



Université Chouaïb Doukkali
Faculté des Sciences, Département d'informatique
EL JADIDA

Master BIBDA

Thème

Algorithme Apriori

RÉALISÉ PAR : DOUAIOUI Hamza

Année universitaire : 2020/2021

ALGORITHME A PRIORI

En informatique et data mining, l'**algorithme Apriori** est un classique algorithme de la recherche sur les associations. Il est utilisé pour la production de [itemset](#). Souvent, par approximations successives, depuis les jeux d'éléments avec un seul élément. En résumé, la base théorique sur lequel l'algorithme est basée sur l'idée que si un ensemble des éléments (itemsets) est fréquente, tous ses sous-ensembles sont fréquents, mais si un itemset est peu fréquente, alors même les jeux qui en contiennent sont fréquents (principe de l'anti-monotonie).

Un domaine dans lequel cet algorithme trouve une grande applicabilité est supermarché / problème de basket-ball. Pour tirer les associations est employé une approche bas vers le haut, où les sous-ensembles fréquents sont construits en ajoutant un élément à la fois (génération de candidats); les groupes de candidats ont ensuite été vérifiées sur les données et l'algorithme se termine quand il n'y a pas d'extensions plus possibles. Dans ce processus, le nombre d'itérations est, où il indique les nombres maximums d'un itemset fréquent.

Il existe d'autres algorithmes ayant des objectifs similaires (Winepi et Minepi), mais sont les plus répandus dans les zones où les données manquent horodatage (Par exemple, les séquences de ADN).

Apriori, même si historiquement significative, il souffre d'une certaine inefficacité. En particulier, la génération de candidats crée de nombreux sous-ensembles. Dans le processus, les sous-ensembles importants sont identifiés seulement après avoir trouvé tous sous-ensembles propres, où S est le sous-groupe particulier d'éléments (support) dans lequel un sous-ensemble particulier d'objets apparaît.[1]

Le projet est constitué de trois package [classes, interfaceGrafic, main]

Le package classes contient deux classes :

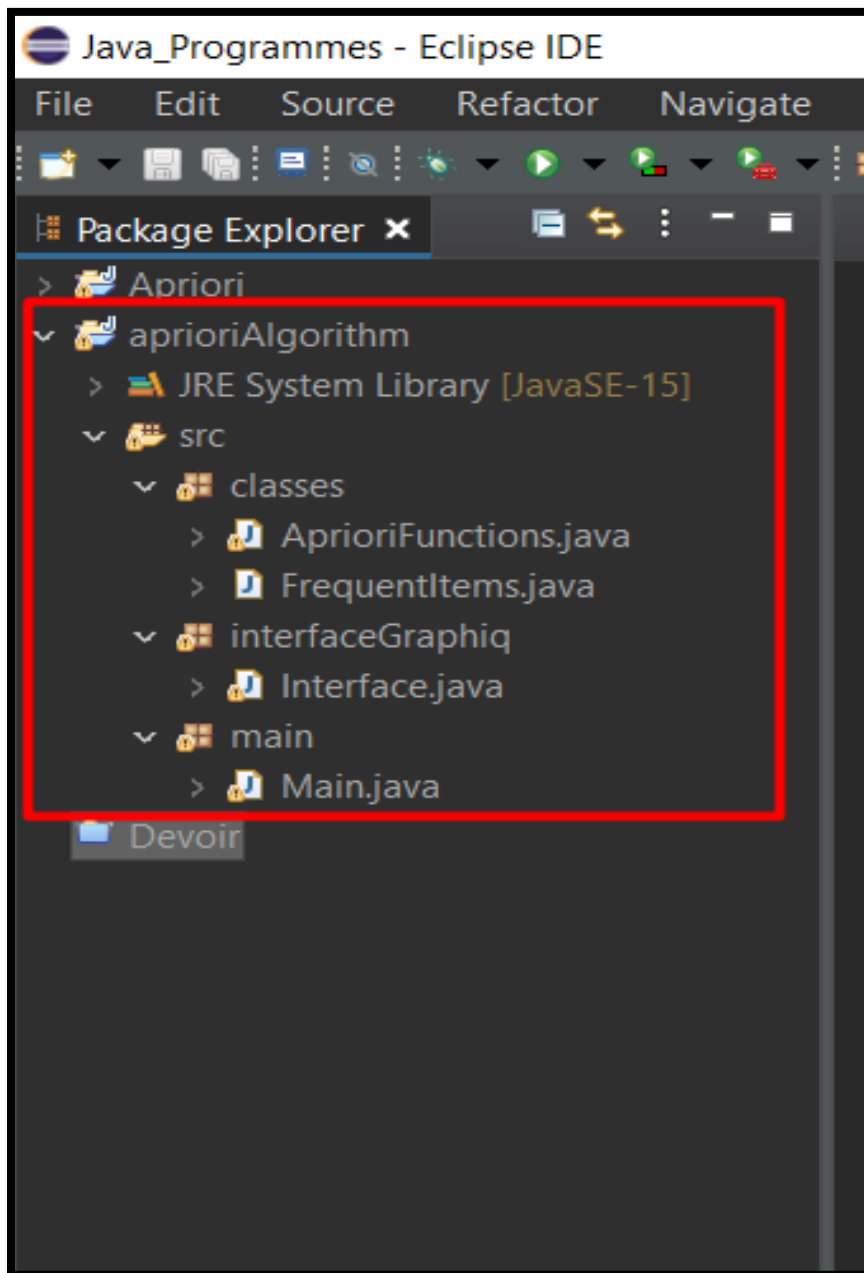
1. AprioriFunctions.java
2. FrequentItems.java

