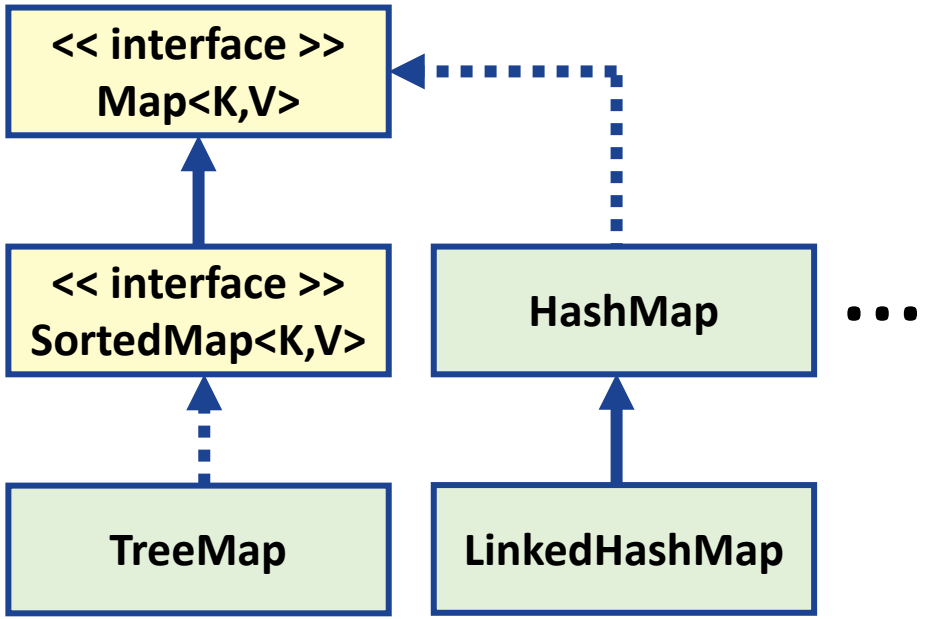
1495 UNIVERSITY OF ABERDEEN

JC2002 Java Programming

Lecture 36: Maps and comparators

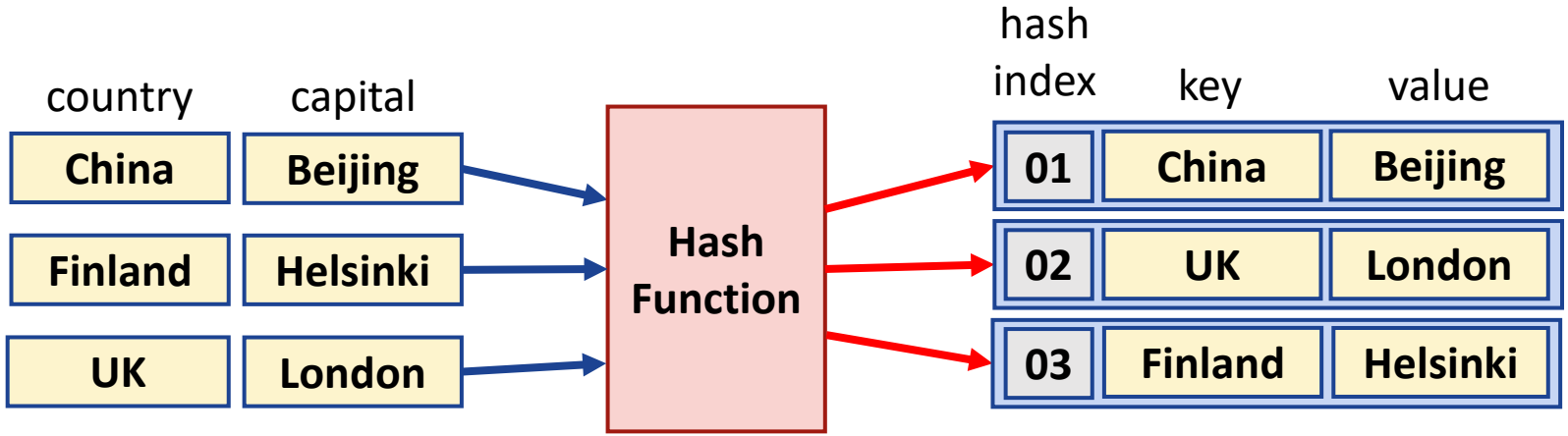
Maps

- **`Map`** elements consists of pairs of ***keys*** **(K)** and ***values*** **(V)**
- When iterating over **`HashMap`**, the order of elements is not guaranteed
- **`LinkedHashMap`** preserves the input order



HashMap

- Cannot contain duplicate keys (but duplicate values are allowed)
- Access elements by using method **`map.get(`**`key`**`)`**, add elements by **`map.put(`**`key,value`**`)`**, test if key exists with **`containsKey()`**



HashMap example

```
1  import java.util.*;
2  public class HashMapExample {
3    public static void main(String[] args) {
4      HashMap<String, String> capitalMap = new HashMap<>();
5      Scanner input = new Scanner(System.in);
6      capitalMap.put("China","Beijing");     capitalMap.put("UK","London");
7      capitalMap.put("Finland","Helsinki"); capitalMap.put("USA","Washington DC");
8      capitalMap.put("India","New Delhi");  capitalMap.put("Germany","Berlin");
9      System.out.print("Give name of a country: ");
10     String country = input.nextLine();
11     String capital = capitalMap.get(country);
12     if(capital != null) {
13       System.out.printf("Capital of %s is %s!\n", country, capital);
14     } else {
15       System.out.printf("Sorry, I don't know the capital of %s.\n",country);
16     }
17   }
18 }
```

```
$ java HashMapExample
Give name of a country: China
Capital of China is Beijing!
$
```

