Data Mining

Midterm Project

Subramanian Suryanarayanan

[ss3635@njit.edu](mailto:ss3635@njit.edu)

Contents:

1. Source Code

2. Input File Screenshots

3. Output Screenshots

1. Source Code:

Apriori.java:

**import** java.io.\*;

**import** java.util.\*;

**import** java.util.Map.\*;

**public** **class** Apriori {

//Concatenating each item specified in the data to the HashMap

**public** **static** **boolean** additems(HashMap < String, Integer > items, String itemset\_arr[], **double** transactions, **double** support, TreeSet < String > itemset, HashMap < String, ArrayList < Integer >> find\_item, **int** addindex) {

String strarray[] = **new** String[itemset.size()];

itemset.toArray(strarray);

itemset.removeAll(itemset);

items.clear();

**if** (addindex == 0) {

String addelement = "";

**for** (**int** i = 0; i < strarray.length; i++) {

**for** (**int** j = i + 1; j < strarray.length; j++) {

addelement = strarray[i] + "," + strarray[j];

itemset.add(addelement);

items.put(addelement, 2);

}

}

} **else** {

**int** ctr = 0;

HashMap < Integer, String > strmap = **new** HashMap < Integer, String > ();

TreeSet < String > strset = **new** TreeSet < String > ();

String addelement = "";

**for** (**int** i = 0; i < strarray.length; i++) {

addelement = strarray[i] + ",";

**for** (**int** j = i + 1; j < strarray.length; j++) {

addelement += strarray[j];

strmap.put(ctr, addelement);

addelement = strarray[i] + ",";

ctr++;

}

}

**for** (**int** i = 0; i < strmap.size(); i++) {

String line = strmap.get(i);

String[] lineSplit = line.split(",");

**for** (**int** j = 0; j < lineSplit.length; j++) {

strset.add(lineSplit[j]);

}

**if** (strset.size() == addindex + 2) {

String str\_arr[] = **new** String[strset.size()];

strset.toArray(str\_arr);

String extra = "";

**for** (**int** k = 0; k < strset.size(); k++) {

extra += str\_arr[k] + ",";

}

strset.removeAll(strset);

extra = extra.substring(0, extra.length() - 1);

itemset.add(extra);

items.put(extra, 2);

} **else**

strset.removeAll(strset);

}

}

**if** (items.size() == 0)

**return** **false**;

**else**

**return** **true**;

}

//Function to calculate Support value of each Item in the specified data

**public** **static** **void** calculatesupport(HashMap < String, Integer > items, String addarr[], **double** transactions, **double** support, TreeSet < String > itemset) {

**for** (**int** i = 0; i < addarr.length; i++) {

**if** ((items.get(addarr[i]) / transactions) < support) {

items.remove(addarr[i]);

itemset.remove(addarr[i]);

}

}

}

//Function to scan the items in the given data

**public** **static** **void** itemscan(HashMap < String, Integer > items, String addarr[], **double** transactions, **double** support, TreeSet < String > itemset, HashMap < String, ArrayList < Integer >> find\_item) {

String[] split\_line = **null**;

**for** (**int** i = 0; i < addarr.length; i++) {

split\_line = addarr[i].split(",");

ObjList obj[] = **new** ObjList[split\_line.length];

**for** (**int** j = 0; j < split\_line.length; j++) {

obj[j] = **new** ObjList();

obj[j].setListVariable(find\_item.get(split\_line[j]));

}

**for** (**int** k = 1; k < obj.length; k++) {

obj[0].objlist.retainAll(obj[k].objlist);

}

items.put(addarr[i], obj[0].objlist.size());

}

}

//Function to extract input from the file

**public** **static** **void** Inputfilecheck(**double** supp\_val, **double** con\_val) {

**double** transactions = 0;

String read\_line;

HashMap < String, Integer > items = **new** HashMap < String, Integer > ();

TreeSet < String > itemset = **new** TreeSet < String > ();

HashMap < String, ArrayList < Integer >> find\_item = **new** HashMap < String, ArrayList < Integer >> ();

TreeSet < String > associations = **new** TreeSet < String > ();

HashMap < String, Integer > frequentitem = **new** HashMap < String, Integer > ();

**int** ctr = 1;

BufferedReader bf = **null**;

**try** {

bf = **new** BufferedReader(**new** FileReader("D:\\amazon.txt"));

**try** {

**while** ((read\_line = bf.readLine()) != **null**) {

String[] split = read\_line.split(",");

**for** (**int** i = 0; i < split.length; i++) {

**if** (items.containsKey(split[i])) {

items.put(split[i], items.get(split[i]) + 1);

ArrayList < Integer > arraylist = **new** ArrayList < Integer > ();

arraylist = find\_item.get(split[i]);

arraylist.add(ctr);

find\_item.put(split[i], arraylist);

} **else** {

items.put(split[i], 1);

ArrayList < Integer > arraylist = **new** ArrayList < Integer > ();

arraylist.add(ctr);

find\_item.put(split[i], arraylist);

itemset.add(split[i]);

}

}

ctr++;

transactions++;

}

String itemset\_arr[] = **new** String[itemset.size()];

itemset.toArray(itemset\_arr);

*calculatesupport*(items, itemset\_arr, transactions, supp\_val, itemset);

frequentitem.putAll(items);

**boolean** return\_value = **true**;

**int** addindex = 0;

**while** (return\_value == **true**) {

return\_value = *additems*(items, itemset\_arr, transactions, supp\_val, itemset, find\_item, addindex);

String addarr[] = **new** String[itemset.size()];

itemset.toArray(addarr);

*itemscan*(items, addarr, transactions, supp\_val, itemset, find\_item);

*calculatesupport*(items, addarr, transactions, supp\_val, itemset);

frequentitem.putAll(items);

addindex++;

}

System.***out***.println();

System.***out***.println("Frequent Items:");

System.***out***.println();

**for** (Entry < String, Integer > entry: frequentitem.entrySet()) {

itemset.add(entry.getKey());

**double** val = 0;

val = (**double**)(entry.getValue()) / transactions;

System.***out***.println("ITEMSET: " + entry.getKey() + " FREQUENCY VALUE : " + entry.getValue() + " SUPPORT VALUE : " + val);

};

String add\_arr[] = **new** String[itemset.size()];

itemset.toArray(add\_arr);

String association\_arr[] = **new** String[itemset.size()];

itemset.toArray(association\_arr);

System.***out***.println();

HashMap < String, Integer > association\_rules = **new** HashMap < String, Integer > ();

**for** (**int** i = 0; i < itemset.size(); i++) {

String[] split\_line = add\_arr[i].split(",");

**if** (split\_line.length > 1) //Association rules cannot be created for individual item sets

{

**for** (**int** j = 0; j < split\_line.length; j++) {

**for** (**int** k = j + 1; k < split\_line.length; k++) {

association\_rules.put(split\_line[j] + "," + split\_line[k], 2);

association\_rules.put(split\_line[j], 2);

association\_rules.put(split\_line[k], 2);

}

}

association\_rules.remove(association\_arr[i]);

String string1 = association\_arr[i];

**for** (Entry < String, Integer > entry: association\_rules.entrySet()) {

String string2 = entry.getKey();

String string3 = entry.getKey() + ",";

String string4 = "," + entry.getKey();

String string5 = "," + entry.getKey() + ",";

