Esercitazione 5 – RTTI, Serializzazione

*Definire la classe concreta **ContinuousItem** che estende la classe **Item** e modella una coppia <Attributo continuo - valore numerico> (e.g., Temperature=30.5)*

Metodi

ContinuousItem(Attribute attribute, Double value)

Output:

Comportamento: richiama il costruttore della super classe

***double** distance(Object a)*

Comportamento: Determina la distanza (in valore assoluto) tra il valore scalato memorizzato nello item corrente (this.getValue()) e quello scalato associato al parametro a. Per ottenere valori scalati fare uso di getScaledValue(...)

*Modificare la classe **Data** come segue:*

Metodi

public Data()

- *Comportamento: popolare l'insieme di transazioni (List<Example> data) iniziali considerando **Temperature** come attributo continuo e non più come discreto. Valori di esempio sono qui riportati:*

```
ex0.add(new Double (37.5));
ex1.add(new Double (38.7));
ex2.add(new Double (37.5));
ex3.add(new Double (20.5));
```

ex4.add(new Double (20.7));

ex5.add(new Double (21.2));

ex6.add(new Double (20.5));

ex7.add(new Double (21.2));

ex8.add(new Double (21.2));

ex9.add(new Double (19.8));

ex10.add(new Double (3.5));

ex11.add(new Double (3.6));

ex12.add(new Double (3.5));

ex13.add(new Double (3.2));

modificare la definizione dell'attributo Temperature in attributeSet come segue:

attributeSet.add(new ContinuousAttribute("Temperature",1, 3.2, 38.7));

Double computePrototype(Set<Integer> idList, ContinuousAttribute attribute)

*Comportamento: Determina il valore prototipo come media dei valori osservati per **attribute** nelle transazioni di **data** aventi indice di riga in **idList***

Object computePrototype(Set idList, Attribute attribute)

*Comportamento: usa lo RTTI per determinare se **attribute** riferisce una istanza di **ContinuousAttribute** o di **DiscreteAttribute**. Nel primo caso invoca computePrototype(idList, (ContinuousAttribute)attribute)*

altrimenti

computePrototype(idList, (DiscreteAttribute)attribute);

*Tuple getItemSet(**int** index)*

*Comportamento: Crea e un istanza di Tuple che modelli la transazione con indice di riga **index** in data. Restituisce il riferimento a tale istanza. Usare lo RTTI per distinguere tra **ContinuousAttribute** e **DiscreteAttribute** (e quindi creare nella tupla un **ContinuousItem** o un **DiscreteItem**)*

Modificare la classe KMeansMiner con l'aggiunta dei metodi per la serializzazione e de-serializzazione di C.

Aggiungere il costruttore:

public KmeansMiner(String fileName) throws FileNotFoundException, IOException, ClassNotFoundException

Input: percorso+ nome file

*Comportamento: Apre il file identificato da fileName, legge l'oggetto ivi memorizzato e lo assegna a **C**.*

public void salva(String fileName) throws FileNotFoundException, IOException

Input: percorso+ nome file

*Comportamento: Apre il file identificato da **fileName** e sarva l'oggetto riferito da C in tale file.*

*Implementare dove serve l'interfaccia **Serializable***

Testare usando la classe MainTest (modificata opportunamente dallo studente).