Le package interfaceGrafic contient une classe :

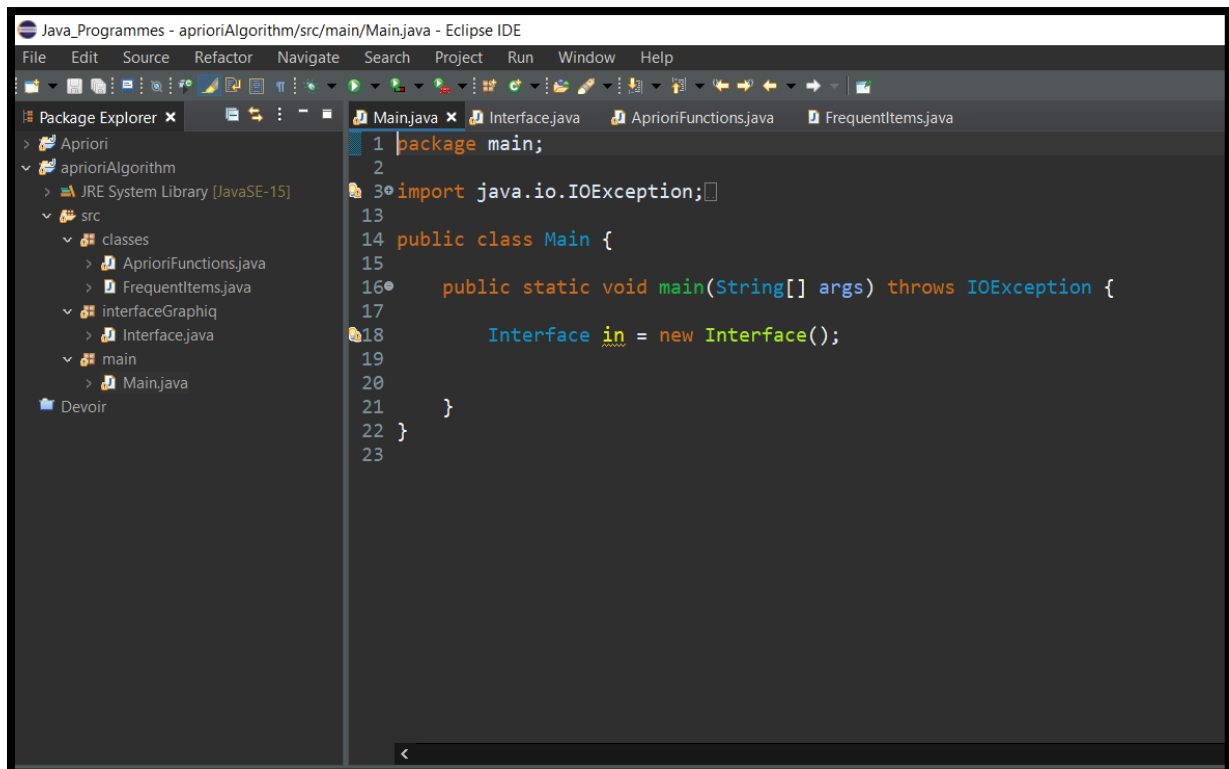
1. Interface.java

Le package main contient une classe :

1. Main.java



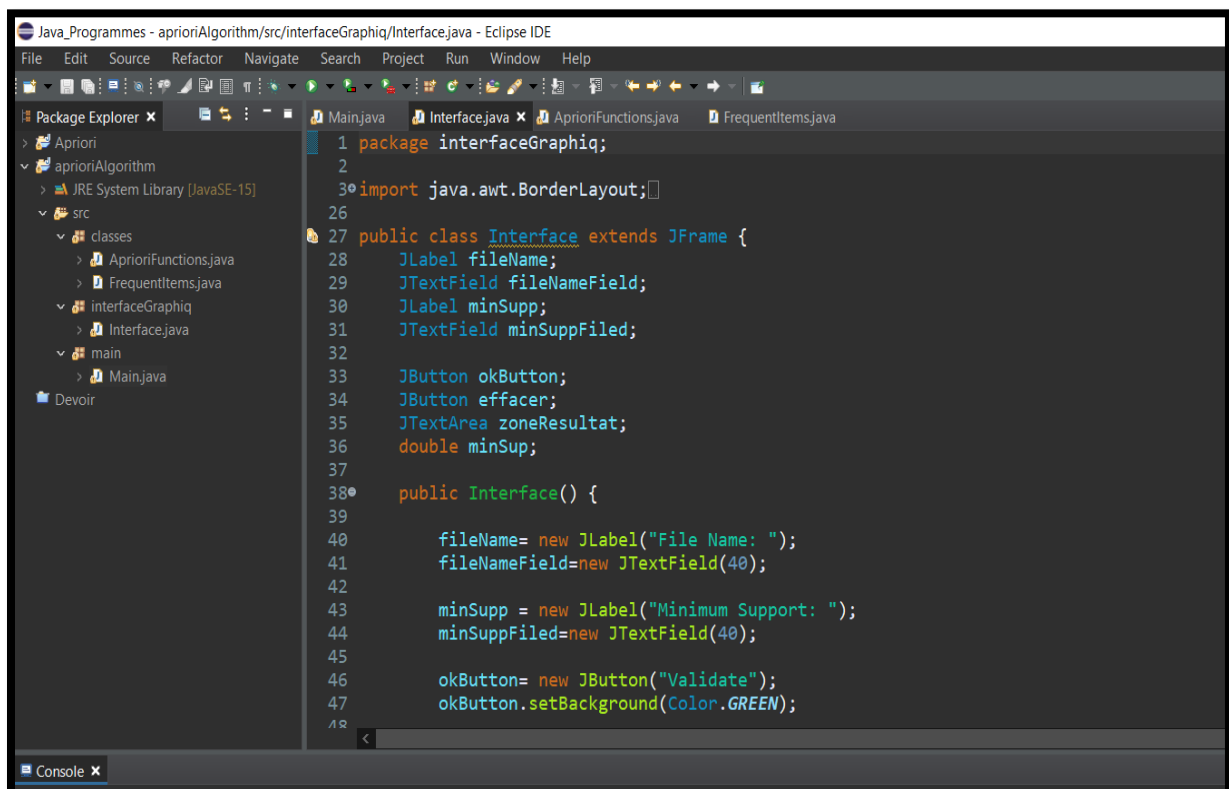
La classe Main.java est la classe principale