**if** (string1.toLowerCase().contains(string5.toLowerCase())) {

string1 = string1.replace(string5, ",");

} **else** **if** (string1.toLowerCase().contains(string3.toLowerCase())) {

string1 = string1.replace(string3, "");

} **else** **if** (string1.toLowerCase().contains(string4.toLowerCase())) {

string1 = string1.replace(string4, "");

} **else** **if** (string1.toLowerCase().contains(string2.toLowerCase())) {

string1 = string1.replace(string2, "");

} **else** {

String[] comma\_splitter = string2.split(",");

**for** (**int** m = 0; m < comma\_splitter.length; m++) {

**if** (string1.contains("," + comma\_splitter[m] + ",")) {

string1 = string1.replace("," + comma\_splitter[m] + ",", ",");

} **else** **if** (string1.contains(comma\_splitter[m] + ",")) {

string1 = string1.replace(comma\_splitter[m] + ",", "");

} **else** **if** (string1.contains("," + comma\_splitter[m])) {

string1 = string1.replace("," + comma\_splitter[m], "");

} **else** **if** (string1.contains(comma\_splitter[m])) {

string1 = string1.replace(comma\_splitter[m], "");

} **else** {

System.***out***.println("Kindly check the specified Input! " + association\_arr[i]);

}

}

}

**if** (string1.length() == 0) {

System.***out***.println("Empty String!");

}

**double** supportcount\_item = frequentitem.get(association\_arr[i]);

**double** supportcount\_set = frequentitem.get(entry.getKey());

**if** ((supportcount\_item / supportcount\_set) >= con\_val) {

associations.add(entry.getKey() + " ---> " + string1);

}

string1 = association\_arr[i];

};

}

association\_rules.clear();

}

System.***out***.println("Association Rules:");

System.***out***.println();

String allrules[] = **new** String[associations.size()];

associations.toArray(allrules);

**for** (**int** i = 0; i < allrules.length; i++) {

System.***out***.println(allrules[i]);

}

} **catch** (IOException e) {

System.***out***.println("Error in Reading the File!");

System.*exit*(0);

}

} **catch** (FileNotFoundException fe) {

System.***out***.println("File Name is not valid!");

System.*exit*(0);

}

}

//Main Function Declaration

**public** **static** **void** main(String args[]) {

**double** supp\_val = **new** Double(args[0]);

**double** con\_val = **new** Double(args[1]);

System.***out***.println("Minimum Support = " + supp\_val + "%");

System.***out***.println("Minimum Confidence = " + con\_val + "%");

supp\_val = supp\_val / 100;

con\_val = con\_val / 100;

*Inputfilecheck*(supp\_val, con\_val);

System.***out***.println();

}

}

ObjList.java:

**import** java.util.ArrayList;

**public** **class** ObjList

{

**public** ArrayList <Integer>objlist;

**public** **void** setListVariable(ArrayList<Integer> temp)

{

objlist = **new** ArrayList<Integer>();

**for**(**int** i=0;i<temp.size();i++)

{

objlist.add(temp.get(i));

