Esercitazione 1 – Classi, Aggregazione, Interfaccie

Le visibilità di attributi, metodi e classi devono essere di volta in volta decise dallo Studente

- Definire la classe **Example**(nel package di default) che modella la entità esempio (inteso come vettore di valori reali).

class *Example {…}*

Membri Attributi

```
Double [] example; // vettore di valori reali
```

Membri Metodi

```
Example (int length)
```

Input: dimensione dell'esempio

Output: //

Comportamento: inizializza example come vettore di dimensione length

```
void set(int index, Double v)
```

Input: index: posizione del valore , v: valore

Comportamento: modifica example inserendo v in posizione index ;

```
Double get(int index)
```

Input: index: posizione di example

Output : valore memorizzato in example[index]

Comportamento: restituisce example[index];

```
double distance(Example newE)
```

Input: newE : istanza di Example

Output: calcola la distanza euclidea tra this.example e new.example

Comportamento: restituisce il valore calcolato;

```
public String toString()
```

Input: newE : istanza di Example

Output: la stringa che rappresenta il contenuto di example

Comportamento: restituisce la stringa;

- Definire la classe **Data** che aggrega la classe **Example**

Membri Attributi

```
Example data []; // che rappresenta il dataset
int numberOfExamples; // che rappresenta il numero di esempi nel dataset
```

Membri Metodi

Data()

Comportamento: Avvalora un oggetto data predefinito (fornito dal docente)

```
int getNumberOfExamples()
```

Output: numero di esempi memorizzati in data

Comportamento: restituisce numberOfExamples

```
Example getExample(int exampleIndex)
```

Input: indice di un esempio memorizzato in data

Output: l'esempio memorizzato in data[exampleIndex]

Comportamento: restituisce data[exampleIndex]

```
double [][] distance()
```

Input: //

Output: matrice triangolare superiore delle distanze Euclidee calcolate tra gli esempi memorizzati in data. Tale matrice va avvalorata usando il metodo distance di Example

Comportamento: restituisce la matrice triangolare superiore delle distanze

```
public String toString()
```

Input: //

Output: stringa che modella lo stato dell'oggetto

Comportamento: crea una stringa in cui memorizza gli esempi memorizzati nell'attributo data, opportunamente enumerati. Restituisce tale stringa

Si usi il metodo main in Data fornito dal docente per ottenere il seguente output:

```
0:[1.0,2.0,0.0]
1:[0.0,1.0,-1.0]
2:[1.0,3.0,5.0]
3:[1.0,3.0,4.0]
4:[2.0,2.0,0.0]

Distance matrix:

0.0    3.0    26.0   17.0   1.0
0.0    0.0    41.0   30.0   6.0
0.0    0.0    0.0    1.0    27.0
0.0    0.0    0.0    0.0    18.0
0.0    0.0    0.0    0.0    0.0
```

- Considerare la classe ***Cluster*** fornita dal docente che modella un cluster come la collezione delle posizioni occupate dagli esempi raggruppati nel Cluster nel vettore data dell'oggetto che modella il dataset su cui il clustering è calcolato (istanza di Data)
- Considerare l'interfaccia ***ClusterDistance*** e la classe ***SingleLinkDistance*** fornite dal docente. Scrivere la classe ***AverageLinkDistance*** che implementa l'interfaccia ***ClusterDistance*** e implementa la distanza average-link tra cluster
- Considerare la classe ***ClusterSet*** parzialmente fornita dal docente che modella un insieme di cluster

  Aggiungere a ClusterSet il metodo:

  ```
  ClusterSet mergeClosestClusters(ClusterDistance distance, Data data)
  ```

Input: distance: oggetto per il calcolo della distanza tra cluster; data: oggetto istanza che rappresenta il dataset in cui si sta calcolando l'oggetto istanza di ClusterSet

Output: nuova istanza di ClusterSet

Comportamento: determina la coppia di cluster più simili (usando il metodo distance di ClusterDistance e li fonde in unico cluster; crea una nuova istanza di ClusterSet che contiene tutti i cluster dell'oggetto ***this*** *a meno dei due cluster fusi al posto dei quali inserisce il cluster risultante dalla fusione (nota bene l'oggetto ClusterSet*

risultante memorizza un numero di cluster che è pari al numero di cluster memorizzato nell'oggetto this meno 1).

- Scrivere la classe ***Dendrogram*** che contiene:

  Membri Attributi

  ```
  private ClusterSet tree[]; // modella il dendrogramma
  ```

  Membri Metodi

  ```
  Dendrogram(int depth)
  ```

  Input: depth: profondità del dendrogramma

  Comportamento: crea un vettore di dimensione depth con cui inizializza tree.

  ```
  void setClusterSet(ClusterSet c, int level)
  ```

  Comportamento: memorizza c nella posizione level di tree

  ```
  ClusterSet getClusterSet(int level)
  ```

  Comportamento: restituisce tree[level]

  ```
  int getDepth()
  ```

  Comportamento: la profondità del dendrogamma (ossia la dimensione di tree)

  Aggiungere i metodi:

  ```
  public String toString() {
  	String v="";
  	for (int i=0;i<tree.length;i++)
  		v+=("level"+i+":\n"+tree[i]+"\n");
  	return v;
  }

  String toString(Data data) {
  	String v="";
  	for (int i=0;i<tree.length;i++)
  		v+=("level"+i+":\n"+tree[i].toString(data)+"\n");
  	return v;
  }
  ```

- Completare la classe ***HierachicalClusterMiner*** parzialmente fornita dal docente implementando il metodo:

  ```
  void mine(Data data, ClusterDistance distance)
  ```

  Comportamento: crea il livello base (livello 0) del dendrogramma che contiene l'istanze di ClusterSet che rappresenta ogni esempio in un cluster separato; per tutti i livelli successivi del dentrogramma (level>=1 e level < `dendrogram.getDepth()`*) costruisce l'istanza di ClusterSet che realizza la fusione dei due cluster più vicini nella istanza del ClusterSet memorizzata al livello level-1 del*

dendrogramma (usare `mergeClosestClusters` *di ClusterSet); memorizza l'istanza di ClusterSet ottenuta per fusione nel livello level del dendrogramma.*

Usare il main della classe ***MainTest*** *fornita dal docente per calcolare l'output mostrato nel file output.txt.*