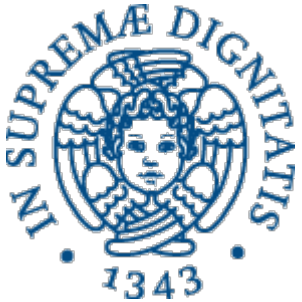

ESERCITAZIONE

Map<K,V>

- Dalla documentazione Java

  - public interface Map<K,V>
  - *An object that maps keys to values. A map cannot contain duplicate keys; each key can map to at most one value.*
  - **https://docs.oracle.com/javase/8/docs/api/java/util/Map.html**

Il metodo put di Map<K,V>

- **V put(K key, V value)**
  - associates the specified value with the specified key in this map (optional operation). If the map previously contained a mapping for the key, the old value is replaced by the specified value. (A map m is said to contain a mapping for a key k if and only if m.containsKey(k) would return true.)
- **Parameters**:
  - key - key with which the specified value is to be associated
  - value - value to be associated with the specified key
- **Returns**:
  - the previous value associated with key, or null if there was no mapping for key. (A null return can also indicate that the map previously associated null with key, if the implementation supports null values.)
- **Throws**:
  - UnsupportedOperationException - if the put operation is not supported by this map
  - ClassCastException - if the class of the specified key or value prevents it from being stored in this map
  - NullPointerException - if the specified key or value is null and this map does not permit null keys or values
  - IllegalArgumentException - if some property of the specified key or value prevents it from being stored in this map

Il metodo put: specifica

- **@param key**:
  - Il parametro key della coppia da inserire
- **@param value**:
  - Il parametro value of della coppia da inserire
- **@requires** *key != null*
- **@modifies** this
- **@effects** <key, value> viene inserita in this
  - se una associazione per key era già presente in this si sostituisce il vecchio campo valore con il valore del parametro value
- **@return** il vecchio valore associatao con key, se era presente una associazione, altrimenti null
- **@throws**: non le discutiamo

Map<K,V>: implementazione

- Supponiamo di implementare l'astrazione Map con una coppia di liste

```
public class pr2Map<K,V> {
// instance variables
  private List<K> keys;
  private List<V> values;
  // costruttore
  public pr2Map( ) {
    keys = new ArrayList<K>( );
    values = new ArrayList<V>( );
  }
```

ArrayList

- public class ArrayList<E>
  extends AbstractList<E>
  implements List<E>, RandomAccess, Cloneable, Serializable

- Resizable-array implementation of the List interface. Implements all optional list operations, and permits all elements, including null. In addition to implementing the List interface, this class provides methods to manipulate the size of the array that is used internally to store the list. (This class is roughly equivalent to Vector, except that it is unsynchronized.)

- http://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html

Map<K,V>

- ***Clausola Overview***
  - *collezione modificabile, non ordinata di coppie <key, value> che implementano una funzione tra insiemi di chiave e valori*
  - elemento tipico: ovvio
- **Nota**: implementare una funzione implica che nella struttura non esistono due coppie della forma <k,v1>, <k,v2>

Funzione di astrazione

- pr2Map è un insieme di coppie <key,value>.

- AF= {<key,value> tali che
  for 0 <= i < keys.size( ),
  keys.get(i)= key && values.get(i) = value}

Invariante di rappresentazione

- keys != null &&
  values != null &&
  keys.size( ) == values.size( ) &&
  for 0 <= i < keys.size( ),
  <keys.get(i), values.get(i)> in AF(c) &&
  for 0 <= i, j < keys.size( ),
  ! (i = j) ➔ !(keys(i).equals(keys(j)))

Invariante di rappresentazione

- Da aggiungere all'invariante per essere maggiormente realistici
  - gli elementi di key devono essere diversi da null
  - gli elementi di value devono essere diversi da null

Implementazione di put


```
public V put(K key, V value) {
  int loc = indexOf(key);
  if (loc != -1) {
    V result = values.get(loc);
    values.set(loc, value);
    return result;
  }
  keys.add(key);
  values.add(value);
  return null;
}
private int indexOf(K key) {
  for (int i = 0; i < keys.size (); i++)
     if (key.equals(keys.get(i)))
          return i;
  return -1;
}
```

put

- Preserva invariante di rappresentazione?
- Due casi da considerare. Quali?

GetKey()

- Implementazione

```
public List<K> getKeys( ) {
        return keys;
}
```

Debugging dell'astrazione

- La rappresentazione è visibile all'esterno dell'astrazione?
  - si: il metodo getKeys( ) restituisce come risultato la lista delle chiavi presenti nell'astrazione
  - il codice cliente può pertanto operare direttamente sulle chiavi
- Soluzione: restituire una coppia delle chiavi tramite l'introduzione di un metodo **clone** che effettua una deep copy
  - implementare clone( ) per esercizio

Estensione

- Aggiungere un metodo che permette di ottenere il valore associato ad una chiave
- Signature
  - public V get(K key)
- Come viene specificato?


```
/** SPEC A
* @requires key diverso da null & key è presente in this
* @effects restituisce il valore associato con la chiave key
*/
/** SPEC B
* @key è presente in this
* @effects restituisce il valore associato con la chiave key
* @throws NullPointerException se key è null
*/
/** SPEC C
* @effects restituisce il valore associato con la chiave key se
* key è presente in this
* oppure null se key non ha associato alcun valore in this
*/
/** SPEC D
* @effects restituisce il valore associato con la chiave key
* @throws NullPointerException se key è null
* /
```


```
// implementazione 1:
public V get(K key) {
  return values.get(indexOf(key));
}

// implementazione 2:
public V get(K key) {
  if (key==null)
    throw new NullPointerException("null key passed to get");
  return values.get(indexOf(key));
}
```


```
// implementazione 3:
public V get(K key) {
  if (key == null || indexOf(key) == -1)
    return null;
  return values.get(indexOf(key));
}

// implementazione 4:
public V get(K key) {
  if (key == null)
    throw new NullPointerException("null key passed to get");
  if (indexOf(key) == -1)
    throw new NoSuchElementException("key not found");
  return values.get(indexOf(key));
}
```

Verifica specifica

	IMPL1	IMPL 2	IMPL 3	IMPL 4
SPEC A				
SPEC B				
SPEC C				
SPEC D				

Verifica specifica

	IMPL1	IMPL 2	IMPL 3	IMPL 4
SPEC A	OK			
SPEC B	OK	OK		
SPEC C	OK		OK	
SPEC D	OK	OK		OK

Testing per Put?