The screenshot shows the Eclipse IDE with the 'Main.java' file open. The Package Explorer on the left shows the project structure: 'Apriori' -> 'aprioriAlgorithm' -> 'src' -> 'classes' -> 'Main.java'. The main editor displays the following code:

```
1 package main;
2
3 import java.io.IOException;
4
5 public class Main {
6     public static void main(String[] args) throws IOException {
7         Interface in = new Interface();
8     }
9 }
```

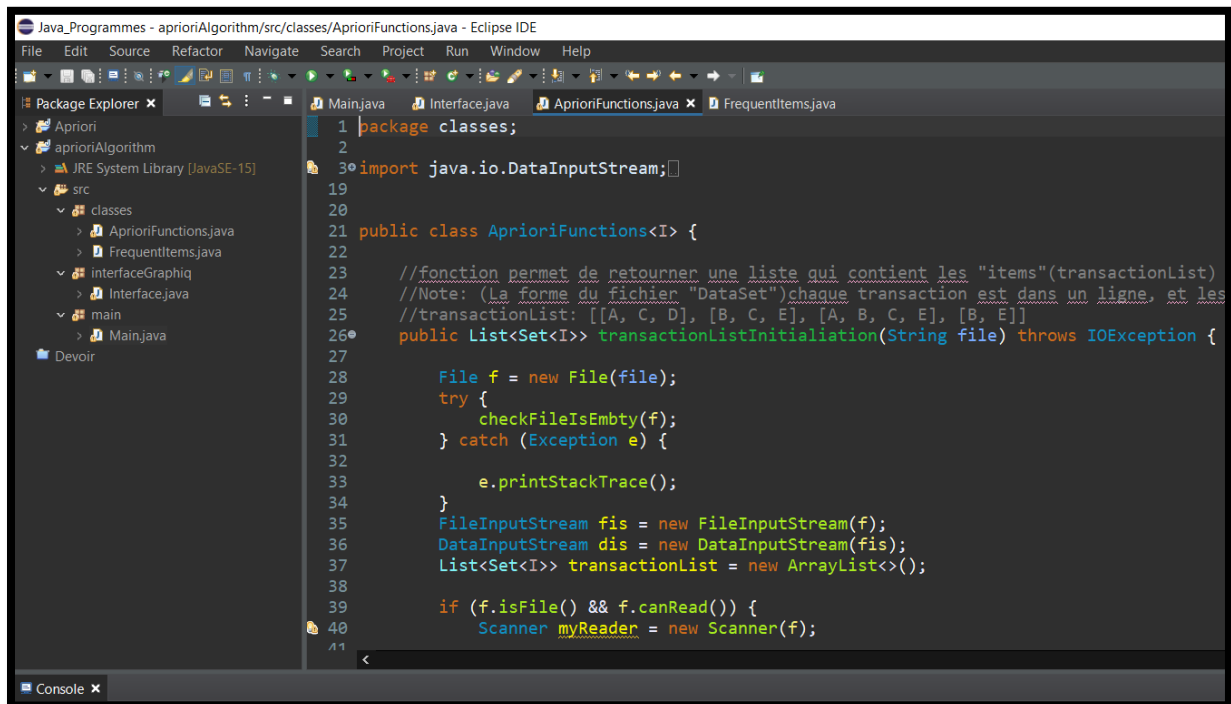
La classe Interface.java nous donne une interface graphique qui permet à l'utilisateur d'interagir avec le système.



The screenshot shows the Eclipse IDE with the 'Interface.java' file open. The Package Explorer on the left shows the project structure: 'Apriori' -> 'aprioriAlgorithm' -> 'src' -> 'interfaceGraphiq' -> 'Interface.java'. The main editor displays the following code:

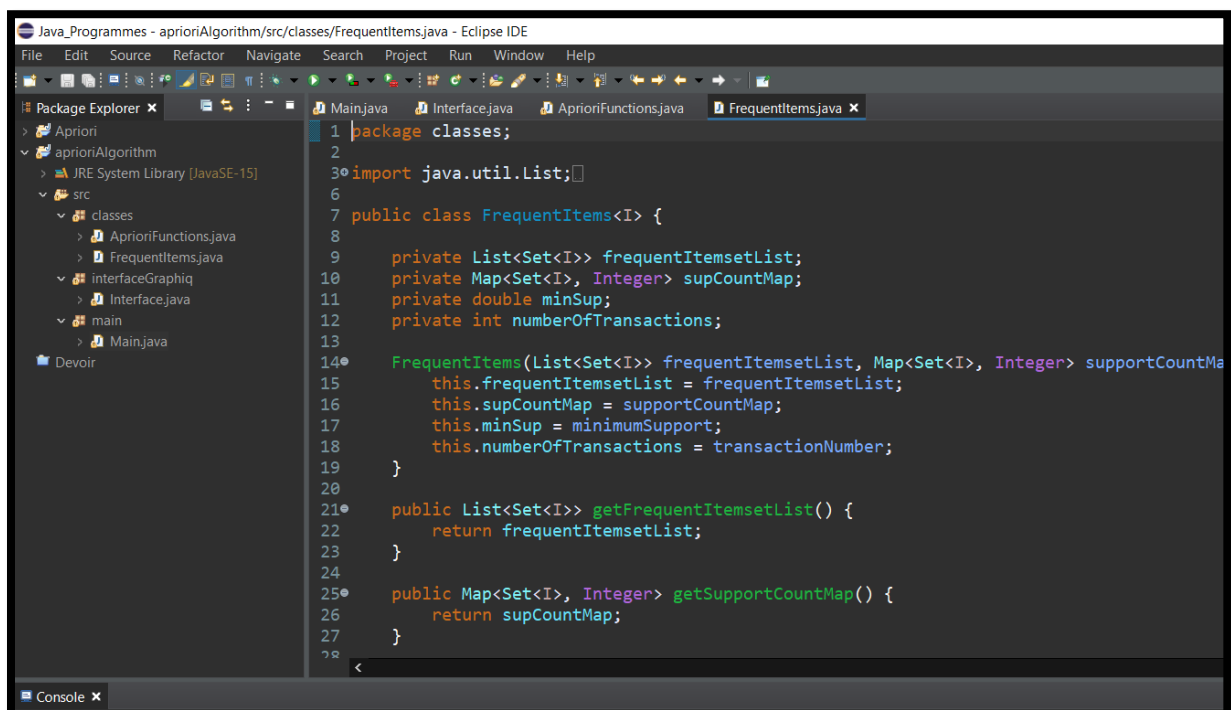
```
1 package interfaceGraphiq;
2
3 import java.awt.BorderLayout;
4
5 public class Interface extends JFrame {
6     JLabel fileName;
7     JTextField fileNameField;
8     JLabel minSupp;
9     JTextField minSuppFiled;
10
11     JButton okButton;
12     JButton effacer;
13     JTextArea zoneResultat;
14     double minSup;
15
16     public Interface() {
17         fileName= new JLabel("File Name: ");
18         fileNameField=new JTextField(40);
19
20         minSupp = new JLabel("Minimum Support: ");
21         minSuppFiled=new JTextField(40);
22
23         okButton= new JButton("Validate");
24         okButton.setBackground(Color.GREEN);
25     }
26 }
```

La classe `AprioriFunctions.java` contient les différentes méthodes de notre algorithme, à savoir la méthode qui permet de retourner une liste qui contient les "items" (transactionList) depuis un fichier (DATASET), la méthode qui permet de générer la liste des itemSet candidats suivants...



