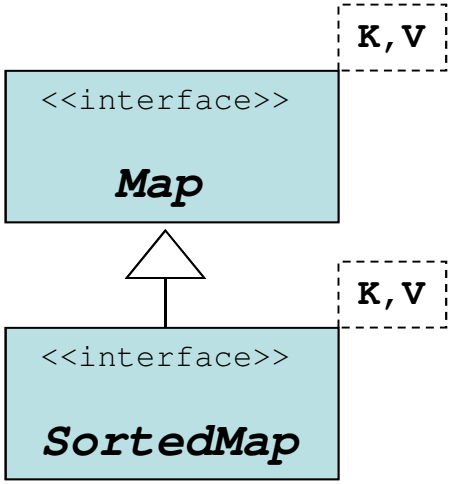
Map

Map

- L'interfaccia Map modella insiemi sui cui elementi si può fare una ricerca per chiave.

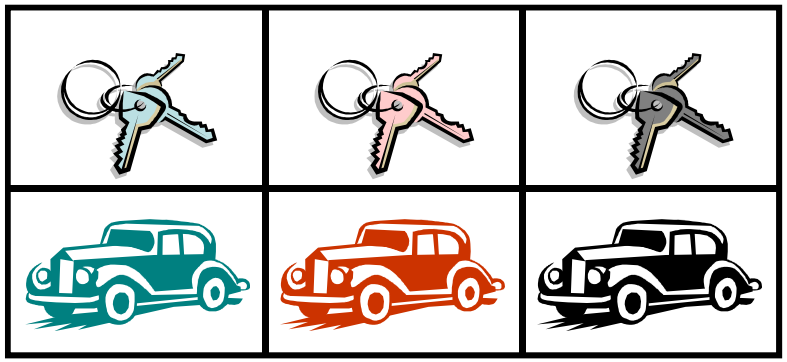
```
public interface Map<K,V>
```



- L'elemento di una Map si dice **entry** della mappa ed è modellato con un oggetto di tipo `Map.Entry<K,V>`
- Esso rappresenta la coppia Key,Value

Regole delle entries

- Ogni oggetto V possiede una sola chiave K
- Non esistono oggetti V con la stessa chiave K



Map.Entry

- E' un interfaccia che modella la coppia K, V della mappa
- Ogni mappa concreta dispone di una enmtry concreta che sarà implementazione di **`Map.Entry`**

- L'oggetto Map.Entry non può venire istanziato, ma viene caricato sulla mappa specificando K e V.

- Ogni entry della mappa dispone di :

1. **`Object getKey()`** → restituisce la chiave K
2. **`Object getValue()`** → restituisce il valore V
3. **`Object setValue(Object o)`** → sostituisce il campo valore alla entry, lasciando invariata la chiave K

Non esiste **`setKey`** perché la chiave non può essere modificata

Viste di Map

- Una mappa può essere riguardata come:
  - Set di chiavi
  - Collection di valori
  - Set di oggetti chiave-valore

  3 viste → 3 metodi:
  Set keySet(); Collection values(); Set entrySet()

