```
/*
 Uso di vector per supportare valori a componenti non modificabili, e a struttura dinamica.
 Questo uso è Possibile ma NON CONVENIENTE.
 SCONSIGLIATO: Implementazione complicata e inefficiente, occore copiare direttamente o
 ricorrendo a clone() (ugualmente inefficiente).
*/


import java.io.*;
import java.util.*;

interface RelazioneAPI <T1,T2>{ //One Java API for the type Relazione Immutable
    public boolean isIn1 (T1 e);
    public boolean isIn2 (T2 e);
    public Vector<T2> getAll2(T1 e);
//  public Vector<T1> getAll1(T2 e);
    public RelazioneAPI<T1,T2> add(T1 x, T2 y);

}

class RelazioneADT<T1,T2> implements RelazioneAPI<T1,T2>{
    private Vector<T1> left;
    private Vector<T2> right;

    public RelazioneADT(){
        left = new Vector<T1>();
        right = new Vector<T2>();
    }
    public boolean isIn1 (T1 e){
        for (int i=0; i<left.size(); i++) {
            if (left.get(i).equals(e)) return true;
        };
        return false;
    }
    public boolean isIn2 (T2 e){
        for (T2 a: right) {if (e.equals(a)) return true;}
        return false;
    }
    public Vector<T2> getAll2(T1 e){
        Vector<T2> r = new Vector<T2>();
        for (int i=0; i<left.size(); i++) {
            if (left.get(i).equals(e)) r.add(right.get(i));
        };
        return r;
    }
    public RelazioneAPI<T1,T2> add(T1 x, T2 y){
        RelazioneADT<T1,T2> rel = new RelazioneADT<T1,T2>();
        for (int i=0; i<left.size(); i++) {
            rel.left.add(left.get(i));
            rel.right.add(right.get(i));
        }
        rel.left.add(x); rel.right.add(y);
        return rel;
    }
}
```