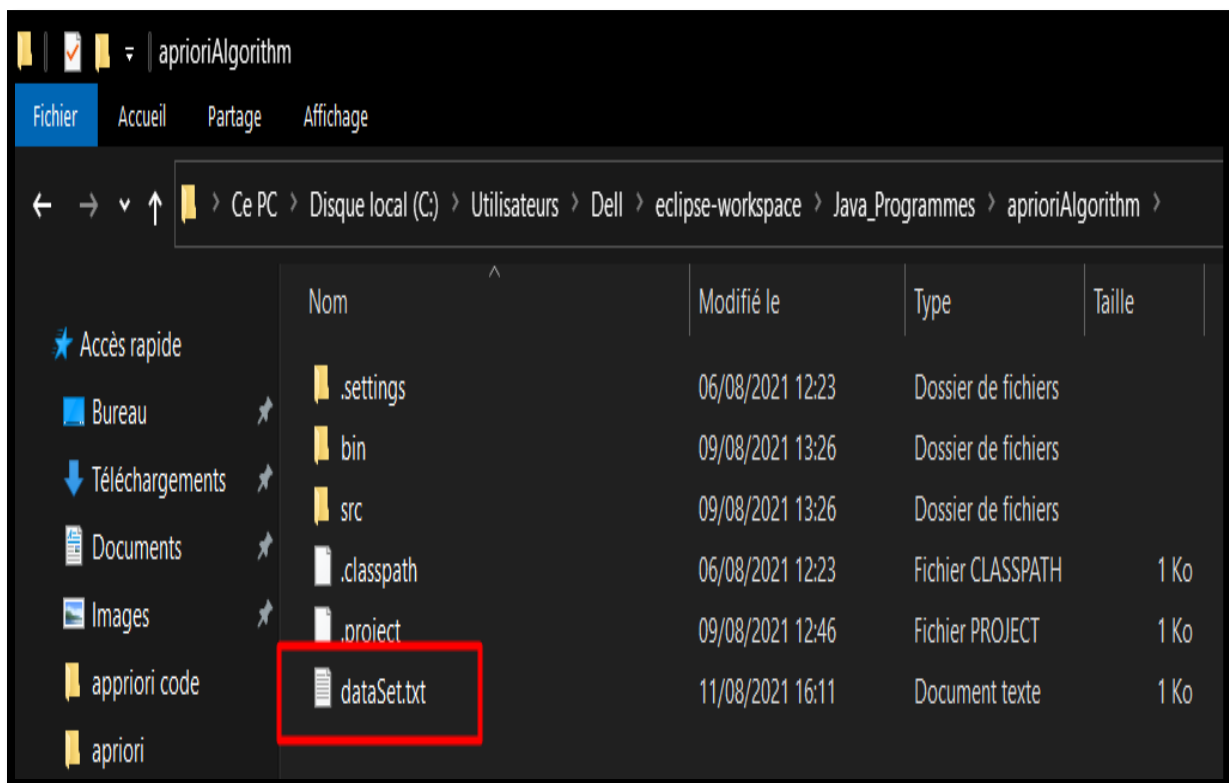
```
1 package classes;
2
3 import java.io.DataInputStream;
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21 public class AprioriFunctions<I> {
22
23     //fonction permet de retourner une liste qui contient les "items"(transactionList)
24     //Note: (La forme du fichier "DataSet")chaque transaction est dans une ligne, et les
25     //transactionList: [[A, C, D], [B, C, E], [A, B, C, E], [B, E]]
26     public List<Set<I>> transactionListInitiation(String file) throws IOException {
27
28         File f = new File(file);
29         try {
30             checkFileIsEmpty(f);
31         } catch (Exception e) {
32
33             e.printStackTrace();
34         }
35         FileInputStream fis = new FileInputStream(f);
36         DataInputStream dis = new DataInputStream(fis);
37         List<Set<I>> transactionList = new ArrayList<>();
38
39         if (f.isFile() && f.canRead()) {
40             Scanner myReader = new Scanner(f);
41
42         }
43     }
44 }
```

La classe `FrequentItems.java` représente la liste des items les plus fréquents, elle contient les différents attributs et les getters et setters.

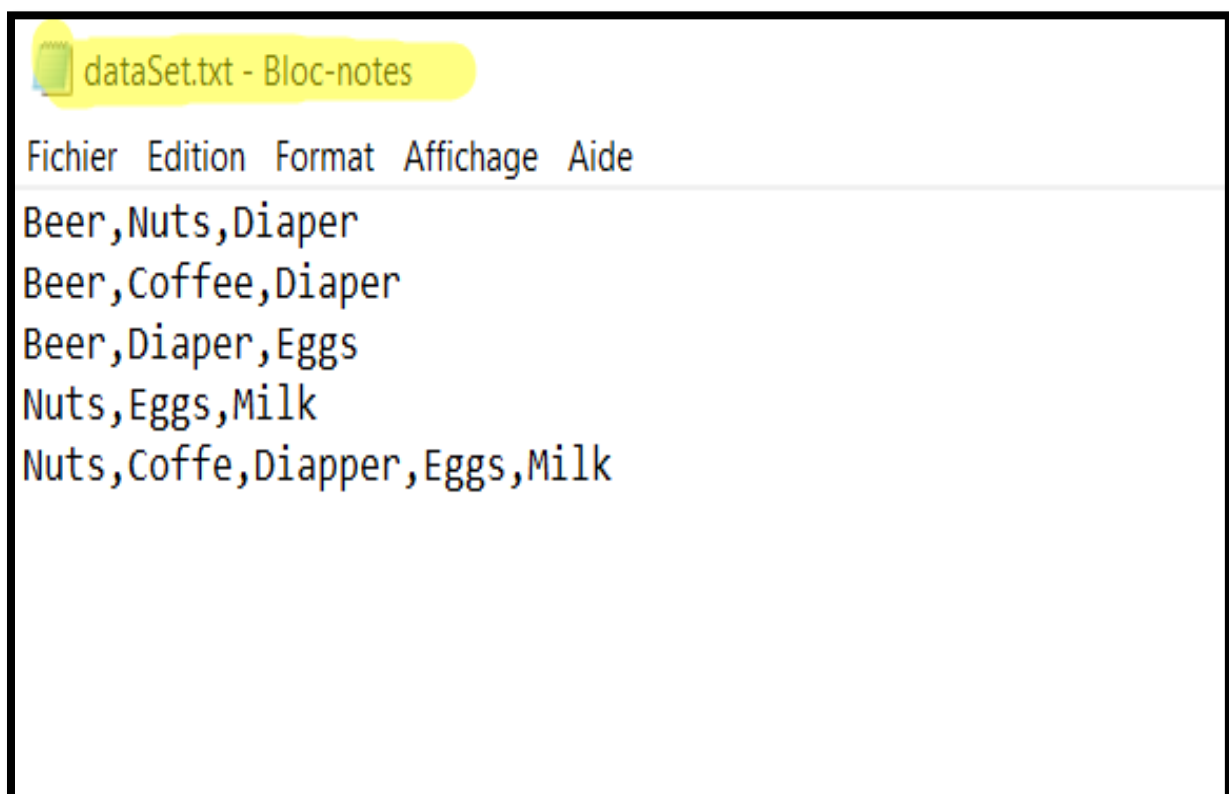


```
1 package classes;
2
3 import java.util.List;
4
5
6
7 public class FrequentItems<I> {
8
9     private List<Set<I>> frequentItemsetList;
10     private Map<Set<I>, Integer> supCountMap;
11     private double minSup;
12     private int numberOfTransactions;
13
14     FrequentItems(List<Set<I>> frequentItemsetList, Map<Set<I>, Integer> supportCountMap,
15                 double minSup, int numberOfTransactions) {
16         this.frequentItemsetList = frequentItemsetList;
17         this.supCountMap = supportCountMap;
18         this.minSup = minSup;
19         this.numberOfTransactions = numberOfTransactions;
20     }
21
22     public List<Set<I>> getFrequentItemsetList() {
23         return frequentItemsetList;
24     }
25
26     public Map<Set<I>, Integer> getSupportCountMap() {
27         return supCountMap;
28     }
29 }
```

