Progettazione del Software

Giuseppe De Giacomo
Dipartimento di Informatica e Sistemistica
Università di Roma "La Sapienza"
http://www.dis.uniroma1.it/~degiacomo

Esercitazione 4

Realizzazione di una implementazione dell'interfaccia Set del Collection Framework

L'interfaccia Set

```
public interface Set extends Collection {

    // Basic Operations
    int size();
    boolean isEmpty();
    boolean contains(Object element);
    boolean add(Object element);    // Optional
    boolean remove(Object element); // Optional
    Iterator iterator();

    // Bulk Operations
    boolean containsAll(Collection c);
    boolean addAll(Collection c);    // Optional
    boolean removeAll(Collection c); // Optional
    boolean retainAll(Collection c); // Optional
    void clear();                    // Optional

    // Array Operations
    Object[] toArray();
    Object[] toArray(Object[] a);
}
```

Struttura base di una classe

```
public class XXX  {

  // campi dati (rappresentazione degli oggetti XXX)

  // costruttori

  // funzioni proprie della classe

  // funzioni speciali ereditate da Object

  // funzioni non pubbliche ausiliarie

}
```

Realizzazione dell'interfaccia Set

```
import java.util.*;

public class InsiemeLista implements Set {

  // campi dati

  // costruttori

  // funzioni proprie della classe
  // (realizzazione delle funzioni di Set)

  // basic operations
  public int size() { ... }

  public boolean isEmpty() { ... }

  public boolean contains(Object e) { ... }

  public boolean add(Object e) { ... }

  public boolean remove(Object e) { ... }

  public Iterator iterator() { ... }

  . . .
```

Realizzazione dell'interfaccia Set (cont.)

```
  . . .
  // bulk Operations
  public boolean containsAll(Collection c) { ... }

  public boolean addAll(Collection c){ // opzionale - non supportata
    throw new UnsupportedOperationException("addlAll() non e' supportata");
  }
  public boolean removeAll(Collection c) { // opzionale - non supportata
    throw new UnsupportedOperationException("removeAll() non e' supportata");
  }
  public boolean retainAll(Collection c) { // opzionale - non supportata
    throw new UnsupportedOperationException("retainAll() non e' supportata");
  }
  public void clear() { // opzionale - non supportata
    throw new UnsupportedOperationException("clear() non e'  supportata");
  }

  // array operations
  public Object[] toArray() { ... }

  public Object[] toArray(Object[] a) { ... }

  // funzioni speciali ereditate da Object

  // funzioni ausiliarie
}
```

Scelta della rappresentazione degli oggetti InsiemeLista

```
import java.util.*;

class Lista {
    Object info;
    Lista next;
}

public class InsiemeLista implements Set {

  // campi dati
  protected Lista inizio;
  protected int cardinalita;
  protected Class elemClass;  // vogliamo creare un insieme di oggetti omogenei


  // costruttori

  // funzioni proprie della classe
  // (realizzazione delle funzioni di Set)
  . . .
```

Costruttore per InsiemeLista

```
import java.util.*;

class Lista {
    Object info;
    Lista next;
}

public class InsiemeLista implements Set {

  // campi dati
  protected Lista inizio;
  protected int cardinalita;
  protected Class elemClass;  // vogliamo creare un insieme di oggetti omogenei


  // costruttori
  public InsiemeLista(Class cl) {
    inizio = null;
    cardinalita = 0;
    elemClass = cl; // vogliamo creare un insieme di oggetti omogenei
  }

  // funzioni proprie della classe
  // (realizzazione delle funzioni di Set)
  . . .
```

Scelta del trattamento delle funzioni speciali

```
public class InsiemeLista implements Set, Cloneable {
                                        //^^^^^^^^^^
  // campi dati
  . . .
  // costruttori
  . . .
  // funzioni proprie della classe
  // (realizzazione delle funzioni di Set)
  . . .

  // funzioni speciali ereditate da Object
  public boolean equals(Object o) {
    // realizzare uguaglianza profonda
    // (verifica sugli elementi della lista che rappresenta l'insieme)
  }

  public Object clone() {
    // realizzare copia profonda
    // (copiare la lista che rappresenta l'insieme)
  }

  public String toString() { ... }

  // funzioni ausiliarie
}
```

