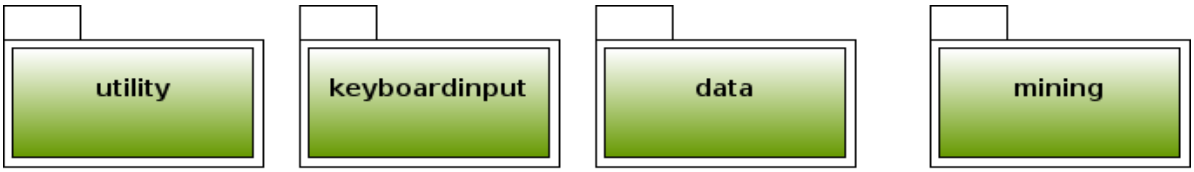
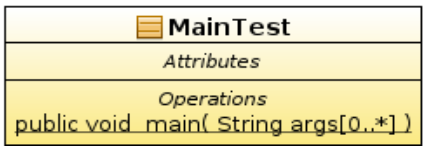
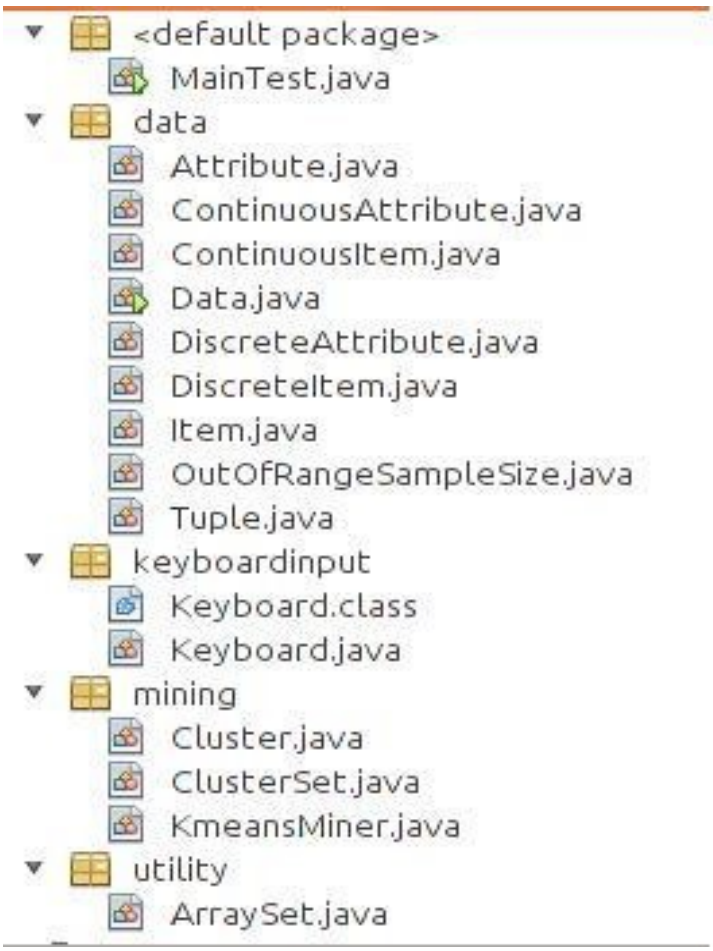
Exercise 3 - Package, Keyboard Input and Exceptions.





Define the utility, keyboardinput,data and mining packages by moving the classes to the package they belong to according to the diagram below. If necessary, also modify the visibility qualifiers ***appropriately.***



■ *Add the Keyboard class (provided by the teacher) that collects class methods for capturing keyboard input*

■ *Modify MainTest so as to establish an interaction with the user to acquire from the keyboard the integer number k of clusters to be discovered and give the user the possibility to decide to repeat the execution of the k-means even with different k-values.*

■ *Modify the Data class by adding the method:*

private int countDistinctTuples()

Input:

Output: Number of distinct transactions stored in the data matrix

Behaviour: Count the number of distinct transactions stored in date (make use of the boolean method compare(int,int))

Example:

a,b,a,c

a,b,d

a,b,a,c

a,b,d,c

There are three separate transactions (highlighted in yellow)

■ *Modify the Data class by adding the given member: private int distinctTuples;*

■ *Modify the implementation of the constructor constructor of the classData by adding the initialisation:*

```
public Data(){
...
distinctTuples=countDistinctTuples();
```

}

■ *Define an OutOfRangeSampleSize class to model a **controlled exception** to be considered if the number k of clusters entered from the keyboard is greater than the number of centroids generated by the set of transactions.*

In such a case, the exception object must be created and raised in the implementation of the sampling(...) method.

```
public int[] sampling(int k) throws OutOfRangeSample
Size// to be modified
```

Behaviour: If k<=0 or k>distinctTuples then an instance object of OutRangeSampleSize is created and ejected. Otherwise we proceed with previously defined implementations

The exception object must be propagated to the main (following the call stack). The handler of this exception must be appropriately defined in the main of the MainTest class.

Examples of output:

```
run:
0:sunny,hot,high,weak,no
1:sunny,hot,high,strong,no
2:overcast,hot,high,weak,yes
3:rain,mild,high,weak,yes
4:rain,cool,normal,weak,yes
5:rain,cool,normal,strong,no
6:overcast,cool,normal,strong,yes
7:sunny,mild,high,weak,no
8:sunny,cool,normal,weak,yes
9:rain,mild,normal,weak,yes
```

10:sunny,mild,normal,strong,yes

11:overcast,mild,high,strong,yes

12:overcast,hot,normal,weak,yes

13:rain,mild,high,strong,no

Enter k:3

Iteration number:3 0:Centroid=(overcast

hot high weak yes) Examples:

[overcast hot high weak yes] dist=0.0

[overcast hot normal weak yes] dist=1.0

AvgDistance=0.5

1:Centroid=(rain cool normal weak yes)

Examples:

[rain mild high weak yes] dist=2.0

[rain cool normal weak yes] dist=0.0

[rain cool normal strong no] dist=2.0

[overcast cool normal strong yes] dist=2.0

[sunny cool normal weak yes] dist=1.0

[rain mild normal weak yes] dist=1.0

AvgDistance=1.33333333333333

2:Centroid=(sunny mild high strong no)

Examples:

[sunny hot high weak no] dist=2.0

[sunny hot high strong no] dist=1.0

[sunny mild high weak no] dist=1.0

[sunny mild normal strong yes] dist=2.0

[overcast mild high strong yes] dist=2.0

[rain mild high strong no] dist=1.0

AvgDistance=1.5

Do you want to repeat the execution?(y/n)