**JAC444 - Lecture 9**

Java Collections Segment 4 - Map

**The Map<K,V> Interface**

A **Map** is an object that maps keys to values.

**A map cannot contain duplicate keys** – The collection of keys is a set

**Each key can map to at most one value** – Mathematical *function abstraction*

# Implementations:

# **HashMap** Hashtable and the constant-time implementation

**TreeMap** The map is sorting according to the natural ordering of its keys

**LinkedHashMap** Hashtable with linked list implementation of **Map** interface

**The Map<K,V> Interface**

|  |  |
| --- | --- |
| **public interface Map<K,V> {**  **// Basic Operations**  **Object put(K key, V value);**  **V get(Object key);**  **Object remove(Object key); boolean containsKey(Object key); boolean containsValue(Object value);**  **int size(); boolean isEmpty();**  **// Bulk Operations**  **void putAll(Map<? extends K, ? extends V> m); void clear();**  **// Collection Views public Set<K> keySet(); public Collection<V> values(); public Set<Map.Entry<K,VL> entrySet();**  **// Interface for entrySet elements public interface Entry<K,V> {**  **K getKey();**  **V getValue();**  **V setValue(V value);**  **}**  **}** | **1**  **2**  **3**  **4** |

Basic

Bulk

View

Entry

Interface

**Collection Views of Map<K,V>**

# The Collection view methods allow a Map to be viewed as a Collection in these three ways: **keySet** — the **Set** of keys contained in the **Map values** — the **Collection** of values contained in the **Map**

# **entrySet** — the **Set** of key-value pairs contained in the **Map**

# The **Map** interface has a nested interface called **Map.Entry**

# The standard **Map** conversion constructor. If there is an object **m** of type **Map**

# **Map<K, V> copy = new HashMap<K, V>(m)** creates an object **copy** of type **HashMap** that contains the same key-value as **m**

**Basic Map Operations**

**import java.util.\*; public class Rate {**

**public static void main(String[] args) { Map<String, Integer> m = new HashMap<>();**

**for (String key : args) { Integer val = m.get(key); Integer newVal = (val == null) ? 1 : val + 1;**

**m.put(key, newVal); }**

**for (Map.Entry<String, Integer> e : m.entrySet()) System.out.println(e.getKey() + "---> " + e.getValue());**

**}**

**}**

**Common Idioms for Map<K,V>**

# How to check if two maps objects **m1** and **m2** have the same keys:

**Answer: m1.keySet().equals(m2.keySet()**

# How to find the common keys of two maps objects **m1** and **m2**

**Answer: Set<K> commonKeys = new HashSet<K>(m1.keySet()); commonKeys.retainAll(m2.keySet());**

# Iterating over key-value pairs:

**for (Map.Entry<K, V> e : m.entrySet())**

**System.out.println(e.getKey() + ": " + e.getValue());**

**Map Implementations**

To build an object of type Map one needs to use the implementations without exposing the implementation

Idioms:

Map<K,V> m1 = new HashMap<K,V>();

Map<K,V> m2 = new TreeMap<K,V>();

Map<K,V> m2 = new LinkedHashMap<K,V>();