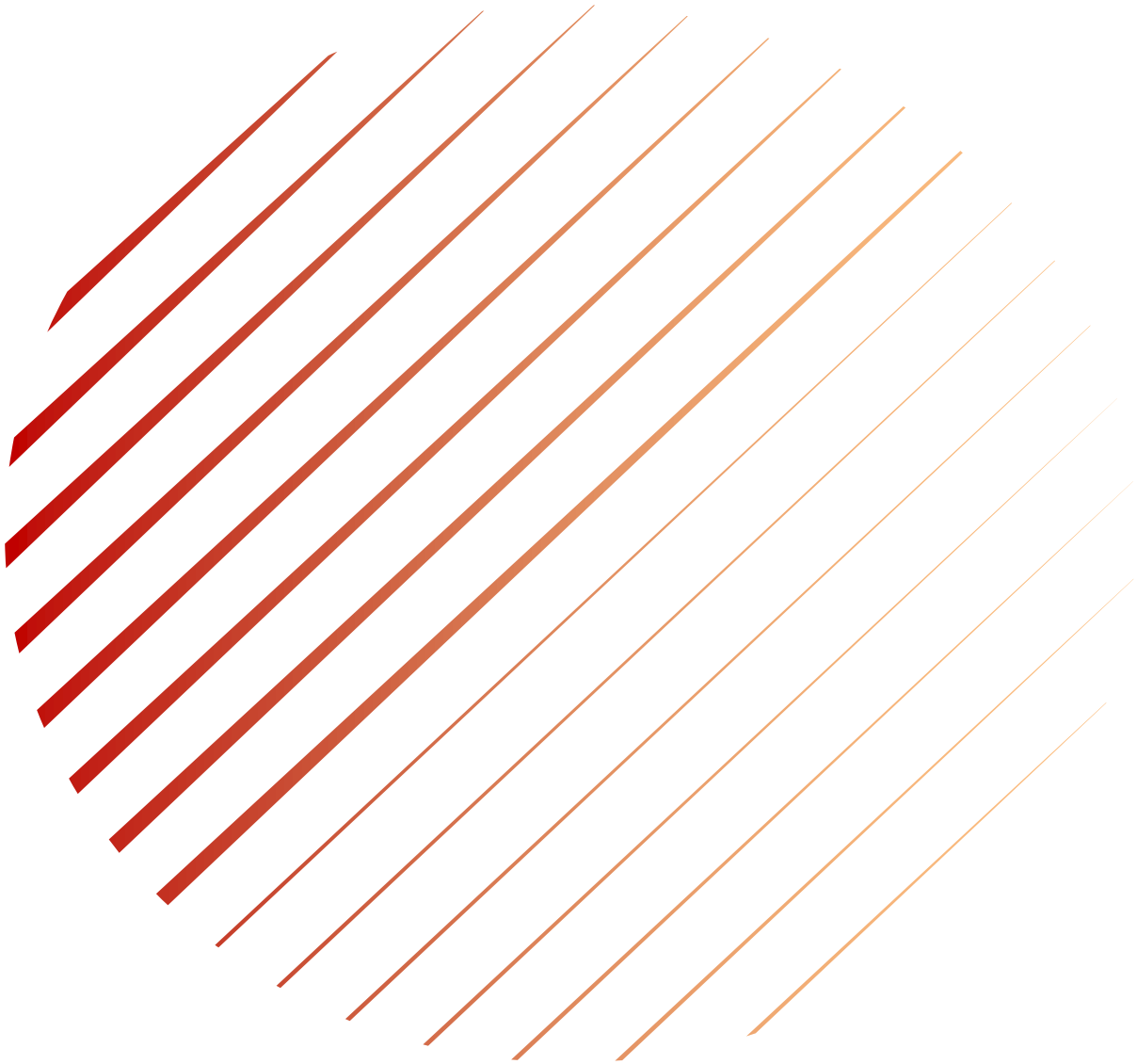
K-Means clustering

Caso di studio di Metodi Avanzati di Programmazione
AA 2022-2023



Realizzato Da

Simone De Girolamo 717450
s.degirolamo1@studenti.uniba.it

Ivan Digioia 716685
i.digioia3@studenti.uniba.it

SOMMARIO

1. INTRODUZIONE

1.1 L'algoritmo k-Means

L'algoritmo del k-Means clustering è un algoritmo di creazione di clustering utilizzato per suddividere i dati in gruppi omogenei basati sulla loro somiglianza. La prima idea del k-Means risale al 1956, l'anno successivo nel 57' il suo l'algoritmo standard verrà proposto da Stuart Lloyd, nel 65' Edward Forgy pubblicò lo stesso metodo e l'algoritmo fino al 67' prenderà il nome di Lloyd-Forgy anno in cui prenderà il nome di k-Means.
Esso è progettato per operare su dati omogenei. Ogni transazione è vista come un insieme di elementi. Dato un numero di cluster K, l'algoritmo divide l'insieme totale delle transazioni in K gruppi (Cluster) generati in base alla distanza tra ogni elemento, e ne calcola un Centroide, ovvero un punto medio tra tutte le transazioni di quel cluster.

K-Means utilizza un approccio "dal basso verso l'alto", in cui i sottoinsiemi vengono creati casualmente dai dati. Ad ogni iterazione assegna ciascuna osservazione al centroide più vicino e ricalcola i centroidi. L'algoritmo termina quando raggiunge il numero massimo di iterazioni.

K-Means utilizza una struttura di partizionamento per suddividere una grande mole di dati in cluster gestibili. Genera K celle, dove ogni cella rappresenta un cluster, e ogni transazione viene assegnata al cluster più vicino. Quindi procede a minimizzare la varianza all'interno di ogni cluster calcolando la distanza tra le singole transazioni ed il centroide. Infine, ripete questo processo spostando gli elementi in base a quanto sono distanti dal prototipo del centroide calcolato.

