```
Esercizio 1 - Soluzione
(a)
module type DICT =
 sig type ('a,'b)dict
  val mk: 'a * 'b -> ('a,'b)dict
  exception IllegalArgumentException of string
end;;
(b)
module Dict =
(struct
   open List
   type ('a,'b)dict = ('a * 'b)list
 (* Presentazione Valore: Un valore ('a,'b)dict ha presentazione [k1/v1;...;kn/vn],
    dove n≥0 e' la size del valore, k1,...,kn sono le n differenti chiavi di accesso
    ai valori v1,...,vn rispettivamente.
    AF(c) = [k1/v1;...;kn/vn] quando c = [(k1,v1);...;(kn,vn)] && n≥0
    I(c) = c = [(k1,v1);...;(kn,vn)] => (ki=kj iff i=j)
    con: [,], &&, delimitatori di liste e operazioni OCaml
 *)
 (* definizione eccezioni pubbliche *)
 (* definizione delle operazioni ausiliarie e pubbliche *)
 end:TAB);;
(c)
   exception IllegalArgumentException of string
   let mk(k,v) = [(k,v)]
(d)
  let dom d = List.map fst d
  let add d (k,v) = if (List.mem k (dom d))
                    then raise (IllegalArgumentException("Dict.add"))
                    else (k,v)::d
  let rec get d k = match d with
        [] -> raise (IllegalArgumentException("Dict.get"))
        |(k1,v1)::ds when k = k1 -> v1
        |_::ds -> get ds k
  let rec set d k v = match d with
        [] -> raise (IllegalArgumentException("Dict.set"))
        |(k1,v1)::ds when k = k1 -> (k,v)::ds
        |d1::ds -> d1::(set ds k v)
   let rec key d v = match d with
        [] -> None
        |(k1,v1)::ds when v = v1 -> Some k1
        |_::ds -> key ds v


Esercizio 2 - Soluzione
(a)
    public Dict(A k, B v) throws IllegalArgumentException;
    public void add(A k, B v) throws IllegalArgumentException;
    public void rem(A k) throws IllegalArgumentException;
    public B get(A k) throws IllegalArgumentException;
    public void set(A k, B v);
    public A key(B v);
(b)
public class DictE<A,B> extends Dict<A,B>{
    private Vector<A> Dom;
    /*
     Presentazione Valore: la stessa della super classe Dict<A,B>
     AF(c) = AF(c.super())
     I(c) = forall k\in [A]:
            c.dom.contains(k) iff ((k!=null)&&(k.equals(key(v)) (for some v\in [B]))
    dove: [T] indica l'insieme dei valori di tipo T (vedi lezioni)
```

```
    */
}
(c)
    public DictE(A k, B v) throws IllegalArgumentException{
        super(k,v);
        Dom = new Vector<A>();
        Dom.add(k);
    }
    public void add(A k, B v) throws IllegalArgumentException{
        if (key(v)!=null) return;
        super.add(k,v);
        Dom.add(k);
    }
    public void rem(A k) throws IllegalArgumentException{
        if (!Dom.contains(k)) return;
        super.rem(k);
        Dom.remove(k);
    }
    public void set(A k, B v){
        if (key(v)!=null) return;
        super.set(A k, B v);
    }
(d)
    public Vector<A> dom() {
        ret = new Vector<A>();
        for(A k: Dom) ret.add(k);
        return ret;
    }
(e)
    public DictE<A,B> clone() throws CloneNotSupportedException{
        DictE<A,B> ret = (DictE<A,B) super.clone();
        ret.Dom = (Vector<A>) Dom.clone();
        return ret;
    }
```