Exercise 5 - RTTI, Serialisation

*Define the concrete **Continuous Item** class that extends the **Item** class and models a pair <Continuous attribute - numeric value> (e.g., Temperature=30.5)*

Methods

ContinuousItem(Attribute attribute, Double value)

Output:

Behaviour: calls up the constructor of the super class

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

Behaviour: Determines the distance (in absolute value) between the scaled value stored in the current item (this.getValue()) and the scaled value associated with the parameter a. To obtain scaled values make use of getScaledValue(...)

*Modify the **Date** class as follows:*

Methods

public Data()

- *Behaviour: populate the initial set of transactions (List<Example> data) considering **Temperature** as a continuous attribute and no longer as discrete. Example values are given here:*

```
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));
```

modify the definition of the Temperature attribute in attributeSet as follows:

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

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

*Behaviour: Determines the prototype value as the average of observed values for **attributes in date** transactions with row index in **idList***

Object computePrototype(Set idList, Attribute attribute)

*Behaviour: uses the RTTI to determine whether **attribute** refers to an instance of **Continuous Attribute** or **DiscreteAttribute**. In the first case it invokes* `computePrototype(idList, (ContinuousAttribute)attribute)`

otherwise

`computePrototype(idList, (DiscreteAttribute)attribute);`

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

*Behaviour: Creates and instance of Tuple that models the transaction with row **index** in date. Returns the reference to that instance. Use RTTI to distinguish between **Continuous Attribute** and **DiscreteAttribute** (and thus create a **Continuous Item** or **DiscreteItem** in the tuple)*

Modify the KMeansMiner class by adding methods for serialisation and de-serialisation of C.

Add the constructor:

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

Input: path + file name

*Behaviour: Opens the file identified by fileName, reads the object stored there and assigns it to **C**.*

public void save(String fileName) throws FileNotFoundException, IOException

Input: path + file name

*Behaviour: Opens the file identified by **fileName** and will be the object referred to by C in that file.*

*Implement the **Serializable** interface where needed*

Test using the MainTest class (modified appropriately by the student).