1.2 Limiti

L'algoritmo K-Means soffre di una serie di inefficienze. È necessario scegliere manualmente il numero di cluster, e questo può influenzare significativamente i risultati. Infatti, un numero di cluster troppo piccolo rispetto al numero di transazioni potrebbe rendere i cluster stessi troppo grandi ed includere dati diversi, il che porta ad una sovrapposizione tra cluster. Viceversa, un grande numero di cluster porta gli stessi ad essere troppo specifici e potrebbero includere all'interno rumore o variazioni casuali nei dati

L'algoritmo inizia selezionando casualmente i centroidi a partire dai dati. Se i centroidi sono scelti in maniera subottimale, l'algoritmo potrebbe convergere verso un ottimo locale invece di un ottimo globale. Di conseguenza, è necessario eseguire l'algoritmo più volte con diverse inizializzazioni, così da poter selezionare il risultato migliore

2. INTRODUZIONE AL PROGETTO

2.1 Descrizione del progetto

Il software realizzato utilizza l'algoritmo k-Means, descritto nella sezione precedente, esso elabora dati da una tabella presente in un database di tipo MySQL.
Il progetto, risultato di esercitazioni, consiste in un'applicazione di tipo Client/Server.

Il server si occupa di ricevere le richieste di un client, il quale può effettuare le seguenti operazioni:
-Generare un numero di cluster partendo dai dati del database e li memorizza in un file .ser
-Caricare da un file .ser i cluster memorizzati

In entrambi i casi, il client dovrà specificare nei criteri di ricerca:
-Il nome della tabella da cui estrarre i dati dal database
-Il numero di cluster

Il server mostra inoltre informazioni sul client connesso:
le operazioni da esso richieste e il loro esito.

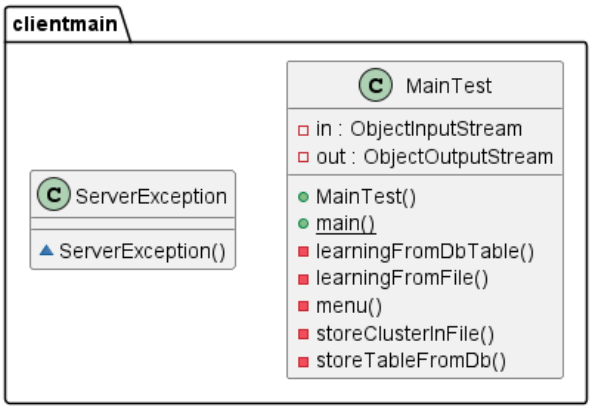
Nella sezione 3 sono riportati anche i diagrammi UML per il client e per il server. Inoltre, nella cartella "Javadoc" è stata allegata la Javadoc creata direttamente dall'IDE di sviluppo (IntelliJ). Nella sezione 5 del documento sono riportati esempi di esecuzione.

3. DIAGRAMMI UML

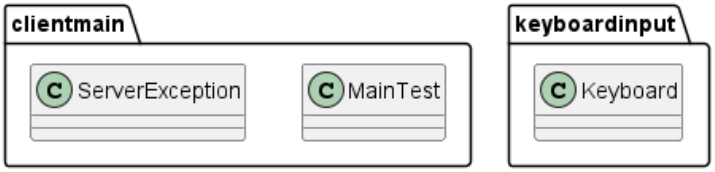
Segue la realizzazione dei diagrammi per la versione Base del MeansServer e KMeansClient

3.1 Client UML

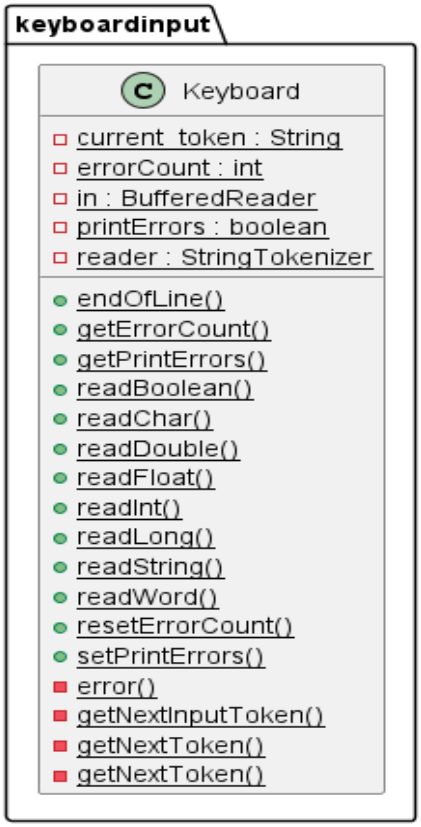
CLIENTMAIN's Class Diagram

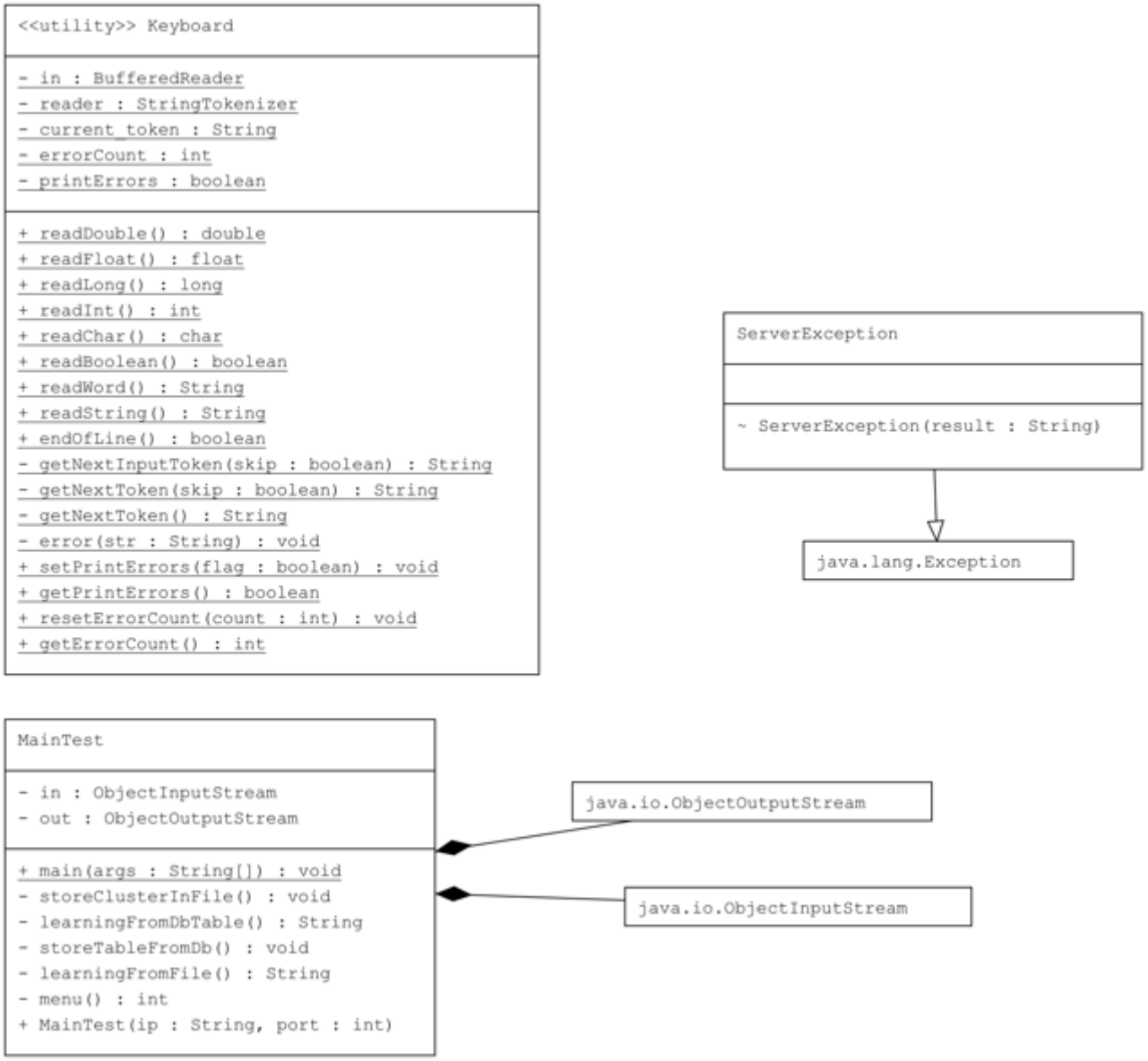


KMEANSCLIENT's Class Diagram

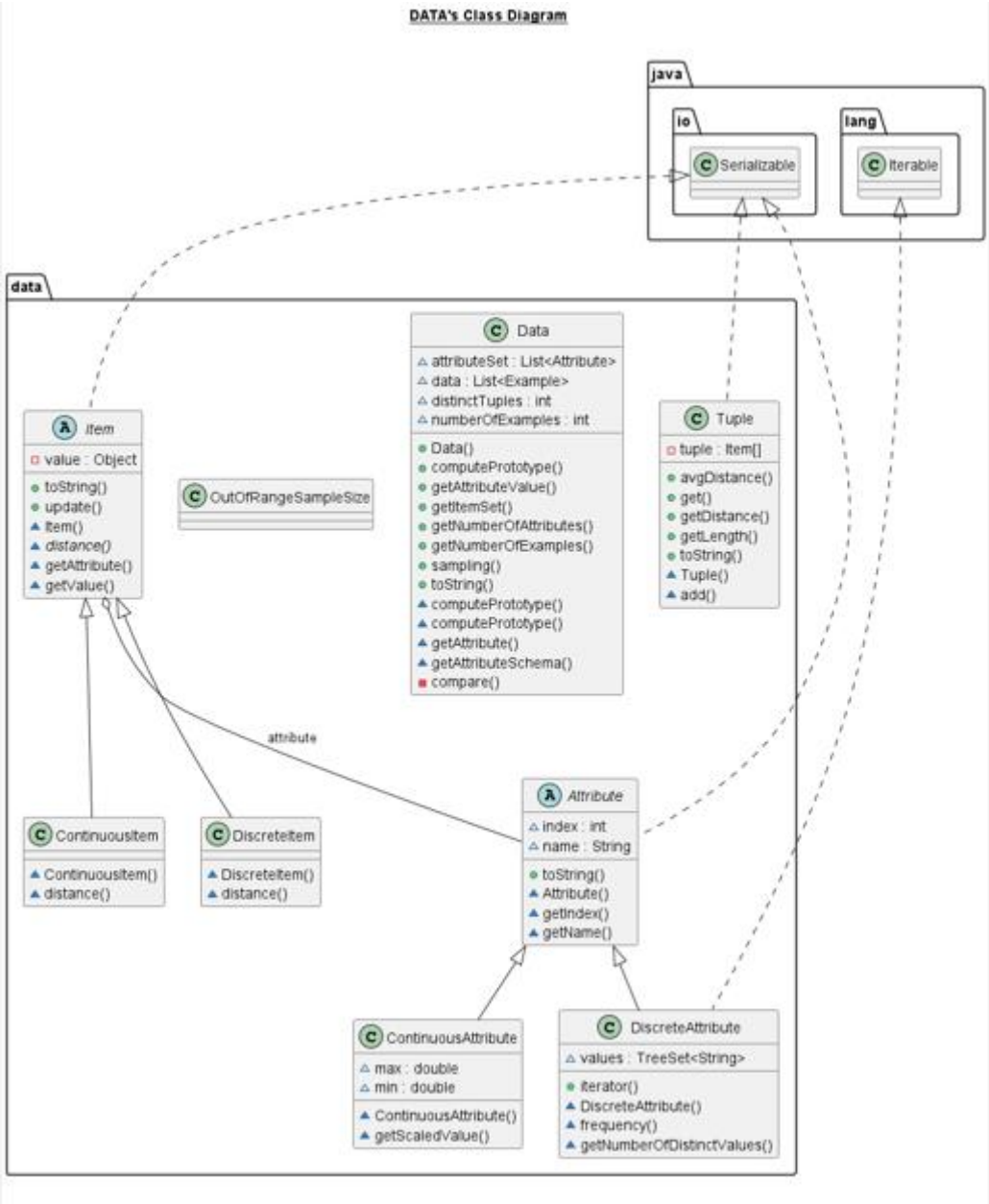


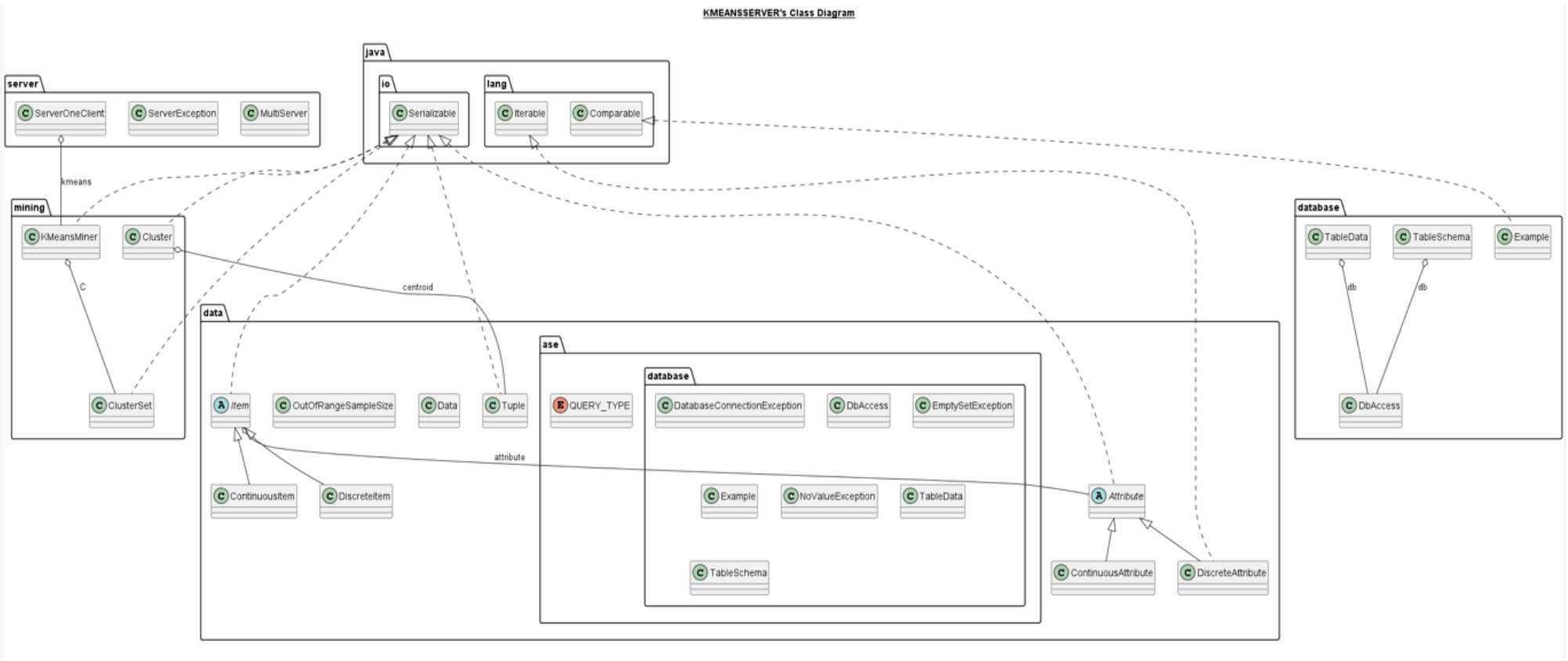
KEYBOARDINPUT's Class Diagram



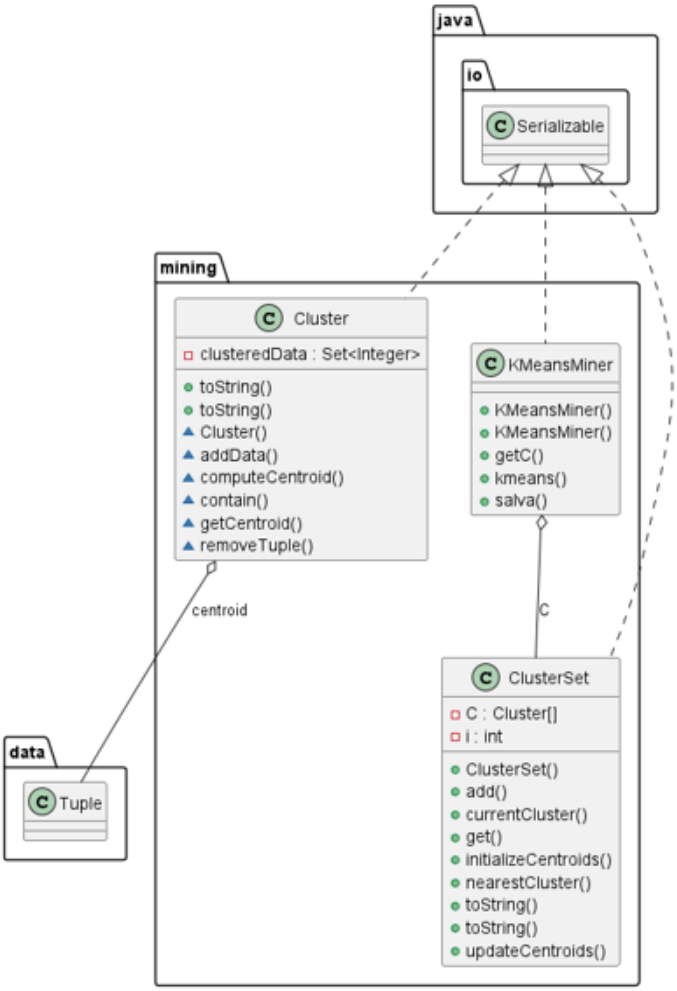

<<utility>> Keyboard
- in : BufferedReader
- reader : StringTokenizer
- current_token : String
- errorCount : int
- printErrors : boolean
+ readDouble() : double
+ readFloat() : float
+ readLong() : long
+ readInt() : int
+ readChar() : char
+ readBoolean() : boolean
+ readWord() : String
+ readString() : String
+ endOfLine() : boolean
- getNextInputToken(skip : boolean) : String
- getNextToken(skip : boolean) : String
- getNextToken() : String
- error(str : String) : void
+ setPrintErrors(flag : boolean) : void
+ getPrintErrors() : boolean
+ resetErrorCount(count : int) : void
+ getErrorCount() : int
ServerException
~ ServerException(result : String)
java.lang.Exception
MainTest
- in : ObjectInputStream
- out : ObjectOutputStream
+ main(args : String[]) : void
- storeClusterInFile() : void
- learningFromDbTable() : String
- storeTableFromDb() : void
- learningFromFile() : String
- menu() : int
+ MainTest(ip : String, port : int)
java.io.ObjectOutputStream
java.io.ObjectInputStream