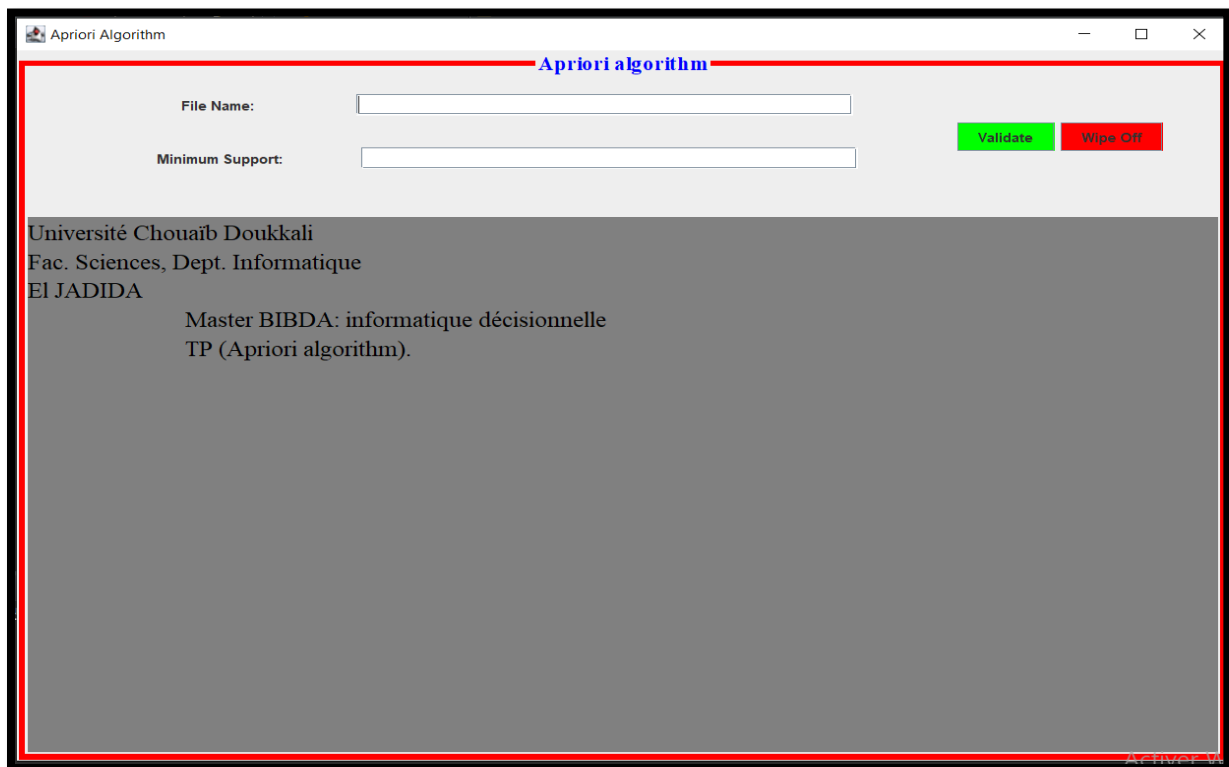
Nos données sont stockées dans un fichier texte par exemple(dataSet.txt)



