

Esercitazione 1 – Classi, Aggregazione, Ereditarietà, Classi Astratte

Si definiscano in Java le classi rappresentare gli attributi (classi *Attribute*, *ContinuousAttribute*, *DiscreteAttribute*) di una transazione (o tupla) e la collezione di transazioni (classe *Data*)

Le visibilità di classi e attributi devono essere definite dallo studente nel rispetto dell'information hiding

- Definire la classe astratta **Attribute** (nel package di default) che modella la entità attributo.

abstract class *Attribute* {...}

Membri Attributi

String **name**; // nome simbolico dell'attributo

int **index**; // identificativo numerico dell'attributo

Membri Metodi

Attribute(String name, **int** index)

Input: nome dell'attributo e identificativo numerico dell'attributo (primo, secondo ... attributo della tupla)

Output: //

Comportamento: inizializza i valori dei membri **name**, **index**

String getName()

Input:

Output : nome dell'attributo

Comportamento: restituisce *name*;

int getIndex()

Input:

Output : identificativo numerico dell'attributo

Comportamento: restituisce *index*;

public **String** toString()

Input:

Output : stringa rappresentante lo stato dell'oggetto

Comportamento: restituisce *name*;

- Definire la classe concreta **ContinuousAttribute** che estende la classe **Attribute** e modella un attributo continuo (numerico). Tale classe include i metodi per “scalare” l’attributo dal dominio dell’attributo all’intervallo $[0, 1]$ al fine da rendere confrontabili attributi aventi domini diversi

Membri Attributi

double *max*;

double *min* ;// rappresentano gli estremi dell'intervallo di valori (dominio) che l'attributo può reamente assumere nel dataset considerato.

Membri Metodi

ContinuousAttribute(String name, int index, double min, double max)

Input: nome, identificativo numerico, valore minimo e valore massimo dell'attributo nel suo dominio

Output : //

Comportamento: Invoca il costruttore della classe madre e inizializza i membri aggiunti per estensione

double getScaledValue(**double** v)

Input: valore dell'attributo da scalare

Output : valore scalato

Comportamento: Calcola e restituisce il valore scalato del parametro passato in input. Lo scaling ha come codominio l'intervallo [0,1]. Lo scaling di v è quindi calcolato come segue:

$$v'=(v-\min)/(\max-\min)$$

Definire la classe concreta **DiscreteAttribute** che estende la classe **Attribute** e rappresenta un attributo discreto (categorico)

Membri Attributi

String values[];// array di oggetti String, uno per ciascun valore del dominio discreto. I valori del dominio sono memorizzati in **values** seguendo un ordine lessicografico.

Membri Metodi

DiscreteAttribute(String name, **int** index, String values[])

Input: nome dell'attributo, identificativo numerico dell'attributo e array di stringhe rappresentanti il dominio dell'attributo

Output : //

Comportamento: Invoca il costruttore della classe madre e inizializza il membro **values** con il parametro in input.

int getNumberOfDistinctValues()

Input: //

Output : numero di valori discreti nel dominio dell'attributo

Comportamento: Restituisce la dimensione di *values*

String getValue(*int* i)

Input: posizione di un valore in *values*

Output : valore discreto in posizione "i" di *values*

Comportamento: Restituisce *values[i]*

- Definire la classe concreta *Data* per modellare l'insieme di transazioni (o tuple o anche denominati esempi)

Membri Attributi

Object *data* [][]; // una matrice $n \times m$ di tipo Object dove ogni riga modella una transazioni descritta dagli attributi riportati sulle colonne (*n*: righe, *m*:colonne

int numberOfExamples; // cardinalità dell'insieme di transazioni
(numero di righe in *data*)

Attribute *attributeSet* []; // un vettore degli attributi in ciascuna tupla
(schema della tabella di dati)

Membri Metodi

Data()

Input:

Output :

Comportamento: Inizializza la matrice *data* [][] con transazioni di esempio (in questo momento, 14 esempi e 5 attributi come riportato nella tabella sottostante);

Inizializza *attributeSet* creando cinque oggetti di tipo *DiscreteAttribute*, uno per ciascun attributo (nella tabella sottostante). Attenzione a modellare correttamente, nome, indice e dominio di ciascun attributo.

Inizializza *numberOfExamples*

Outlook	Temperature	Humidity	Wind	PlayTennis
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Overcast	Hot	High	Weak	Yes
Rain	Mild	High	Weak	Yes
Rain	Cool	Normal	Weak	Yes
Rain	Cool	Normal	Strong	No
Overcast	Cool	Normal	Strong	Yes
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Rain	Mild	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes
Overcast	Mild	High	Strong	Yes
Overcast	Hot	Normal	Weak	Yes
Rain	Mild	High	Strong	No

int getNumberOfExamples()

Input://

Output: cardinalità dell'insieme di transazioni

*Comportamento: restituisce *numberOfExamples**

int getNumberOfAttributes()

Input://

Output: cardinalità dell'insieme degli attributi

*Comportamento: restituisce la dimensione di *attributeSet**

Attribute[] getAttributeSchema()

Input: //

Output: restituisce lo schema dei dati

Comportamento: restituisce `attributeSet`

Object getValue(**int** exampleIndex, **int** attributeIndex)

Input: indice di riga , indice di colonna in riferimento alla matrice memorizzata in `data`

Output: valore assunto in `data` dall'attributo in posizione `attributeIndex`, nella riga in posizione `exampleIndex`

Comportamento: restituisce `data[exampleIndex][attributeIndex]`

public String toString()

Input: //

Output: stringa che modella lo stato dell'oggetto

Comportamento: crea una stringa in cui memorizza lo schema della tabella (vedi `attributeSet`) e le transazioni memorizzate in `data`, opportunamente enumerate. Restituisce tale stringa

■ *Si definisca un metodo `main` in `Data` che consenta il test delle classi implementate, in particolare che permetta la stampa dell'insieme di transazioni.*

Esempio di output:

Outlook,Temperature,Humidity,Wind Playtennis

1:sunny,hot,high,weak,no,

2:sunny,hot,high,strong,no,

3:overcast,hot,high,weak,yes,

4:rain,mild,high,weak,yes,

5:rain,cool,normal,weak,yes,

6:rain,cool,normal,strong,no,

7:overcast,cool,normal,strong,yes,

8:sunny,mild,high,weak,no,

9:sunny,cool,normal,weak,yes,

10:rain,mild,normal,weak,yes,

11:sunny,mild,normal,strong,yes,

12:overcast,mild,high,strong,yes,

13:overcast,hot,normal,weak,yes,

14:rain,mild,high,strong,no,