3.2 Server UML

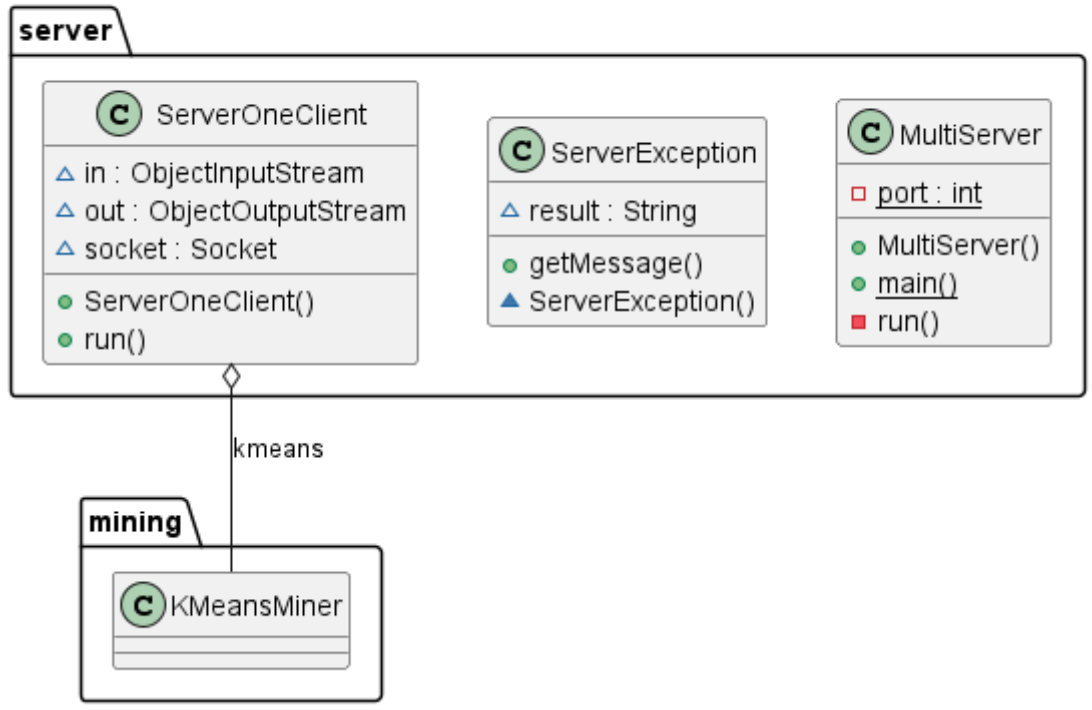


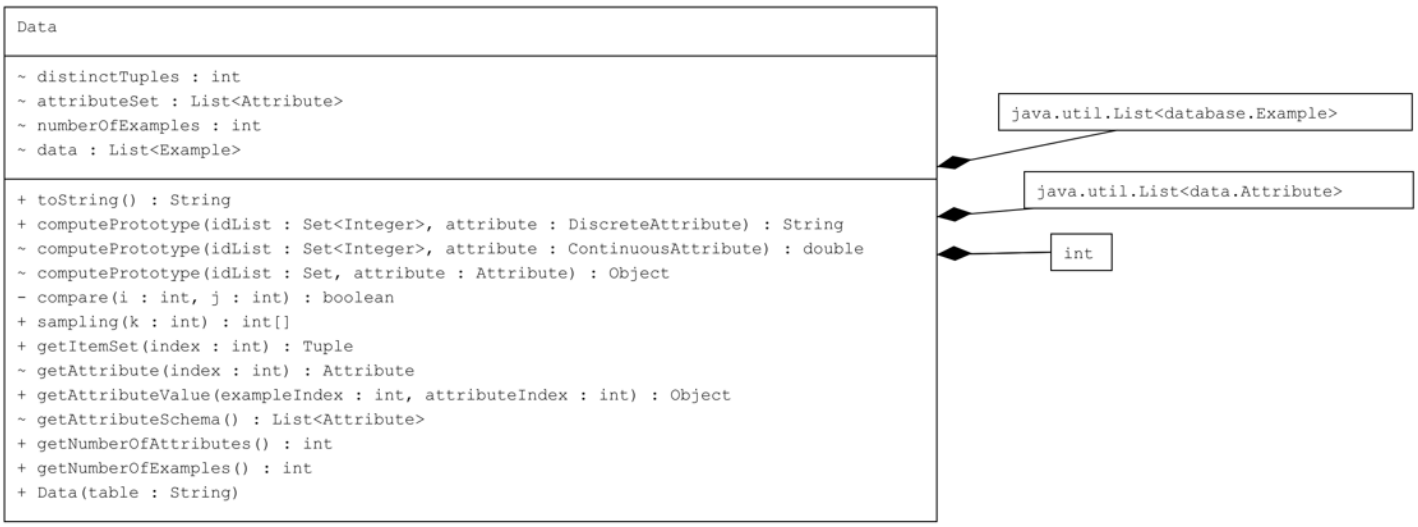

KMEANSSERVER's Class Diagram
java
io
Serializable
lang
Iterable
Comparable
server
ServerOneClient
ServerException
MultiServer
kmeans
mining
KMeansMiner
Cluster
C
centroid
ClusterSet
data
Item
OutOfRangeSampleSize
Data
Tuple
ase
QUERY_TYPE
database
DatabaseConnectionException
DbAccess
EmptySetException
attribute
ContinuousItem
DiscreteItem
Example
NoValueException
TableData
TableSchema
Attribute
ContinuousAttribute
DiscreteAttribute
database
TableData
TableSchema
Example
db
db
DbAccess