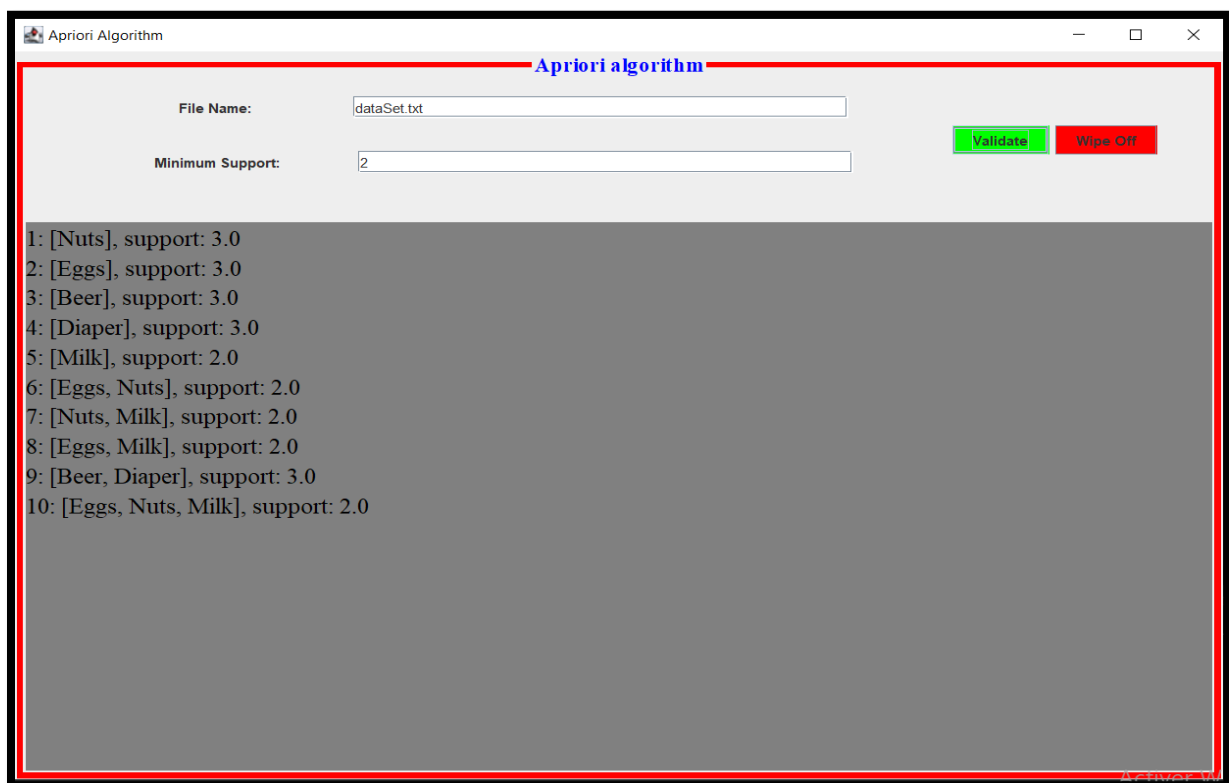
Voilà la forme des données dans notre fichier, on représente chaque transaction dans une ligne et on sépare les items par des virgules.



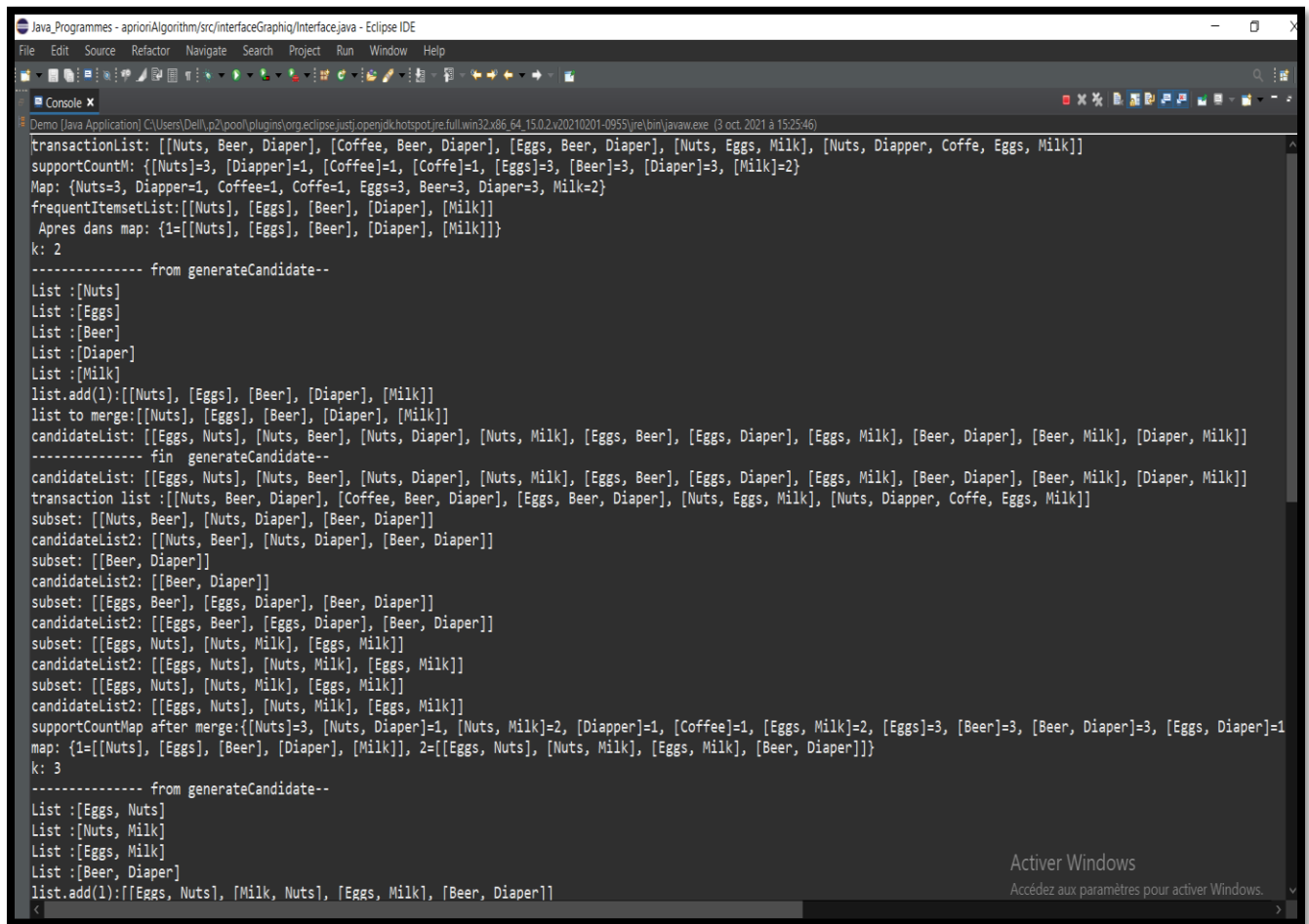
La figure suivante représente notre interface graphique, on donne le nom du notre fichier texte exemple (dataSet.txt), et le support minimum par exemple 2, après on clique sur le bouton ‘Valider’ pour avoir les résultats, le bouton ‘Wipe Off’ permet d’effacer tout.



