

ΕΠΕΞΗΓΗΣΗ ΚΩΔΙΚΑ:

Αρχικά δημιουργήσαμε τις κλάσεις που θα χρησιμοποιήσουμε στον κώδικα μας την Main, HuffmanNode , Print, Files.

Στην HuffmanNode δηλώσαμε δύο μεταβλητές την data για τις συχνότητες και την c για τους χαρακτήρες από 0 μέχρι 127 επίσης δηλώσαμε το left και right ως δυο pointers.

Στην MyComparator συγκρίνουμε με βάση τις τιμές δεδομένων των κόμβων.

Στην TreeMaker δημιουργούμε μια function με όνομα create που παίρνει ως παραμέτρους τον πίνακα charfreq για τις συχνότητες των χαρακτήρων ένα charArray για τους χαρακτήρες και ένα n για το μέγεθος των πινάκων. Μετά δημιουργούμε μια ουρά προτεραιότητας και με την βοήθεια της for γεμίζουμε τα Nodes με τους χαρακτήρες και τις συχνότητες τους και τους βάζουμε σε ουρά(queue)

Στη συνέχεια δημιουργούμε ένα root node με τιμή null.

Μετά κάνουμε ένα while loop που παίρνει τα δύο μικρότερα Nodes από την ουρά μέχρι το size της να γίνει 1. Μέσα στο while παίρνουμε την πρώτη τιμή της ουράς με το peek και την καταχωρούμε στην μεταβλητή x και με το poll διαγράφουμε την τιμή από την ουρά μετά παίρνουμε την δεύτερη τιμή της ουράς και κάνουμε το ίδιο για την μεταβλητή y. Στη συνέχεια δηλώνουμε ένα Node f στο οποίο αναθέτουμε την

πρόσθεση της συχνότητας των 2 Node x και y . Μετά ορίζουμε το πρώτο Node ως το αριστερό παιδί και το δεύτερο ως το δεξί παιδί. Ύστερα αναθέτουμε την τιμή του root ως το f και χρησιμοποιώντας την add προσθέτουμε την Node f στην ουρά προτεραιότητας . Τέλος επιστρέφουμε την τιμή root.

Στην κλάση Files δημιουργήσαμε μια μέθοδο Read File που παίρνει παράμετρο ReadName για το path του αρχείου, ένα πίνακα ακεραίων για τις συχνότητες και ένα πίνακα χαρακτήρων για τους χαρακτήρες ascii. Μετά χρησιμοποιήσαμε την εντολή BufferedReader σε συνδυασμό με το FileReader για να μπορέσουμε να διαβάσουμε το δέντρο από το αρχείο.

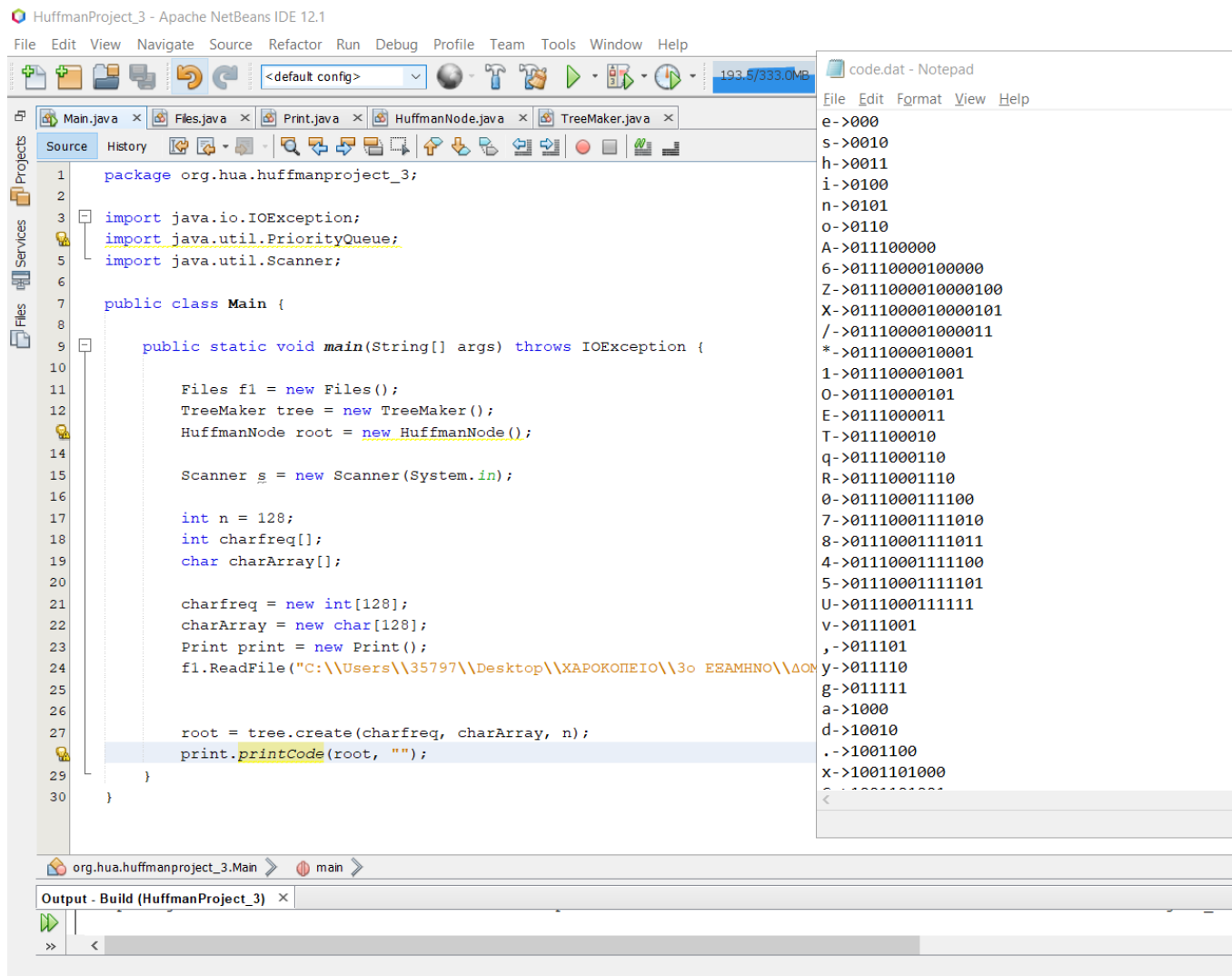
Συνεχίζοντας χρησιμοποιήσαμε ένα while loop .Μέσα στο loop διαβάζουμε τους χαρακτήρες και τις συχνότητες από το αρχείο με την βοήθεια του indexOf και substring(indexOf: βρίσκει την θέση μετά από ένα συγκεκριμένο string που έχουμε δήλωση, substring: παίρνει ότι υπάρχει μετά από την θέση του indexOf) και τις αποθηκεύουμε στους πίνακες array και asciitable χρησιμοποιώντας το Character.toString(μετατρέπει το char σε string) και το Integer.valueOf(μετατρέπει το string σε integer).

Μετά δημιουργήσαμε μια μέθοδο writeToFile που παίρνει ως παραμέτρους ένα χαρακτήρα και μια συμβολοσειρά για να γράψουμε το χαρακτήρα και την κωδικοποίησή του.

Στην κλάση Print δημιουργήσαμε μια function printcode που παίρνει τους παραμέτρους ένα node και μια συμβολοσειρά δηλώσαμε ένα constructor για να μπορούμε να χρησιμοποιήσουμε την κλάση files και κάναμε ένα if για να ελέγξουμε εάν στο αριστερό και δεξιό παιδί η τιμή είναι null, εάν είναι και οι δυο null τυπώνουμε στο αρχείο codes.dat την κωδικοποίηση του χαρακτήρα. Εάν δεν είναι null τότε αυτοκαλείται και αποθηκεύει στο string το string και 0 ή 1.

Στην Main δηλώσαμε ένα πίνακα charArray του οποίου αρχικοποιήσαμε τις θέσεις του με χαρακτήρες ascii από 0 έως και 127. Μετα δηλώσαμε ένα πίνακα charfreq με 128 θέσεις στον οποίο αποθηκεύσαμε τις συχνότητες των χαρακτήρων με την βοήθεια της κλάσης files. Δηλώσαμε το root στο οποίο αποθηκεύσουμε την τιμή που επιστρέφει η create και τέλος καλούμε την printcode για να τυπώσουμε την κωδικοποίηση των χαρακτήρων.

ΠΑΡΑΔΕΙΓΜΑΤΑ ΕΚΤΕΛΕΣΗΣ:



The screenshot displays the Apache NetBeans IDE interface for a project named 'HuffmanProject_3'. The main editor window shows the source code of the 'Main' class, which implements a Huffman tree algorithm. The code includes imports for 'IOException', 'PriorityQueue', and 'Scanner', and defines a 'main' method that reads a file, builds a Huffman tree, and prints the resulting binary codes for each character.

```
1 package org.hua.huffmanproject_3;
2
3 import java.io.IOException;
4 import java.util.PriorityQueue;
5 import java.util.Scanner;
6
7 public class Main {
8
9     public static void main(String[] args) throws IOException {
10
11         Files fl = new Files();
12         TreeMaker tree = new TreeMaker();
13         HuffmanNode root = new HuffmanNode();
14
15         Scanner s = new Scanner(System.in);
16
17         int n = 128;
18         int charfreq[];
19         char charArray[];
20
21         charfreq = new int[128];
22         charArray = new char[128];
23         Print print = new Print();
24         fl.ReadFile("C:\\Users\\35797\\Desktop\\ΧΑΡΟΚΟΠΕΙΟ\\3ο ΕΞΑΜΗΝΟ\\ΔΟΜ");
25
26         root = tree.create(charfreq, charArray, n);
27         print.printCode(root, "");
28     }
29 }
30
```

The 'Output - Build (HuffmanProject_3)' window at the bottom shows the execution results, displaying the binary codes for each character from 'e' to 'x'.

```
e->000
s->0010
h->0011
i->0100
n->0101
o->0110
A->01110000
6->01110000100000
Z->0111000010000100
X->0111000010000101
/->011100001000011
*->0111000010001
1->011100001001
O->01110000101
E->0111000011
T->011100010
q->0111000110
R->01110001110
0->0111000111100
7->01110001111010
8->01110001111011
4->01110001111100
5->01110001111101
U->0111000111111
v->0111001
,->011101
y->011110
g->011111
a->1000
d->10010
.->1001100
x->1001101000
```

Created from: Μυριάνθη Αποστολίδη , Χάρη Παπαμιχαήλ, Φώτη Γαλανό