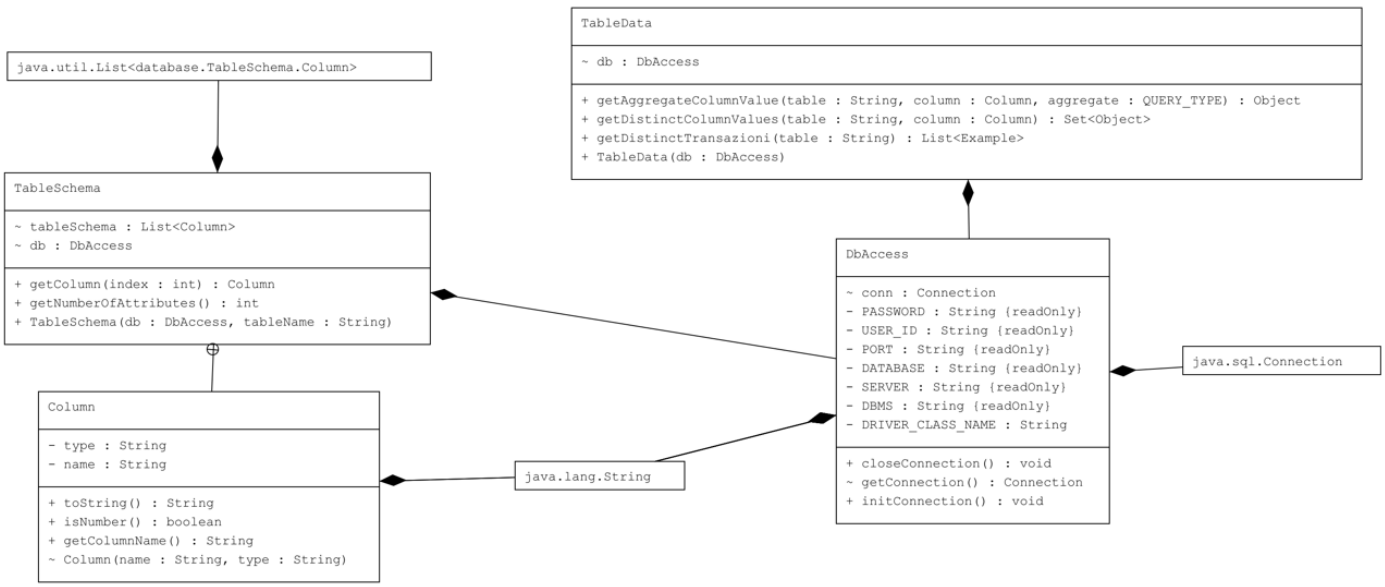
MINING's Class Diagram

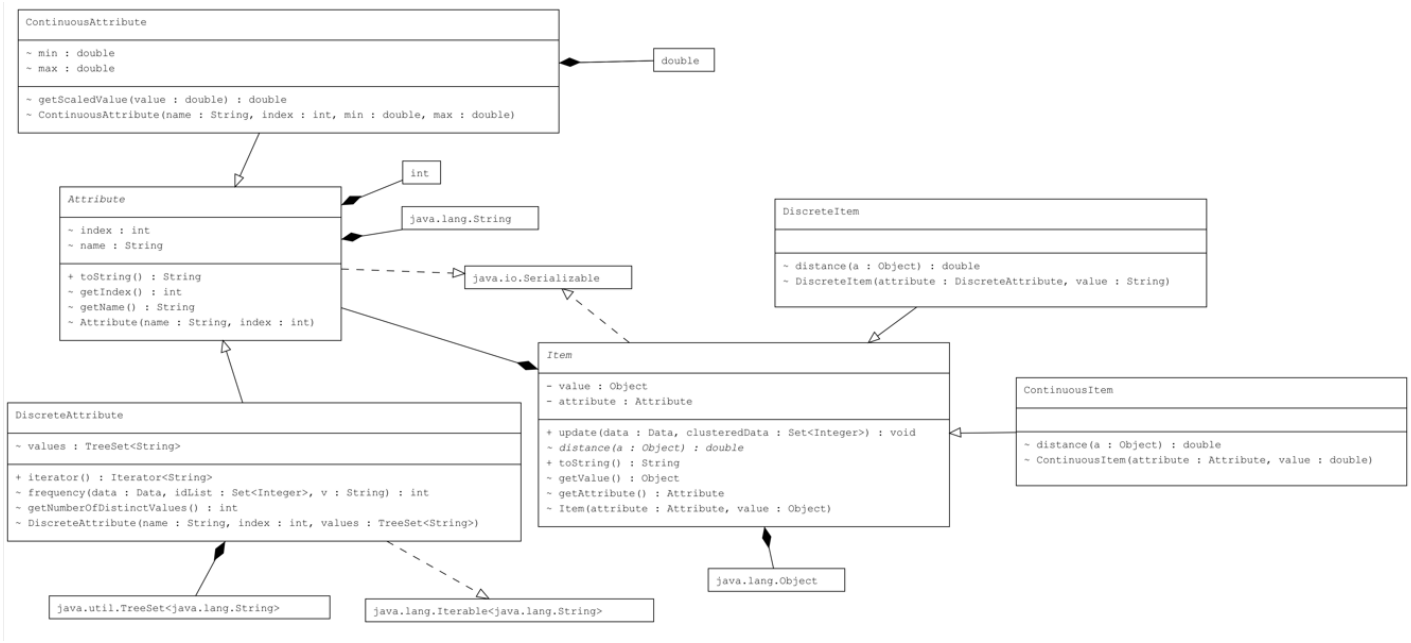


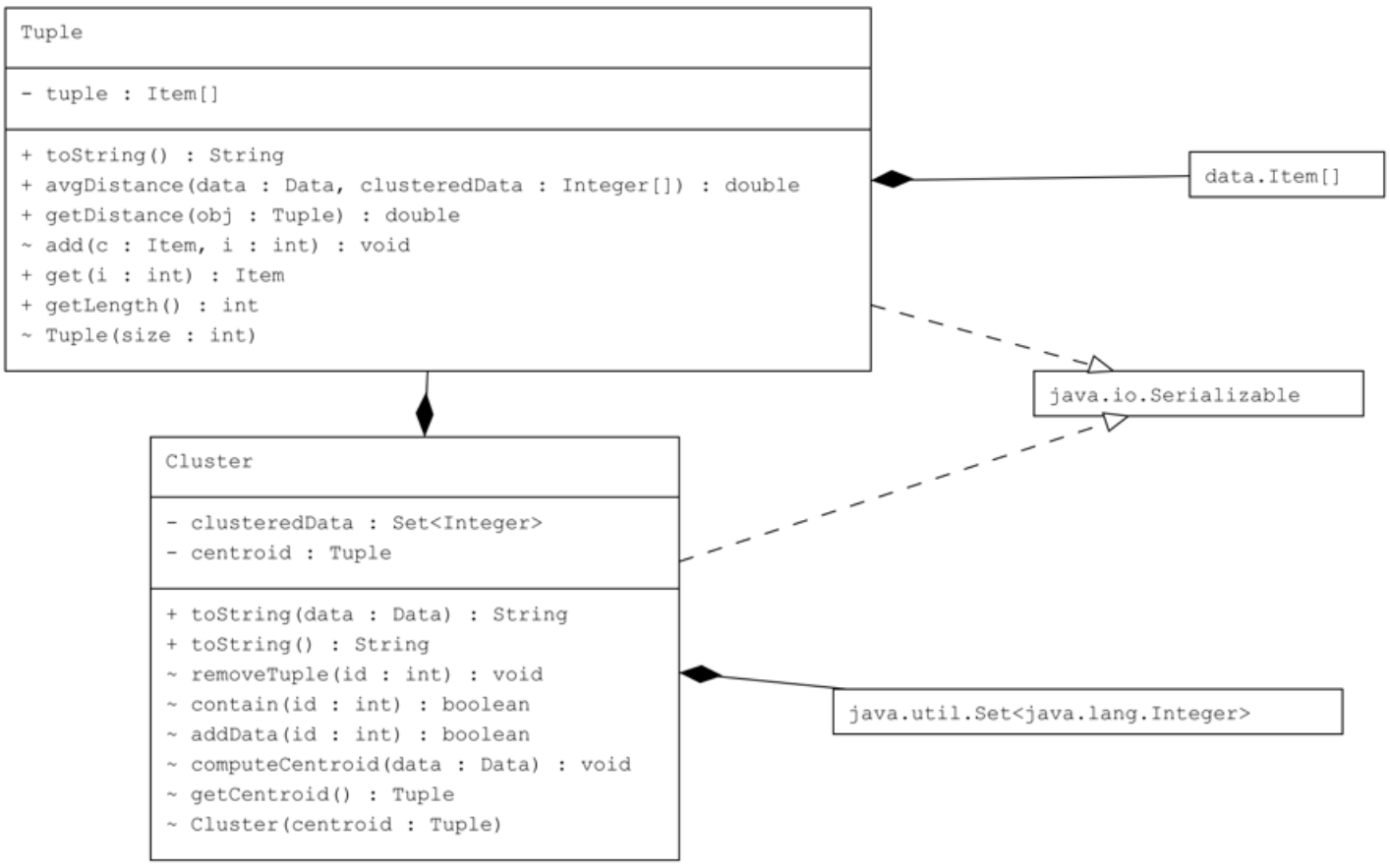
SERVER's Class Diagram




Data
~ distinctTuples : int
~ attributeSet : List<Attribute>
~ numberOfExamples : int
~ data : List<Example>
+ toString() : String
+ computePrototype(idList : Set<Integer>, attribute : DiscreteAttribute) : String
~ computePrototype(idList : Set<Integer>, attribute : ContinuousAttribute) : double
~ computePrototype(idList : Set, attribute : Attribute) : Object
- compare(i : int, j : int) : boolean
+ sampling(k : int) : int[]
+ getItemSet(index : int) : Tuple
~ getAttribute(index : int) : Attribute
+ getAttributeValue(exampleIndex : int, attributeIndex : int) : Object
~ getAttributeSchema() : List<Attribute>
+ getNumberOfAttributes() : int
+ getNumberOfExamples() : int
+ Data(table : String)
java.util.List<database.Example>
java.util.List<data.Attribute>
int


TableData
~ db : DbAccess
+ getAggregateColumnValue(table : String, column : Column, aggregate : QUERY_TYPE) : Object
+ getDistinctColumnValues(table : String, column : Column) : Set<Object>
+ getDistinctTransazioni(table : String) : List<Example>
+ TableData(db : DbAccess)
java.util.List<database.TableSchema.Column>
TableSchema
~ tableSchema : List<Column>
~ db : DbAccess
+ getColumn(index : int) : Column
+ getNumberOfAttributes() : int
+ TableSchema(db : DbAccess, tableName : String)
DbAccess
~ conn : Connection
- PASSWORD : String {readOnly}
- USER_ID : String {readOnly}
- PORT : String {readOnly}
- DATABASE : String {readOnly}
- SERVER : String {readOnly}
- DBMS : String {readOnly}
- DRIVER_CLASS_NAME : String
+ closeConnection() : void
~ getConnection() : Connection
+ initConnection() : void
java.sql.Connection
Column
- type : String
- name : String
+ toString() : String
+ isNumber() : boolean
+ getColumnName() : String
~ Column(name : String, type : String)
java.lang.String


ContinuousAttribute
~ min : double
~ max : double
~ getScaledValue(value : double) : double
~ ContinuousAttribute(name : String, index : int, min : double, max : double)
double
Attribute
~ index : int
~ name : String
+ toString() : String
~ getIndex() : int
~ getName() : String
~ Attribute(name : String, index : int)
int
java.lang.String
java.io.Serializable
DiscreteItem
~ distance(a : Object) : double
~ DiscreteItem(attribute : DiscreteAttribute, value : String)
Item
- value : Object
- attribute : Attribute
+ update(data : Data, clusteredData : Set<Integer>) : void
~ distance(a : Object) : double
+ toString() : String
~ getValue() : Object
~ getAttribute() : Attribute
~ Item(attribute : Attribute, value : Object)
ContinuousItem
~ distance(a : Object) : double
~ ContinuousItem(attribute : Attribute, value : double)
DiscreteAttribute
~ values : TreeSet<String>
+ iterator() : Iterator<String>
~ frequency(data : Data, idList : Set<Integer>, v : String) : int
~ getNumberOfDistinctValues() : int
~ DiscreteAttribute(name : String, index : int, values : TreeSet<String>)
java.lang.Object
java.util.TreeSet<java.lang.String>
java.lang.Iterable<java.lang.String>


Tuple
- tuple : Item[]
+ toString() : String
+ avgDistance(data : Data, clusteredData : Integer[]) : double
+ getDistance(obj : Tuple) : double
~ add(c : Item, i : int) : void
+ get(i : int) : Item
+ getLength() : int
~ Tuple(size : int)
data.Item[]
java.io.Serializable
Cluster
- clusteredData : Set<Integer>
- centroid : Tuple
+ toString(data : Data) : String
+ toString() : String
~ removeTuple(id : int) : void
~ contain(id : int) : boolean
~ addData(id : int) : boolean
~ computeCentroid(data : Data) : void
~ getCentroid() : Tuple
~ Cluster(centroid : Tuple)
java.util.Set<java.lang.Integer>

4. GUIDA ALL' INSTALLAZIONE

4.1 Installazione Server

Per il corretto funzionamento del progetto lato server è necessario:
- Spostare l'intera cartella del progetto sul desktop;
- Installare MySQL 8.0;
- Installare Java Runtime Environment (JRE) versione 20;
- Avviare il server MySQL;
- Eseguire lo script MySQL presente nella cartella "SQL Connector". Tale script inizializza il database con tabelle e tuple di esempio.[i]

Per avviare il server è possibile aprire il file "*Eseguibile Server.bat*" contenuto nella cartella *"Eseguibile/Base"*. Alternativamente, è possibile avviare il server tramite riga di comando indicando (parendo dalla cartella in cui si trova il file *Eseguibile/Base/KmeansServer.jar*):
- La directory in cui è contenuto il java.exe (se non è contenuto nel PATH)
- Il comando -jar che indica di avviare un file .jar

La riga sarà simile a:
C:\$PathTo$\java.exe -jar KMeansServer.jar

4.1 Installazione Client

Per il corretto funzionamento del progetto lato client è necessario:
- Installare Java Runtime Environment (JRE) versione 20;
- Avviare il server[ii]

Per avviare il client è possibile aprire il file *Eseguibile Client.bat* contenuto nella cartella *"Eseguibile/Base"*. Alternativamente, è possibile avviare il client tramite riga di comando indicando (partendo dalla cartella in cui si trova il file *Eseguibile/Base/KmeansClient.jar*):
- La directory in cui è contenuto il java.exe (se non è contenuto nel PATH)
- Il comando -jar che indica di avviare un file .jar
- L'indirizzo IP a cui è collegato il server (di default 127.0.0.1)
- La porta su cui è in ascolto il server (di default 8080)

La riga sarà simile a:
C:\$pathTo$\java.exe -jar KmeansClient.jar 127.0.0.1 8080

5. GUIDA UTENTE

Nella cartella principale del progetto è presente una sottocartella *"File memorizzati"*, nella quale verranno salvati (e caricati) in file *.ser* tutti i pattern trovati. In essa sono presenti già dei file a scopo di esempio
Le tabelle di esempio presenti nello script MySQL si chiamano *"playtennis"*.

5.1 Guida base alla interazione da console

Nella cartella *"Eseguibile"* eseguire il file *"Eseguibile generale Base.bat"*. Si apriranno due distinte schermate a linea di comando: una per il server e una per il client

1) Avvio server:

```
Started: ServerSocket[addr=0.0.0.0/0.0.0.0,localport=8080]
```

2) Avvio Client:

```
Socket[addr=/127.0.0.1,port=8080,localport=64291]
Scegli una opzione
(1) Carica Cluster da File
(2) Carica Dati
Risposta:
```

3) Carica Cluster da File

```
Risposta:1
Nome tabella:playtennis
Numero iterate:3
0:Centroid=(rain 7.5200000000000005 normal weak yes )
Examples:
[rain 12.0 normal weak yes ] dist=0.1478547854785478
[sunny 12.5 normal strong yes ] dist=2.1643564356435645
[rain 13.0 high weak yes ] dist=1.1808580858085809
[rain 0.0 normal weak yes ] dist=0.2481848184818482
[sunny 0.1 normal weak yes ] dist=1.2448844884488448

AvgDistance=0.9972277227722772
1:Centroid=(overcast 19.77 normal weak yes )
Examples:
[overcast 30.0 high weak yes ] dist=1.3376237623762375
[overcast 0.1 normal strong yes ] dist=1.6491749174917492
[overcast 29.21 normal weak yes ] dist=0.31155115511551157

AvgDistance=1.0994499449944994
2:Centroid=(rain 13.95 high strong no )
Examples:
[sunny 30.3 high strong no ] dist=1.5396039603960396
[rain 12.5 high strong no ] dist=0.04785478547854782
[rain 0.0 normal strong no ] dist=1.4603960396039604
[sunny 13.0 high weak no ] dist=2.0313531353135312

AvgDistance=1.2698019801980198

Vuoi scegliere una nuova operazione da menu?(y/n)
```

4) Carica Dati

```
Scegli una opzione
(1) Carica Cluster da File
(2) Carica Dati
Risposta:2
Nome tabella:playtennis
Numero di cluster:2
Clustering output:0:Centroid=(sunny 18.599999999999998 high strong no )
Examples:
[sunny 30.3 high strong no ] dist=0.38613861386138626
[sunny 13.0 high weak no ] dist=1.1848184818481848
[rain 12.5 high strong no ] dist=1.2013201320132012

AvgDistance=0.9240924092409241
1:Centroid=(rain 10.767777777777777 normal weak yes )
Examples:
[overcast 30.0 high weak yes ] dist=2.634726806013935
[rain 13.0 high weak yes ] dist=1.0736707004033736
[rain 0.0 normal weak yes ] dist=0.3553722038870553
[rain 0.0 normal strong no ] dist=2.3553722038870553
[overcast 0.1 normal strong yes ] dist=2.352071873854052
[sunny 0.1 normal weak yes ] dist=1.352071873854052
[rain 12.0 normal weak yes ] dist=0.04066740007334069
[sunny 12.5 normal strong yes ] dist=2.0571690502383575
[overcast 29.21 normal weak yes ] dist=1.6086541987532088

AvgDistance=1.5366418123293812

Vuoi ripetere l'esecuzione?(y/n)
```

In caso di assenza del file o della tabella, verrà stampato un messaggio di avviso e sarà possibile effettuare una nuova operazione

5) Casi particolari:

```
Socket[addr=/127.0.0.1,port=8080,localport=50826]
Scegli una opzione
(1) Carica Cluster da File
(2) Carica Dati
Risposta:2
Nome tabella:cicco
[404] La tabella cicco non esiste
Nome tabella:
```

```
addr = /127.0.0.1
Socket[addr=/127.0.0.1,port=8080,localport=50970]
Scegli una opzione
(1) Carica Cluster da File
(2) Carica Dati
Risposta:2
Nome tabella:playtennis
Numero di cluster:15
[!] Grandezza del cluster indicato troppo grande
Vuoi ripetere l'esecuzione?(y/n)
```

Per uscire dal programma, basterà digitare “n” o qualsiasi altra lettera oltre ad “y” oppure chiudere direttamente la schermata

NOTE

[i] In alternativa si può aprire con un editor di testo e copiare il contenuto nella shell MySQL
[ii] Per passare dalla versione base a quella estesa o viceversa, assicurarsi di utilizzare la giusta versione del server (\Base\). Se necessario, chiudere il server esteso prima di aprire il server base