Exemple



La figure suivante représente notre console dans lequel on affiche les différentes étapes de l'exécution de notre algorithme.



```
Java Programmes - aprioriAlgorithm/src/interfaceGraphiq/Interface.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help

Console
Demo [Java Application] C:\Users\Delil\p2\pool\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_15.0.2.v20210201-0955\jre\bin\javaw.exe (3 oct. 2021 à 15:25:46)

transactionList: [[Nuts, Beer, Diaper], [Coffee, Beer, Diaper], [Eggs, Beer, Diaper], [Nuts, Eggs, Milk], [Nuts, Diaper, Coffe, Eggs, Milk]]
supportCountM: {[Nuts]=3, [Diapper]=1, [Coffee]=1, [Coffe]=1, [Eggs]=3, [Beer]=3, [Diaper]=3, [Milk]=2}
Map: {Nuts=3, Diapper=1, Coffee=1, Coffe=1, Eggs=3, Beer=3, Diapers=3, Milk=2}
frequentItemsetList: [[Nuts], [Eggs], [Beer], [Diaper], [Milk]]
Après dans map: {1=[[Nuts], [Eggs], [Beer], [Diaper], [Milk]]}
k: 2
----- from generateCandidate--
List :[Nuts]
List :[Eggs]
List :[Beer]
List :[Diaper]
List :[Milk]
list.add(1):[[Nuts], [Eggs], [Beer], [Diaper], [Milk]]
list to merge:[[Nuts], [Eggs], [Beer], [Diaper], [Milk]]
candidateList: [[Eggs, Nuts], [Nuts, Beer], [Nuts, Diaper], [Nuts, Milk], [Eggs, Beer], [Eggs, Diaper], [Eggs, Milk], [Beer, Diaper], [Beer, Milk], [Diaper, Milk]]
----- fin generateCandidate--
candidateList: [[Eggs, Nuts], [Nuts, Beer], [Nuts, Diaper], [Nuts, Milk], [Eggs, Beer], [Eggs, Diaper], [Eggs, Milk], [Beer, Diaper], [Beer, Milk], [Diaper, Milk]]
transaction list :[[Nuts, Beer, Diaper], [Coffee, Beer, Diaper], [Eggs, Beer, Diaper], [Nuts, Eggs, Milk], [Nuts, Diapper, Coffe, Eggs, Milk]]
subset: [[Nuts, Beer], [Nuts, Diaper], [Beer, Diaper]]
candidateList2: [[Nuts, Beer], [Nuts, Diaper], [Beer, Diaper]]
subset: [[Beer, Diaper]]
candidateList2: [[Beer, Diaper]]
subset: [[Eggs, Beer], [Eggs, Diaper], [Beer, Diaper]]
candidateList2: [[Eggs, Beer], [Eggs, Diaper], [Beer, Diaper]]
subset: [[Eggs, Nuts], [Nuts, Milk], [Eggs, Milk]]
candidateList2: [[Eggs, Nuts], [Nuts, Milk], [Eggs, Milk]]
subset: [[Eggs, Nuts], [Nuts, Milk], [Eggs, Milk]]
candidateList2: [[Eggs, Nuts], [Nuts, Milk], [Eggs, Milk]]
supportCountMap after merge:{[Nuts]=3, [Nuts, Diaper]=1, [Nuts, Milk]=2, [Diapper]=1, [Coffee]=1, [Eggs, Milk]=2, [Eggs]=3, [Beer]=3, [Beer, Diaper]=3, [Eggs, Diaper]=1}
map: {1=[[Nuts], [Eggs], [Beer], [Diaper], [Milk]], 2=[[Eggs, Nuts], [Nuts, Milk], [Eggs, Milk], [Beer, Diaper]]}
k: 3
----- from generateCandidate--
List :[Eggs, Nuts]
List :[Nuts, Milk]
List :[Eggs, Milk]
List :[Beer, Diaper]
list.add(1):[[Eggs, Nuts], [Milk, Nuts], [Eggs, Milk], [Beer, Diaper]]
```

Références :

[1] : <https://boowiki.info/art/algorithmes/algorithmme-a-priori.html>