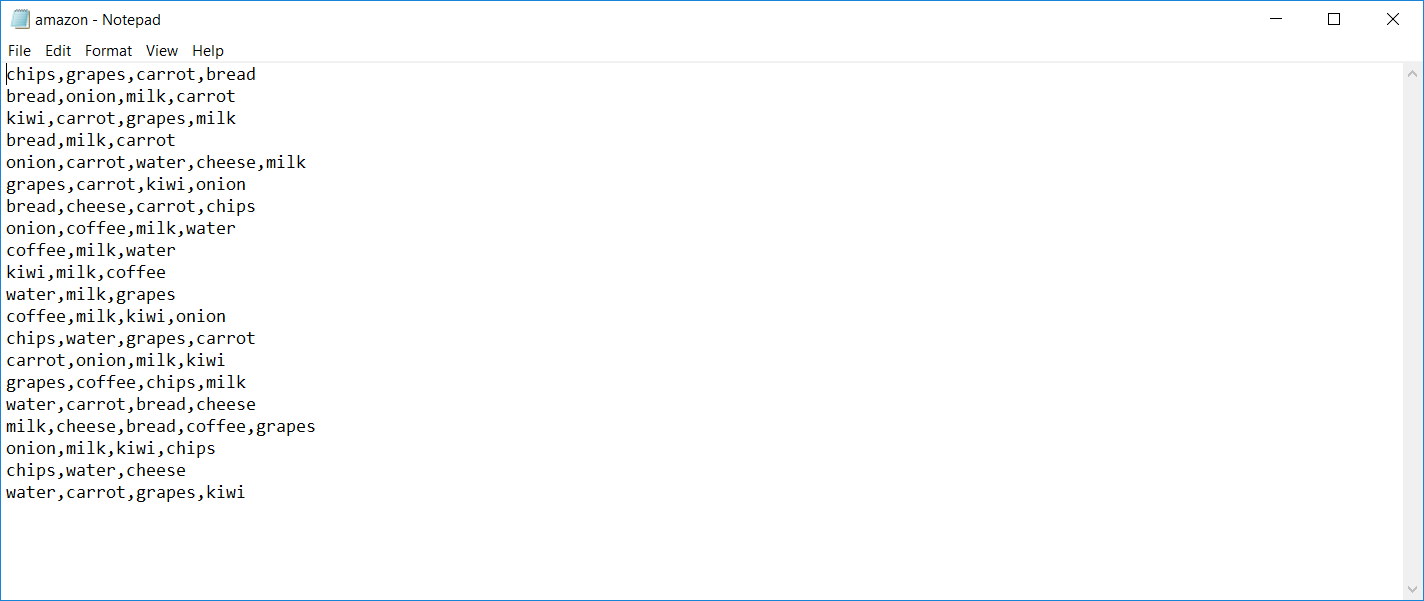
}

}

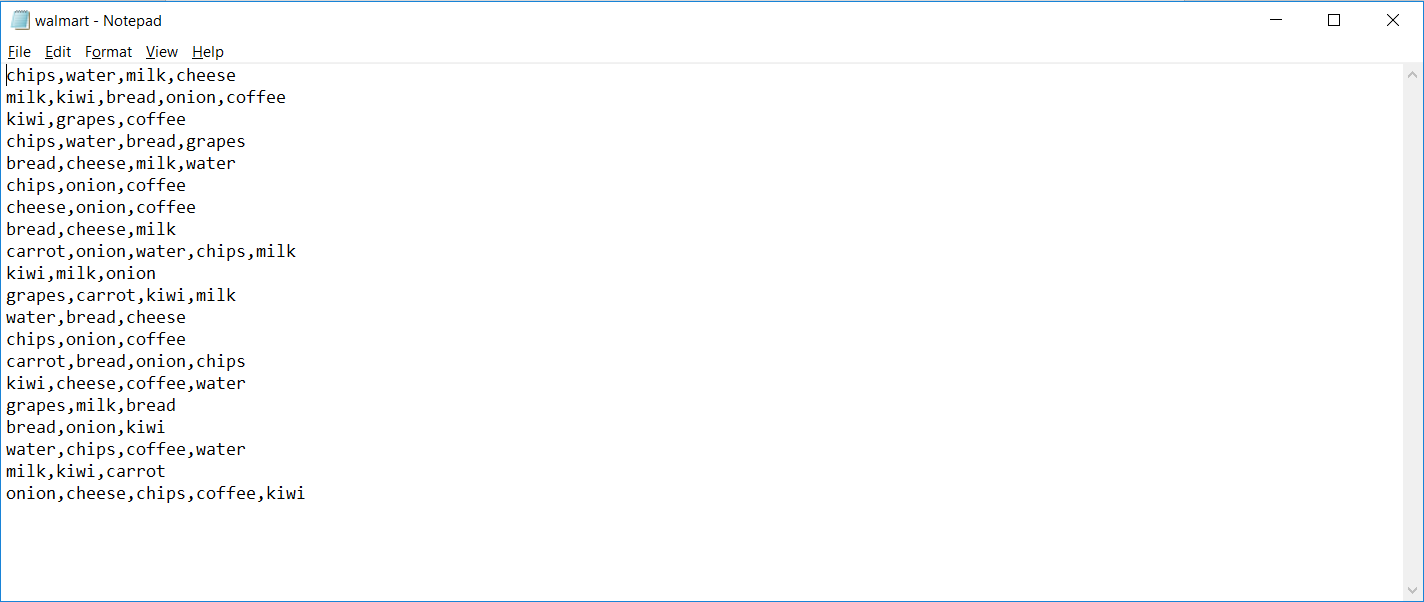
}

2. Input File Screenshots:

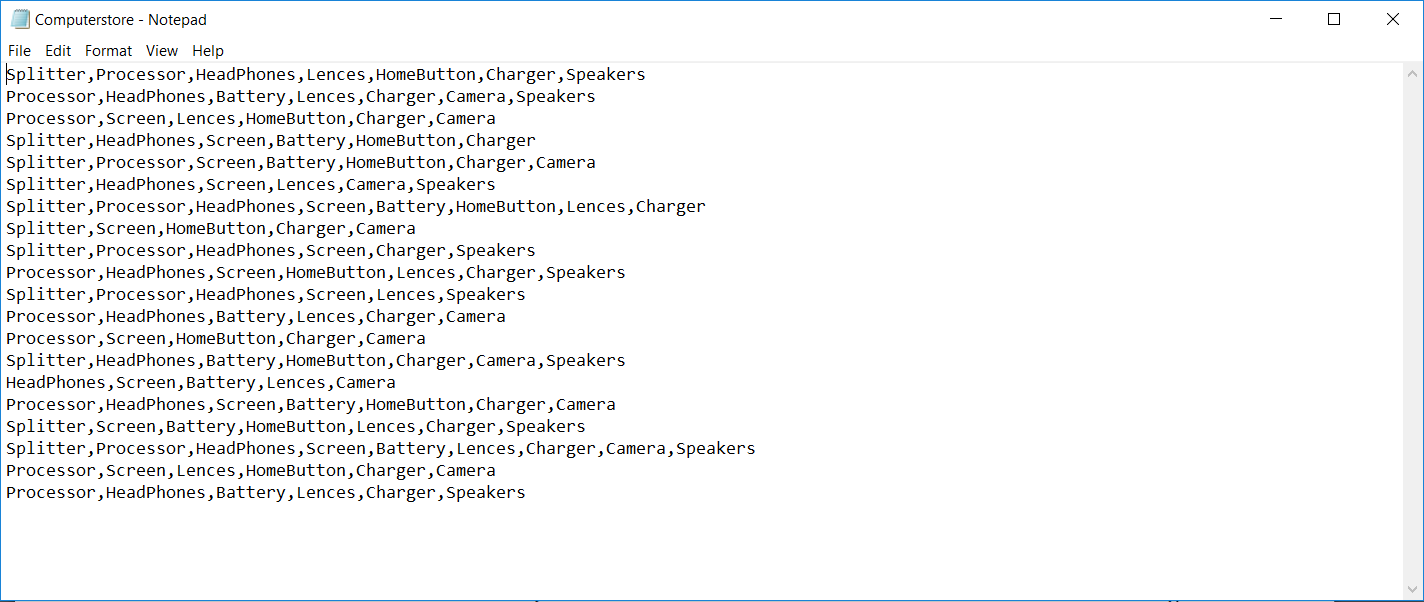
Amazon.txt:



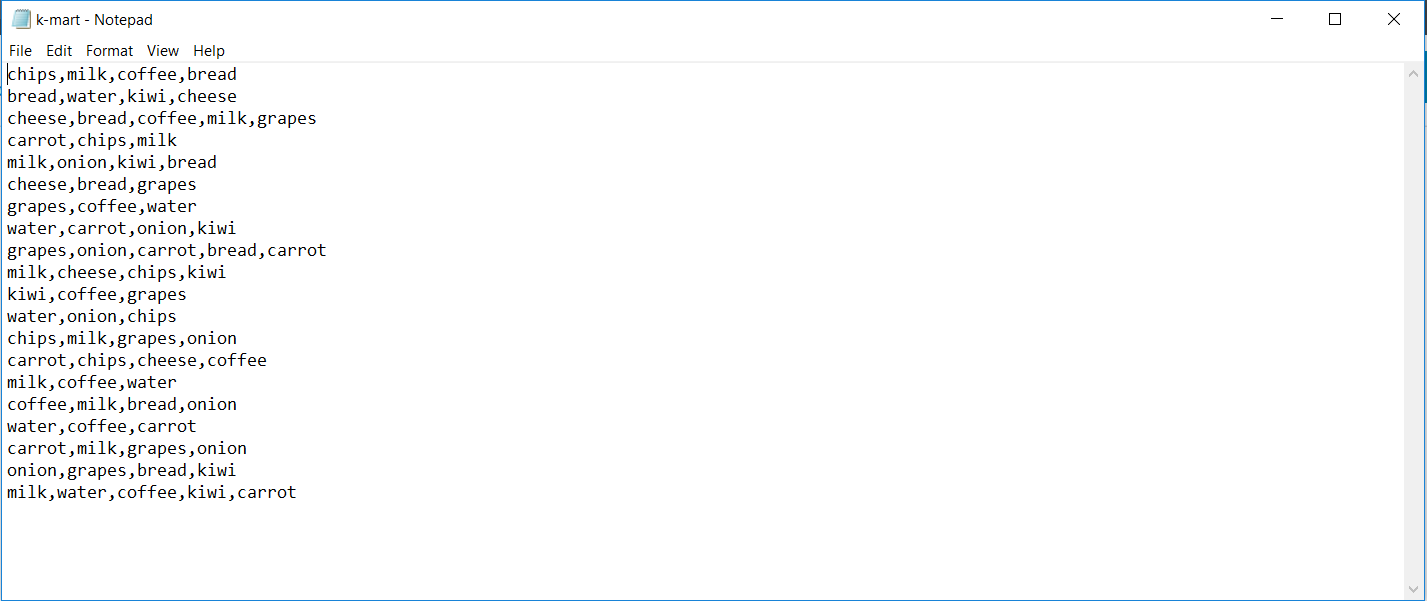
Walmart.txt:



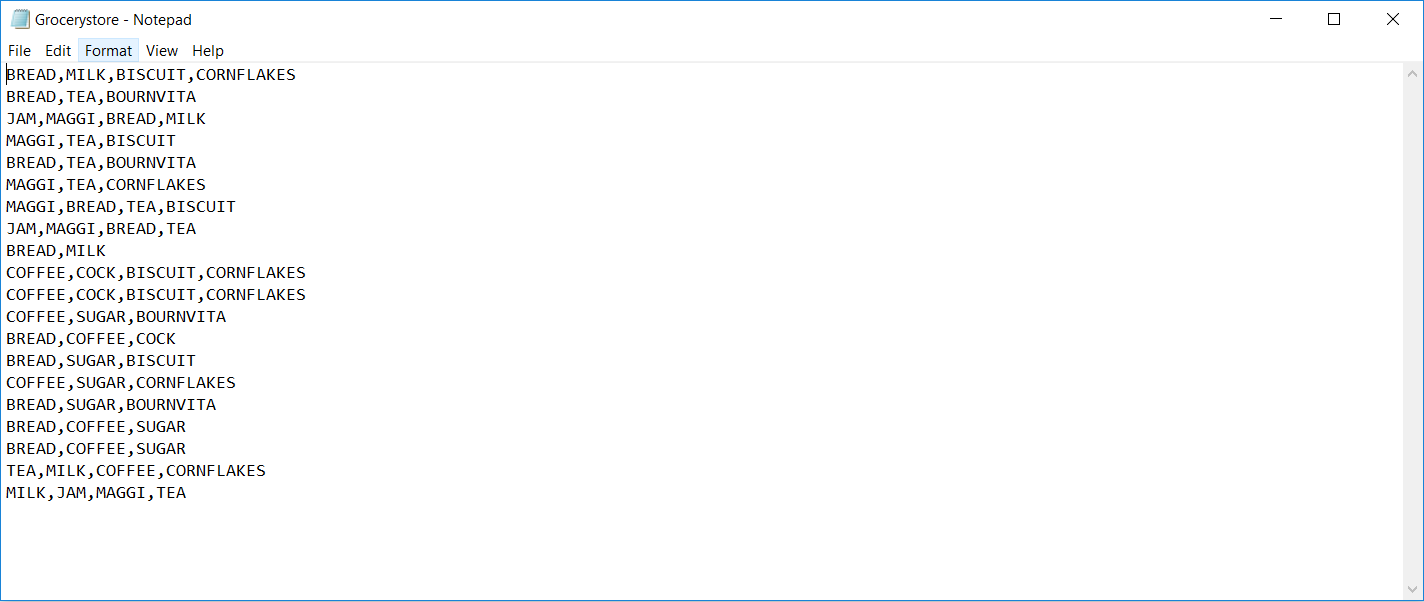
Computerstore.txt:



K-Mart.txt

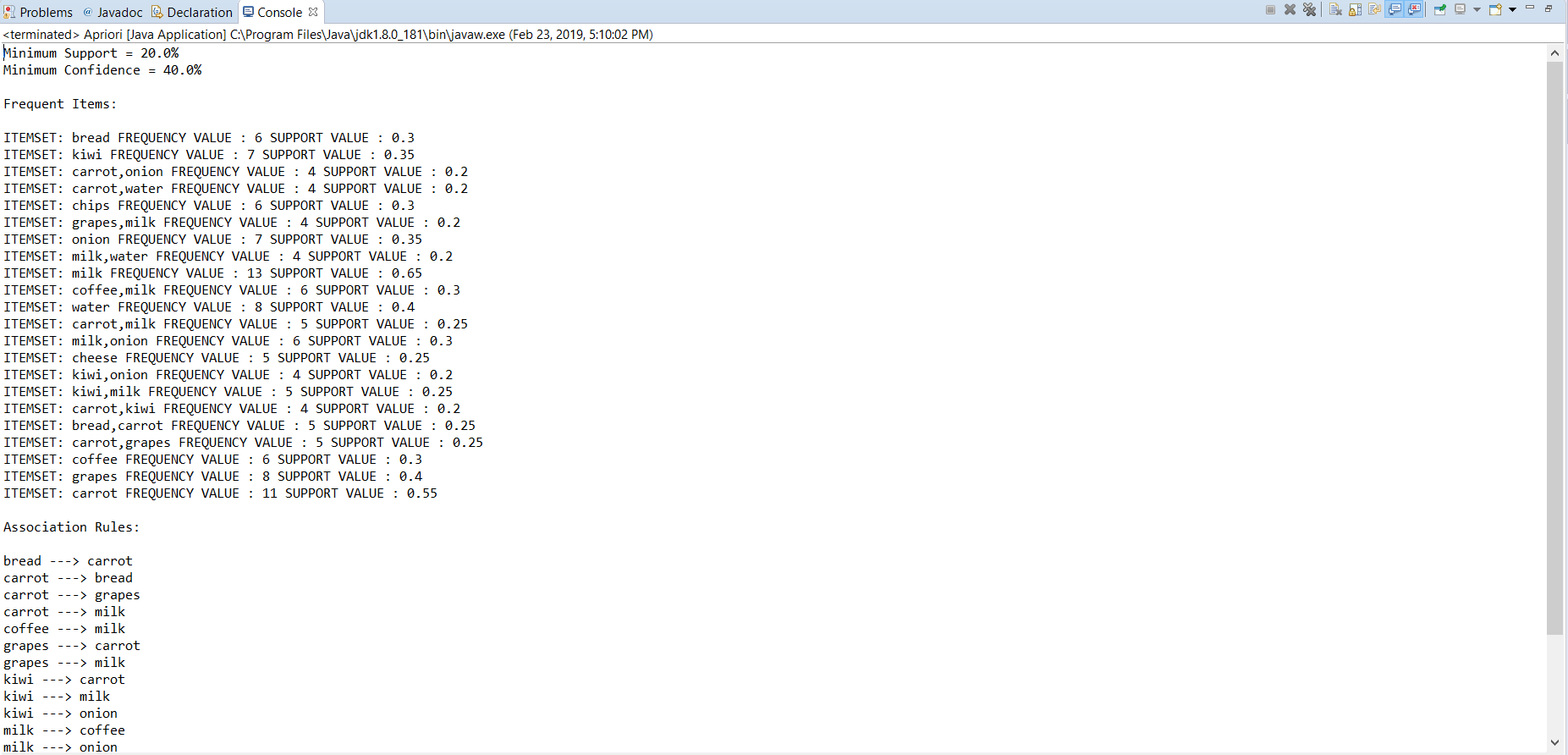


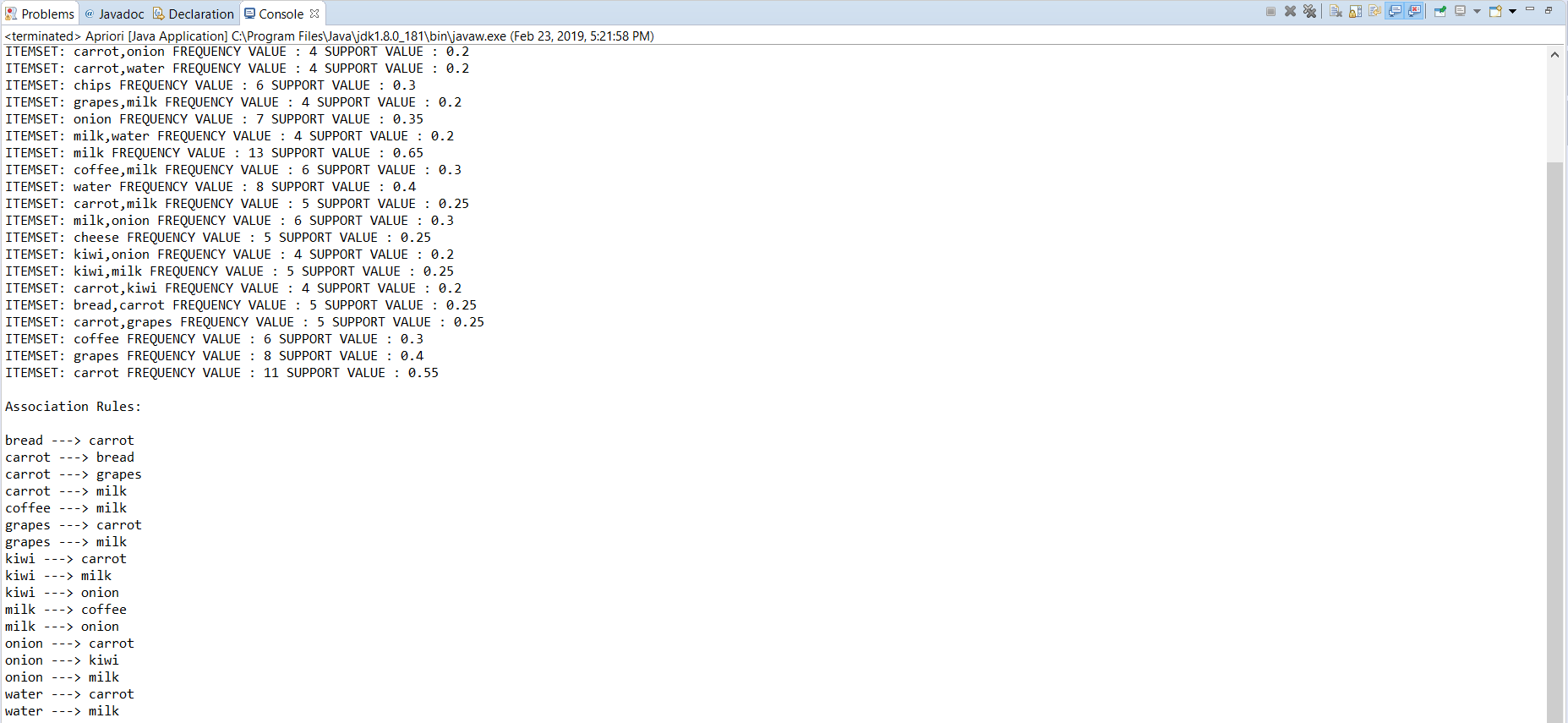
Grocerystore.txt:



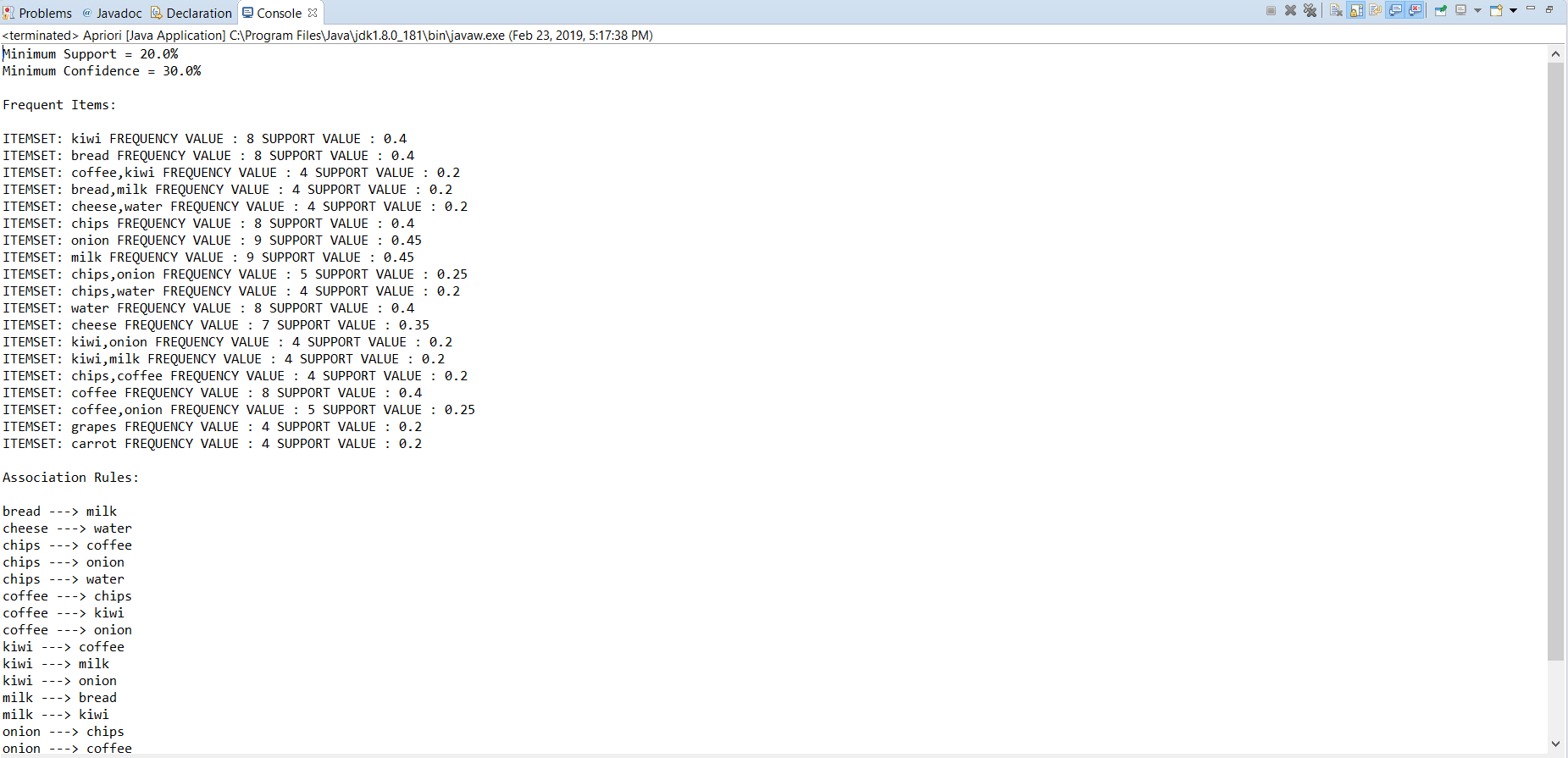
3. Output Screenshots:

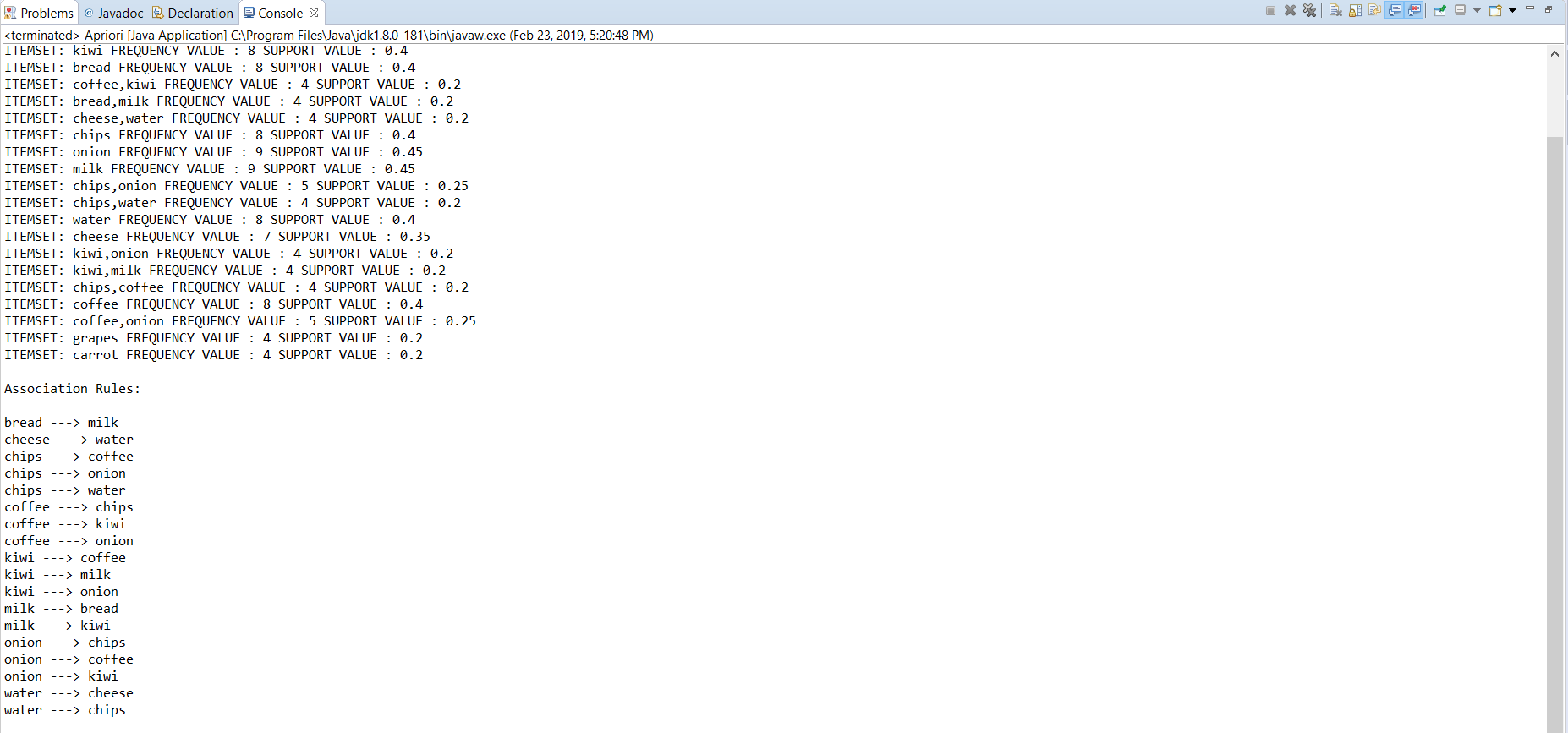
Amazon.txt:



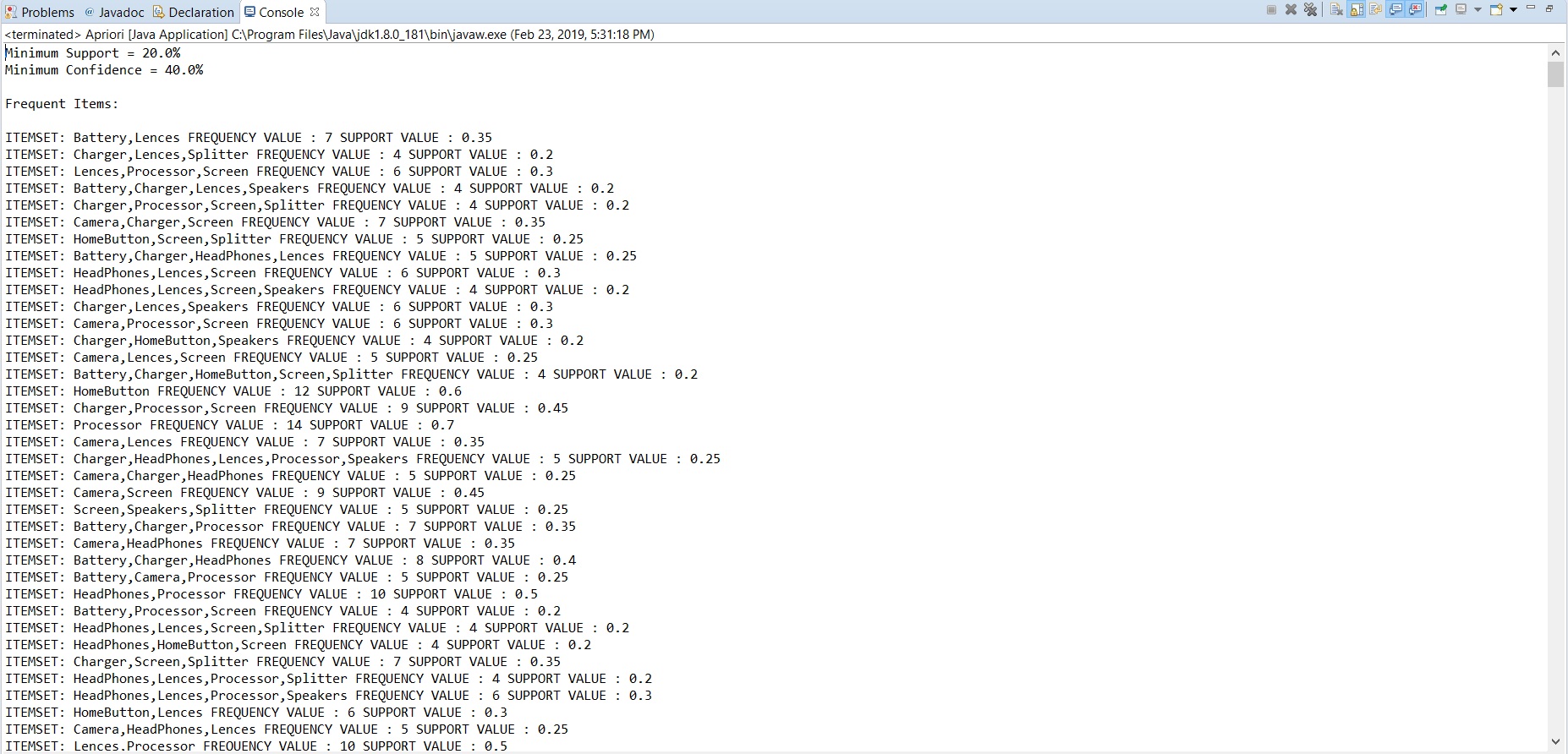


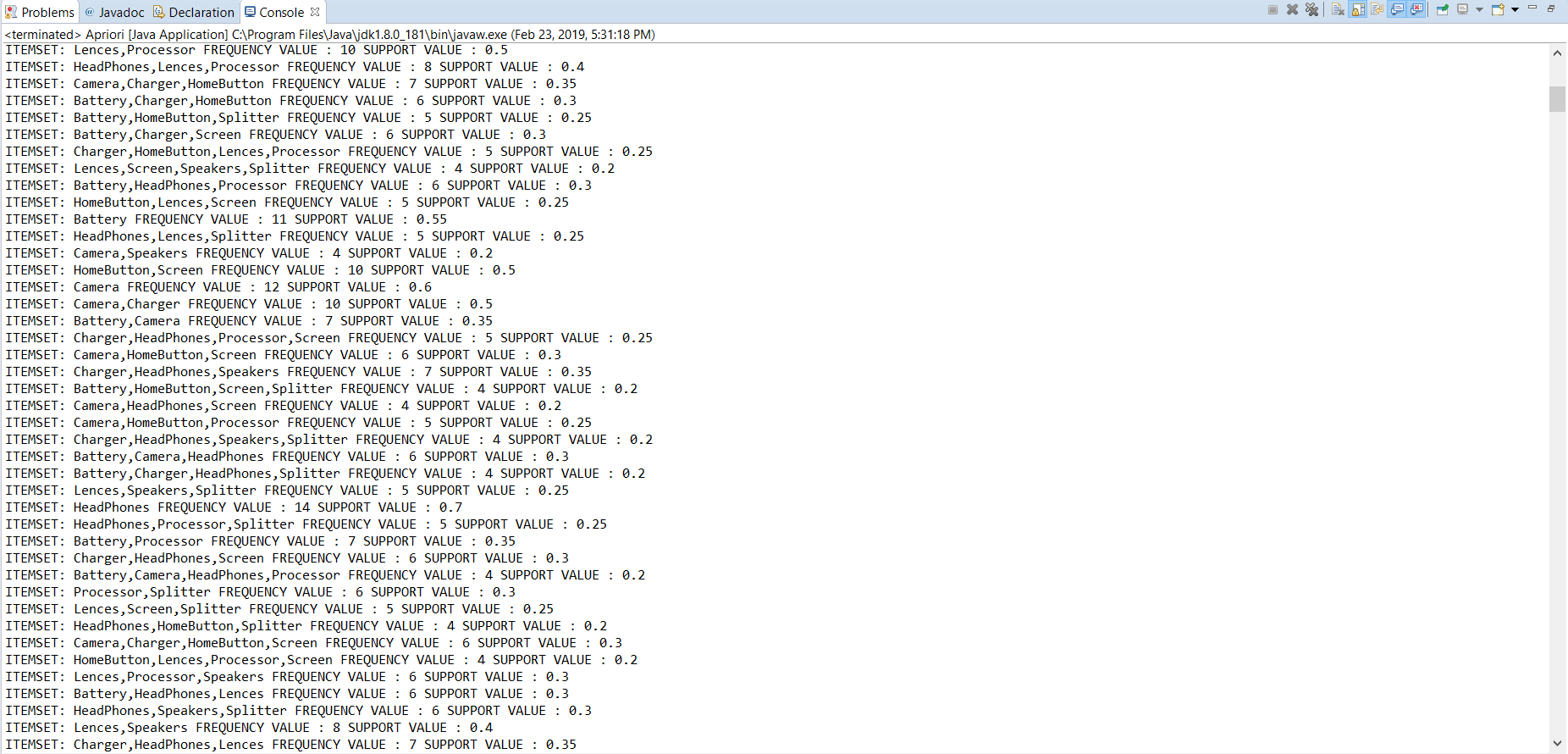
Walmart.txt:

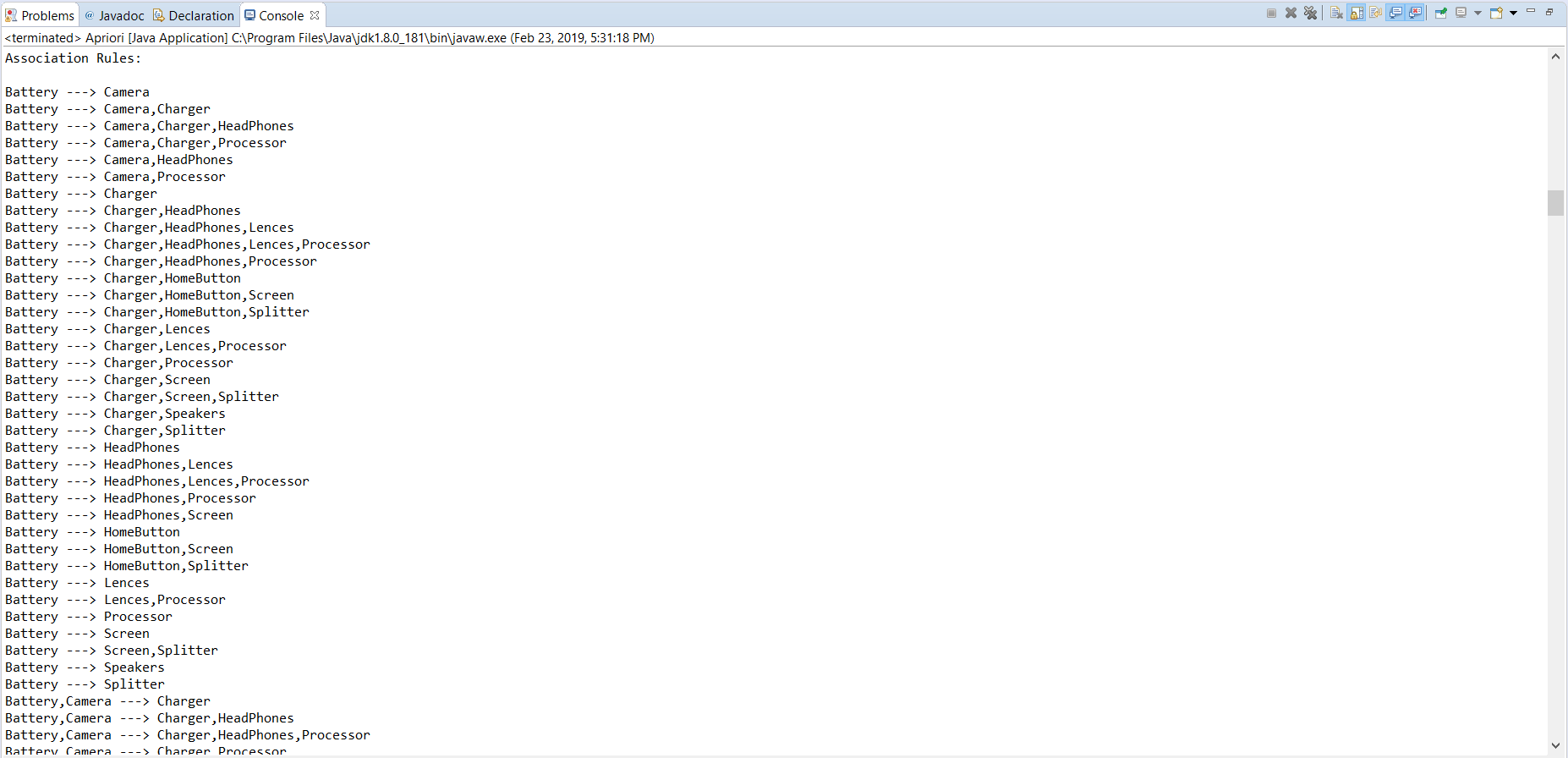




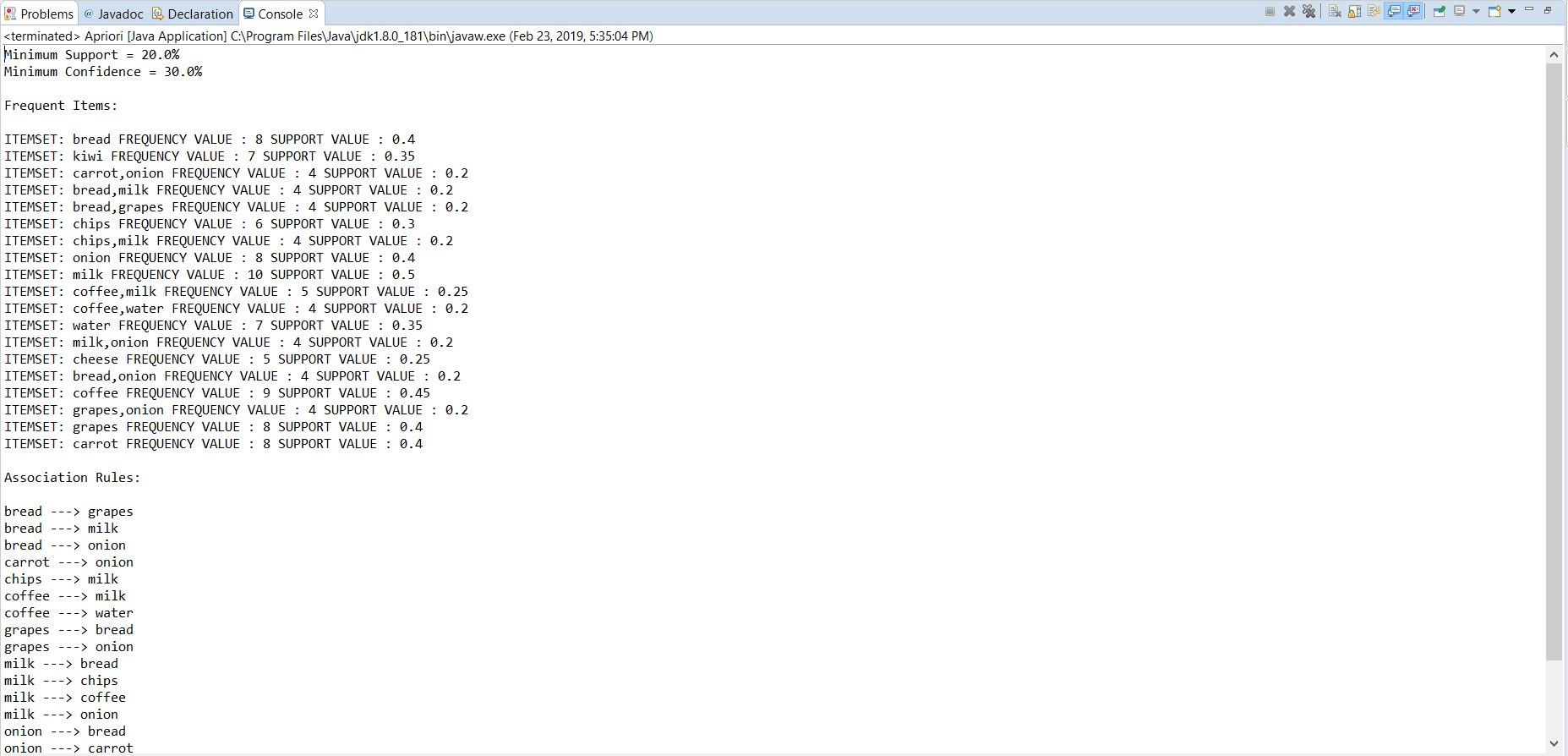
Computerstore.txt:

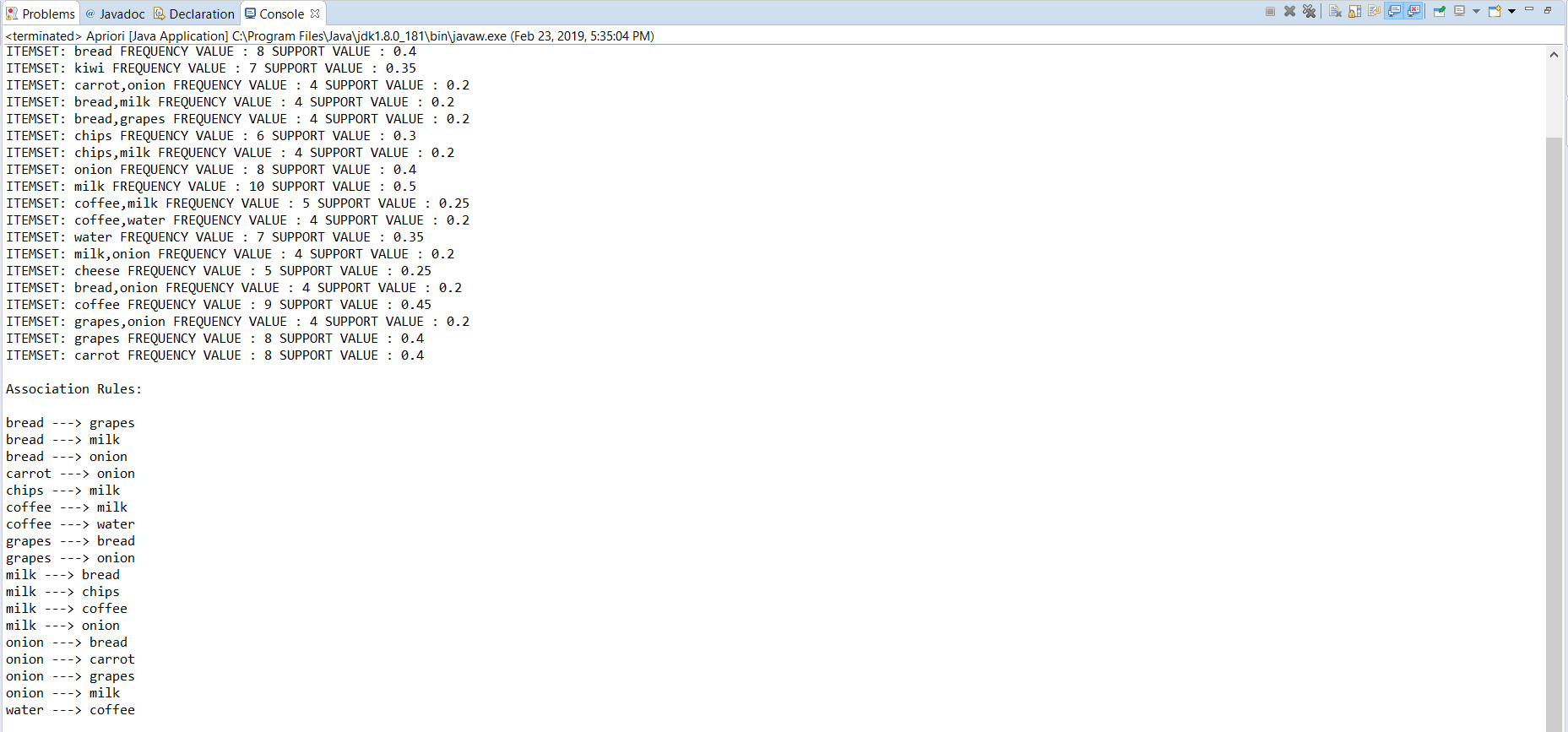






K-Mart.txt:





Grocerystore.txt:

