Esercitazione 4 – Contenitori & Generics

- *Modificare la classe DiscreteAttribute come di seguito riportato:*
  - *modificare la dichiarazione del membro values in modo da usare un contenitore generics di tipo TreeSet<String>*
  - *la classe deve ora implementare l'interfaccia generics Iterable<String> e quindi fornire la realizzazione per il metodo public Iterator<T> iterator()*
  - *eliminazione del metodo String getValue(int i)*

- *Modificare la classe Data come di seguito riportato:*
  - *definire la inner class Example (inner in Data, visibilità friendly) per modellare ciascuna transazione. La classe implementa l'interfaccia generics Comparable<Example> e include i seguenti membri* **(decidere opportunamente la visibilità di ciascun membro di Example):**
    - *List<Object> example=new ArrayList<Object>(); // array di Object che rappresentano la singola transazione (o riga di una tabella)*
    - *void add(Object o) // aggiunge o in coda ad example*
    - *Object get(int i) // restituisce lo i-esimo riferimento collezionato in example*
    - *int compareTo(Example ex) // restituisce 0, -1, 1 sulla base del risultato del confronto. 0 se i due esempi includono gli stessi valori. Altrimenti il risultato del compareTo(...) invocato sulla prima coppia di valori in disaccordo.*
    - *public String toString()// restutuisce una stringa che rappresenta lo stato di example (fare uso di for-each)*

- *modificare la dichiarazione del membro attributeSet in modo da usare un contenitore generics di tipo List<Attribute> :*

List<Attribute> attributeSet = new LinkedList<Attribute>();

- *modificare la dichiarazione del membro data in modo da usare un contenitore generics List<Example> :*

  List<Example> data =new ArrayList<Example>();

- modificare la il costruttore di Data in modo da popolare data senza esempi duplicato. *A tale scopo usare un TreeSet come mostrato nel seguito*

```
public Data(){
        //data
        TreeSet<Example> tempData = new TreeSet<Example>();
        Example ex0=new Example();
        Example ex1=new Example();
        … // COMPLETARE
        ex0.add(new String ("sunny"));
        ex1.add(new String ("sunny"));
        … // COMPLETARE
        ex0.add(new String ("hot"));
        ex1.add(new String ("hot"));
        … // COMPLETARE
        ex0.add(new String ("high"));
        ex1.add(new String ("high"));
        … // COMPLETARE
        ex0.add(new String ("weak"));
        ex1.add(new String ("strong"));
        … // COMPLETARE
```

```
        ex0.add(new String ("no"));
        ex1.add(new String ("no"));
        … // COMPLETARE
        tempData.add(ex0);
        tempData.add(ex1);
        … // COMPLETARE
        data=new ArrayList<Example>(tempData);
        … // COMPLETARE
        // inizializza numberOfExamples
        … // COMPLETARE
        //inizializza explanatory Set
    }
```

- *rimuovere il metodo countDistinctTuples() : tale metodo non è più necessario dal momento che ora data non contiene sicuramente transazioni duplicate*

Modificare conseguentemente i metodi che utilizzano membri e metodi introdotti finora.

- *Rimuovere la classe ArraySet dal progetto e sostituire l'uso della stessa con il contenitore HashSet. Modificare quindi la dichiarazione di clusteredData come segue*

Set<Integer> clusteredData=new HashSet<Integer>().

E modificare dove e se necessario le classi nel progetto che usino clusteredData.