται από τα αρχεία από ascii (128) σε extended ascii (256)

ΠΑΡΑΔΕΙΓΜΑ ΕΚΤΕΛΕΣΗΣ ΚΩΔΙΚΑ:

```
--- exec-maven-plugin:1.5.0:exec (default-cli) @ Huffman ---
File with frequencies already exists
File with tree created or replaced
File with codes already exists
File test.dat coded to file named testcoded.dat
-----
BUILD SUCCESS
-----
Total time: 9.658 s
Finished at: 2021-01-11T21:34:44+02:00
-----
```

Παρατηρούμε ότι το αρχείο test.dat που φτιάξαμε για δοκιμή κωδικοποιείται και αποθηκεύεται στο testcoded.dat

REPORT 5ου ΜΕΡΟΥΣ

ΤΡΟΠΟΣ ΣΥΓΓΡΑΦΗΣ ΕΡΓΑΣΙΑΣ: Η ομάδα δούλεψε συνεργατικά μέσω βιντεοκλήσης.

ΕΞΗΓΗΣΗ ΚΩΔΙΚΑ: Στο τέταρτο μέρος πλέον υπάρχουν 3 κλάσεις που περιέχουν την μέθοδο main() . Η κλάση App η οποία δημιουργεί το δέντρο Huffman και παράγοντας τα βασικά αρχεία (frequencies.dat, tree.dat ,codes.dat) , η κλάση Encoder η οποία δέχεται ένα αρχείο ascii ως παράμετρο και δημιουργεί το κωδικοποιημένο αρχείο με όνομα που δίνει ο χρήστης στη δεύτερη παράμετρο της main() και η κλάση Decoder η οποία δέχεται ένα κωδικοποιημένο αρχείο και το αποκωδικοποιεί σε ένα αρχείο με όνομα που ορίζει ο χρήστης στην δεύτερη παράμετρο της main(). Αφού διαβάσει το δέντρο που είναι αποθηκευμένο ως serialized αντικείμενο ("tree.dat") και τον πίνακα με bytes που είναι αποθηκευμένος στο κωδικοποιημένο αρχείο(τον οποίο περνάμε σε ένα bitset()) αρχίζει η προσπέλαση του δέντρου (διαβάζουμε επίσης από το κωδικοποιημένο αρχείο τον αριθμό των bits για να παραλείψουμε τα άχρηστα επιπλέον bits). Με την χρήση λοιπόν του bitset ξεκινώντας από τη ρίζα όπου συναντάμε 0 πάμε αριστερά και όπου 1 πάμε δεξιά ώπου να φτάσουμε σε φύλλο. Όταν συναντάμε φύλλο με την χρήση της κλάσης FileWriter() καταγράφουμε στο νέο αρχείο τον χαρακτήρα που αντιστοιχεί στο κάθε μονοπάτι του δέντρου και ξαναρχίζουμε την προσπέλαση από ρίζα για τον επόμενο χαρακτήρα.

ΠΡΟΒΛΗΜΑ ΑΠΟΚΩΔΙΚΟΠΟΙΗΣΗΣ :

Δυστυχώς η αποκωδικοποίηση δεν λειτουργεί σωστά υπάρχει κάποιο θέμα πιθανότατα στην ανάγνωση του δέντρου με αποτέλεσμα να εμφανίζονται διαφορετικοί χαρακτήρες από ότι θα έπρεπε στο αποκωδικοποιημένο αρχείο. Μετά από αρκετό ψάξιμο δυστυχώς δεν βρέθηκε κάποια λύση . Υπάρχουν στο συμπίεσμένο αρχείο μαζί με το project αρχεία που δοκιμάστηκαν στον κωδικοποιητή και στον ακωδικοποιητή.

ΠΑΡΑΔΕΙΓΜΑ ΕΚΤΕΛΕΣΗΣ ΚΩΔΙΚΑ (Decoder):

```
File test_encoded.dat decoded to file named test_decoded.dat

Process finished with exit code 0
```

Δοκιμαστικά αρχεία (Περιέχονται μέσα στο project) :

```
test.dat
test_decoded.dat
test_encoded.dat
```