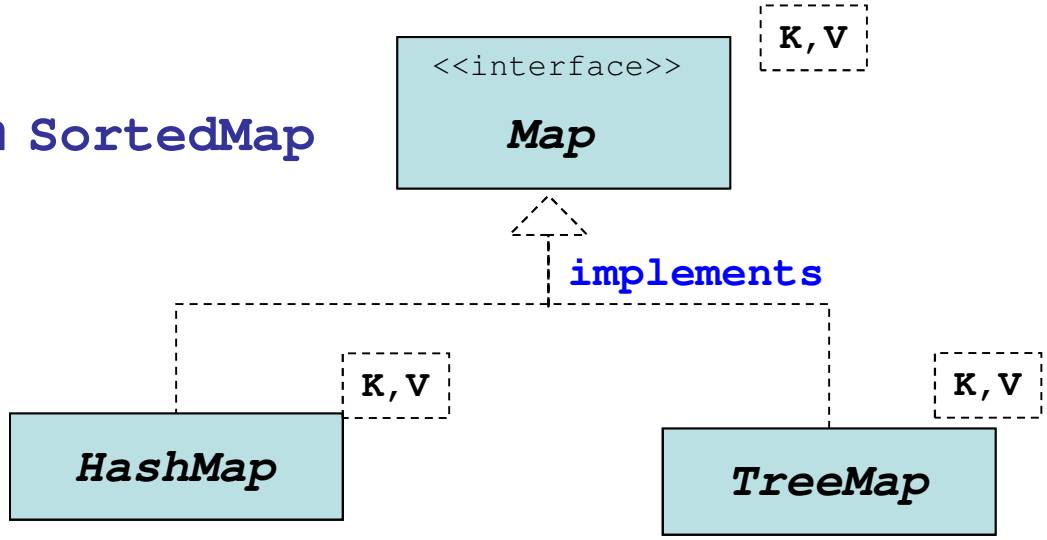
Esempio di navigazione di un Map:

```
Set entries = personale.entrySet();
Iterator iter = entries.iterator();
while(iter.hasNext()){
        Map.Entry entry = (Map.Entry)iter.next();
        Object key = entry.getKey();
        Object value = entry.getValue();
}
```

Implementazioni

Le principali classi che implementano **`Map`** sono:

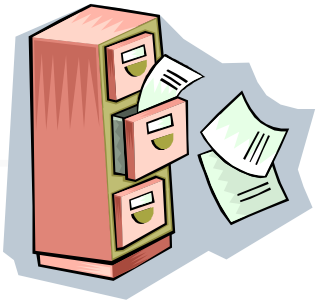
- **`HashMap<K,V>`**
- **`TreeMap <K,V>`**
  - che è anche una **`SortedMap`**



Metodi comuni:

- **`V`** **`put(K key, V value)`** → carica sulla mappa
- **`V`** **`get(K key)`** → ricerca per chiave, torna null se non trova!
- **`boolean`** **`containsKey(K key)`** → verifica la chiave
- **`boolean`** **`containsValue(V value)`** → verifica il valore

Class HashMap



- E’ utilizzato per la gestione di Map
  non ordinati, utilizza Hash table

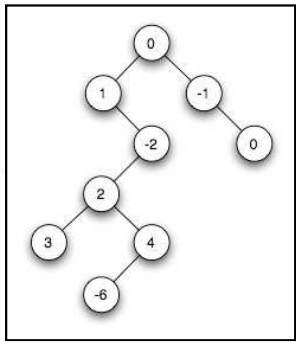
- Le **chiavi** relative agli oggetti della mappa dovrebbero ridefinire il metodo hashCode in modo che

  2 oggetti uguali per equals devono produrre lo stesso hashcode

  e il metodo equals – per distinguere le chiavi doppie

- Per ottimizzare le operazioni, la struttura dati che lo rappresenta è un array i cui elementi sono liste, ciascuna lista è un *bucket*
- Il numero di elementi (buckets) e il fattore di carico per ciascuno si possono impostare col costruttore
  (valori di default sono: initialCapacity = 16 e loadFactor = 0.75)

Class TreeMap



- E’ utilizzato per la gestione di Map ordinati e utilizza una struttura ad albero

- Le **chiavi** relative agli oggetti della mappa **devono** fornire il metodo compareTo dell'interfaccia standard Comparable

oppure

- si devono prevedere (per utilizzare criteri diversi) delle classi di tipo Comparator per definire un’ordine delle chiavi

- Per i tipi wrapper e per le stringhe è già definito un ordine naturale

- Indipendentemente da come vengono inseriti nell’insieme, vengono ciclati secondo l’ordine delle chiavi
- Tempo di esecuzione di `containsKey, get, put e remove` è ***log(n)***