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

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

class RelazioneADTP<T1,T2> implements RelazioneAPI<T1,T2>{
    private Vector<pair> R;

    public RelazioneADTP(){
        R = new Vector<pair>();
    }
    private boolean isEmpty(){
        return R.size()==0;
    }
    public boolean isIn1(T1 e){
        for(pair p: R){
            if (p.l.equals(e)) return true;
        }
        return false;
    }
    public Vector<T2> getAll2(T1 e){
        Vector<T2> r = new Vector<T2>();
        for(pair p: R){
            if (e.equals(p.l)) r.add(p.r);
        }
        return r;
    }
    public void add(T1 e1, T2 e2){
        R.add(new pair(e1,e2));
    }
    class pair{
        T1 l; T2 r;
        pair(T1 x, T2 y){
            l=x; r=y;
        }
    }
}

/* Esercizio 3.
 Siano RelazioneAPI un API per relazioni con le operazioni definite come sopra,
 e RelazioneADTP un ADT che la implementa. Si estenda RelazioneADT per definire
 un tipo FunRel per relazioni che sono funzioni finite su generici tipi ed aventi,
 in aggiunta alle operazioni di RelazioneAPI, l'operazione apply che applicata
 ad un valore del dominio della funzione restituisce l'immagine della funzione.
 L'operazione apply solleva eccezione Undefined se applicata su valori su cui
 è indefinita.
 L'operazione add solleva eccezione AlreadyDefinedException se si tenta di cambiare
 l'immagine di un valore definito.
 soluzione del 19/5/2015

class FunRel<T1,T2> extends RelazioneADTP<T1,T2>{
    public void add(T1 x, T2 y) throws AlreadyDefinedException {
        Vector<T2> r = getAll2(x);
        if(r.size()==0){super.add(x,y); return;}
        if(!r.get(0).equals(y)) throw new AlreadyDefinedException("add");
    }
```

```
    public T2 apply(T1 u) throws UndefinedException {
        Vector<T2> im = getAll2(u);
        if (im.size()==0) throw new UndefinedException("apply");
        return im.get(0);
    }
}

 La soluzione data mosta come l'esercizio NON PUO' AVERE SOLUZIONE. Infatti,
 l'operazione add di RelazioneAPI<T1,T2>, ovvero add RelazioneADTP<T1,T2> non
 può essere implementata, risp. overriden, con add che solleva un'eccezione.
 Sotto diamo una soluzione per un esercizio 3 dove la frase:
 "solleva eccezione AlreadyDefinedException"
 è cambiata in
 "lascia lo stato invariato"
 soluzione del 20/5/2015
*/


class UndefinedException extends Exception{
    UndefinedException(String x){
        super(x);
    }
}
class AlreadyDefinedException extends Exception{
    AlreadyDefinedException(String x){
        super(x);
    }
}


class FunRel<T1,T2> extends RelazioneADTP<T1,T2>{
    public void add(T1 x, T2 y) {
        Vector<T2> r = getAll2(x);
        if(r.size()==0) {super.add(x,y); return;}
    }
    public T2 apply(T1 u) throws UndefinedException {
        Vector<T2> im = getAll2(u);
        if (im.size()==0) throw new UndefinedException("apply");
        return im.get(0);
    }
}



/* Esecizio 5.
Mostrare un programma che utilizza le classi sopra definite per:
 1- definire una funzione ff: ff(2)=2, ff(3)=6. Quindi,
 2- applicare ad essa le operazioni opportune per:
      - calcolare e stampare ff(3)
      - calcolre getAll2 di ff con 3
      - estendere ff con ff(2)=5
      - calcolare e stampare ff(2)
*/

class Main{
    public static void main(String[] in){
        FunRel<Integer,Integer> ff = new FunRel<Integer,Integer>();
        ff.add(2,2);
        ff.add(3,6);
        try {
```

```
            System.out.println("ff(3)="+ff.apply(3));
        }
        catch (Exception e) {
            System.out.println("ff(3)=Undefind");
        }
        Vector<Integer> g = ff.getAll2(3);
        System.out.print("ff.getAll2(3)=[ ");
        for (int i=0; i<g.size(); i++)
            System.out.print(g.get(i).intValue()+" ");
        System.out.println("]");
        ff.add(2,5);
        try {
            System.out.println("ff(2)="+ff.apply(2));
        }
        catch (Exception e) {
            System.out.println("ff(2)=Undefind");
        }
    }
}
```