Realizzazione delle funzioni: equals()

```
. . .
// campi dati
protected Lista inizio;
protected int cardinalita;
protected Class elemClass;
. . .
// funzioni speciali ereditate da Object
public boolean equals(Object o) {
  if (o != null && getClass().equals(o.getClass())) {
    InsiemeLista ins = (InsiemeLista)o;
    if (!elemClass.equals(ins.elemClass)) return false;
    // ins non e' un insieme del tipo voluto
    else if (cardinalita != ins.cardinalita) return false;
    // ins non ha la cardinalita' giusta
    else {
      // verifica che gli elementi nella lista siano gli stessi
      Lista l = inizio;
      while (l != null) {
        if (!appartiene(l.info,ins.inizio)) //appartiene(): funz. ausiliaria
          return false;
        l = l.next;
      }
      return true;
    }
  }
  else return false;
}
. . .
```

Realizzazione delle funzioni: clone()

```
. . .
// campi dati
protected Lista inizio;
protected int cardinalita;
protected Class elemClass;
. . .
// funzioni speciali ereditate da Object
. . .
 public Object clone() {
  try {
    InsiemeLista ins = (InsiemeLista) super.clone();
    // chiamata a clone() di Object che esegue la copia campo a campo;
    // questa copia e' sufficiente per i campi cardinalita e elemClass
    // ma non per il campo inizio del quale va fatta una copia profonda
    ins.inizio = copia(inizio); //copia() - funz. ausiliaria
    return ins;
  } catch(CloneNotSupportedException e) {
    // non puo' accadere perche' implementiamo l'interfaccia cloneable,
    // ma va comunque gestita
    throw new InternalError(e.toString());
  }
}
. . .
```

Realizzazione delle funzioni: toString()

```
. . .
// campi dati
protected Lista inizio;
protected int cardinalita;
protected Class elemClass;
. . .
// funzioni speciali ereditate da Object
. . .
public String toString() {
  String s = "{ ";
  Lista l = inizio;
  while (l != null) {
    s = s + l.info + " ";
    l = l.next;
  }
  s = s + "}";
  return s;
}
. . .
```

Realizzazione delle funzioni: size(), isEmpty(), contains()

```
. . .
// campi dati
protected Lista inizio;
protected int cardinalita;
protected Class elemClass;
. . .
// funzioni proprie della classe
// (realizzazione delle funzioni di Set)

// basic operations

public int size() {
  return cardinalita;
}

public boolean isEmpty() {
  return inizio == null;
}

public boolean contains(Object e) {
  if (!elemClass.isInstance(e)) return false;
  else return appartiene(e,inizio);
}
. . .
```

Realizzazione delle funzioni: add(), remove()

```
protected Lista inizio;
protected int cardinalita;
protected Class elemClass;
. . .
// basic operations
 public boolean add(Object e) {
   if (!elemClass.isInstance(e)) return false;
   else if (appartiene(e,inizio)) return false;
   else {
     Lista l = new Lista();
     l.info = e;
     l.next = inizio;
     inizio = l;
     cardinalita = cardinalita + 1;
     return true;
   }
 }
 public boolean remove(Object e) {
   if (!elemClass.isInstance(e)) return false;
   if (!appartiene(e,inizio)) return false;
   else {
     inizio = cancella(e,inizio);
     cardinalita = cardinalita - 1;
     return true;
   }
 }
. . .
```

Realizzazione delle funzioni: iterator()

```
public class InsiemeLista implements Set, Cloneable {

// campi dati
protected Lista inizio;
protected int cardinalita;
protected Class elemClass;
. . .
// funzioni proprie della classe
// (realizzazione delle funzioni di Set)// basic operations
. . .
public Iterator iterator() {
  return new IteratorInsiemeLista(this);  //nota!
}
. . .
```

L'interfaccia Iterator

```
public interface Iterator {
  boolean hasNext();
  Object next();
  void remove(); // Optional
}
```

Esempio di uso dell'interfaccia Iterator

```
class CleneteDiSet {
  . . .
  public static void StampaElementi(Set s) {
    Iterator it = s.iterator();
    while(it.hasNext()) {
      Object o = it.nex();
      System.out.println(o);
    }
  }
  . . .
}
```

La classe IteratorInsiemeLista

```
// Quanto segue deve stare nello stesso package di InsiemeLista

import java.util.*;

public class IteratorInsiemeLista implements Iterator {
  private Lista rif;

  public IteratorInsiemeLista(InsiemeLista ins) {
    rif = ins.inizio; //nota inizio e' accessibile perche'
                      //InsiemeLista e' nello stesso package!!!
  }

  // Realizzazione funzioni di Itarator
  public boolean hasNext() {
    return rif != null;
  }

  public Object next() {
    Object e = rif.info;
    rif = rif.next;
    return e;
  }

  public void remove() {
    throw new UnsupportedOperationException("remove() non e' supportata");
  }
}
```

Il codice

- InsiemeLista.java
- Main.java
- Main.java con HashSet