Collections of Pairs: Maps

Richard Warburton

@richardwarburto | www.insightfullogic.com

pluralsight

— 256 —

Kvartal *n* (*pl* -er) trimestre *m*; terme *m*

Kvarter *n* (*pl* -er) quart *m* d'heure; quartier *m* (*mil.* et ville); quart *m* d'aune

Kvast *c* (*pl* –e et –er) houppe *f*

kvik *a* vif; éveillé

Kvinde *c* (*pl* –r) femme *f*

†**Kvisle** *c* (*pl* –r) branche *f* de rivière

Kvist *c* (*pl* –e) 1. petite branche; brindille *f*; 2. mansarde *f*

kvit *a* quitte [tance

kvittere acquitter; donner quit-

Kvittering *c* (*pl* –er) quittance *f*

Kvæg *n* bétail *m*; bestiaux *m*/*pl*

Kvægsølv *n* mercure *m* (= Kviksølv *n*)

kvæle étrangler; étouffer; suffoquer

— 257 —

Kvælstof *n* (gaz) azote *m*; nitrogène *m* *chem*

kvæste contusionner; **Kvæstning** *c* (*pl* –er) contusion *f*

Kylling *c* (*pl* –er) poulet *m*; poussin *m*

Kyndelmisse *c* la Chandeleur; la Purification (2 févr)

Kyper *c* (*pl* –e) tonnelier *m*; encaveur *m*

Kys *n* (*pl* –) baiser *m*

kysk *a* chaste; **K–hed** *c* chasteté *f*

Kyst *c* (*pl* –er) côte *f*; rivage *m*; bord *m*

Kæde *c* (*pl* –r) chaîne *f* (aussi tissure); collier *m*; suite *f* *fig*

kæk *a* hardi; audacieux; **K–hed** *c* hardiesse *f*; audace *f*

Kælder *c* (*pl* –e) cave *f*; **– etage** *c* sous-sol *m*; souterrain *m*

60 9

pluralsight

Key → Value

Outline

- Why use a Map?
- Views over Maps
- Sorted & Navigable Maps
- Java 8 Enhancements
- Implementations
- Summary

pluralsight

Why use a Map?

Motivation and API Overview

```
V put(K key, V value)

void putAll(Map<? extends K, ? extends V> values)
```

Adding & Replacing

`put` for a single value, `putAll` for another Map.

`null` keys and values are implementation specific

```
V get(Object key)
boolean containsKey(Object key)
boolean containsValue(Object value)
```

Looking Up Elements

Objects to allow more flexible generics contracts

```
V remove(Object key)

void clear()
```

Removing

```
int size()

boolean isEmpty()
```

Querying the size

Same semantics as on `Collection`

`Collection` and `Map`

`Map` is the only collections that don't extend or implement the `Collection` interface.

Views over Maps

keySet(), values(), entrySet()

Sorted and Navigable Maps

Traversal in Key Ascending Order

`SortedMap` superseded by `NavigableMap`

See `SortedSet` & `NavigableSet`

pluralsight

```
K firstKey();
K lastKey();

SortedMap<K, V> tailMap(E fromKey);
SortedMap<K, V> headMap(E toKey);
SortedMap<K, V> subMap(K fromKey, K toKey);
```

SortedMap

Defines an interface for a map with ordering

Subviews based upon key.

```
Map.Entry<K,V> firstEntry();
Map.Entry<K,V> lastEntry();

Map.Entry<K,V> pollFirstEntry();
Map.Entry<K,V> pollLastEntry();
```

First/Last Entries

Poll methods remove element as well as returning it.

```
Map.Entry<K,V> lowerEntry(K key);
Map.Entry<K,V> higherEntry(K key);

K lowerKey(K key);
K higherKey(K key);
```

Navigating by key

Allows moving to a lower/higher element in the map.

```
Map.Entry<K,V> floorEntry(K key);
Map.Entry<K,V> ceilingEntry(K key);

K floorKey(K key);
K ceilingKey(K key);
```

Navigating by key

Allows moving to a less than or equal/greater than or equal element in the map.

```
NavigableMap<K, V> descendingMap()
NavigableSet<K> descendingKeySet()

NavigableSet<K> navigableKeySet()
```

Reversing the order

Can't override `keySet()` due to backwards compatibility concerns.

```
NavigableMap<K, V> tailMap(E fromKey, boolean incl);
NavigableMap<K, V> headMap(E toKey, boolean incl);

NavigableMap<K, V> subMap(K fromKey,
  boolean frominclusive, K toKey, boolean toInclusive);
```

NavigableMap views

Java 8 Enhancements

pluralsight

Altering and Removing

replace(key, value)

replaceAll(BiFunction <K, V, V>)

remove(key, value)

Updating Values

getOrDefault

putIfAbsent

compute

computeIfAbsent

computeIfPresent

merge

`forEach` – callback based iteration

Implementations

Different approaches and performance tradeoffs

General Purpose Maps

HashMap | LinkedHashMap | TreeMap

`HashMap`

- Good general purpose implementation
- Uses the `.hashcode()` method (just like `HashSet`)
- Maintains an array of buckets
  - `hash % bucket_count`
- Buckets are linked lists to accommodate collisions
- Buckets can be trees
- The number of buckets increases with more elements

Map Visualiser

https://github.com/RichardWarburton/map-visualiser

TreeMap

Implemented using red-black tree

A balanced Binary Tree

Navigable and Sorted

Uses comparable/comparator to define the order

LinkedHashMap

Based Upon `HashMap`

Maintains An Order

Either Insertion, or Access

```
protected boolean removeEldestEntry(
    Map.Entry<K,V> eldest)
```

Helpful for implementing Caches

Called by the `put` and `putAll` methods

WeakHashMap

Weak references keys

Can be removed when unreachable

Used as a cache

pluralsight

`IdentityHashMap` vs `HashMap`

`IdentityHashMap`

`==`

`System.identityHashCode()`

Faster/Less Memory

Low collision likelihood

Intentionally violates Map contract

Useful for serialisation/graph traversal

HashMap

`obj.equals()`

`obj.hashCode()`

Use normally situations

Avoids coupling map to key implementation

EnumMap

Use if you have keys that are enums

Faster than other maps

Implementation based upon bitsets

Stores a single long for <= 64 elements

Algorithmic Performance

	put	get/containsKey	next
`HashMap`	O(N), Ω(1)	O(log N), Ω(1)	O(Capacity/N)
`LinkedHashMap`	O(N), Ω(1)	O(log N), Ω(1)	O(Capacity/N)
`IdentityHashMap`	O(N), Ω(1)	O(N), Ω(1)	O(Capacity/N)
`TreeMap`	O(log N)	O(log N)	O(log N)
`EnumMap`	O(1)	O(1)	O(1)

pluralsight

Summary

Summary

Maps associate keys and values

5 key implementations

API still improving in Java 8

Whatever you need, Java has you covered