LinkedHashMap example

```
1  import java.util.*;
2  public class LinkedHashMapExample {
3    public static void main(String[] args) {
4      HashMap<Integer, String> olympics = new HashMap<>();
5      olympics.put(2008,"Beijing");          olympics.put(2012,"London");
6      olympics.put(2016,"Rio de Janeiro");   olympics.put(2020,"Tokyo");
7      System.out.println("Recent summer olympics (HashMap): ");
8      System.out.println(olympics);
9      olympics = new LinkedHashMap<>();
10     olympics.put(2008,"Beijing");          olympics.put(2012,"London");
11     olympics.put(2016,"Rio de Janeiro");   olympics.put(2020,"Tokyo");
12     System.out.println("Recent summer olympics (LinkedHashMap): ");
13     System.out.println(olympics);
14   }
15 }
```

Note the different order for `HashMap` and `LinkedHashMap` objects!

```
$ java LinkedHashMapExample
Recent summer olympics (HashMap):
{2016=Rio de Janeiro, 2020=Tokyo, 2008=Beijing, 2012=London}
Recent summer olympics (LinkedHashMap):
{2008=Beijing, 2012=London, 2016=Rio de Janeiro, 2020=Tokyo}
$
```

Word count example

```
1  import java.util.*;
2  public class WordCountExample {
3    public static void main(String[] args) {
4      Map<String,Integer> map = new HashMap<>();
5      Scanner input = new Scanner(System.in);
6      String text = input.nextLine();
7      String[] words = text.split(" ");
8      for(String token : words) {
9        String word = token.toLowerCase();
10       if(map.containsKey(word)) {
11         map.put(word, map.get(word) + 1);
12       } else { map.put(word,1); }
13     }
14     Set<String> keys = map.keySet();
15     TreeSet<String> sortedKeys = new TreeSet<>(keys);
16     for(String key : sortedKeys) {
17       System.out.printf("%-10s%-10d%n", key, map.get(key));
18     }
19   }
20 }
```

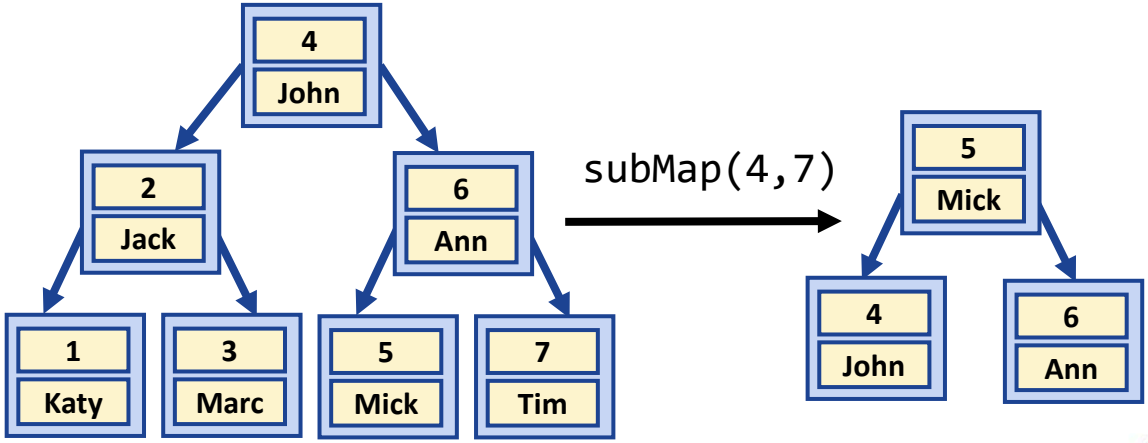
```
$ java WordCountExample
some words are more common than
some other words
are       1
common    1
more      1
other     1
some      2
than      1
words     2
$
```

This is just to order the words alphabetically

TreeMap

- **`TreeMap`** has similar internal structure as `TreeSet`, allowing fast operation of **`containsKey()`**, **`get()`**, **`put()`**, and **`remove()`**
  - Can be easily divided in submaps that have specific range of keys

Rank	Name
1.	Katy
2.	Jack
3.	Marc
4.	John
5.	Mick
6.	Ann
7.	Tim



TreeMap example

```
1  import java.util.*;
2  public class TreeMapExample {
3    public static void main(String[] args) {
4      String[] teams = {"Celtic", "Rangers", "Aberdeen", "Livingstone",
5       "Hearts", "St. Johnstone", "St. Mirren", "Hibernian", "Motherwell",
6       "Ross County", "Kilmarnock", "Dundee Utd"};
7      TreeMap<Integer, String> leagueTable = new TreeMap<>();
8      for(int i=0; i<teams.length; i++) {
9        leagueTable.put(i+1,teams[i]);
10     }
11     System.out.print("The first team: ");
12     System.out.println(leagueTable.firstEntry().getValue());
13     System.out.println("The top-3 teams: ");
14     System.out.println(leagueTable.headMap(4));
15     System.out.println("The teams in positions 6-8: ");
16     System.out.println(leagueTable.subMap(6,9));
17   }
18 }
```

TreeMap example: output

```
...
7       TreeMap<Integer, String> leagueTable = new TreeMap<>();
8       for(int i=0; i<teams.length; i++) {
9         leagueTable.put(i+1,teams[i]);
10      }
11      System.out.print("The first team: ");
12      System.out.println(leagueTable.firstEntry().getValue());
13      System.out.println("The top-3 teams: ");
14      System.out.println(leagueTable.headMap(4));
15      System.out.println("The teams in positions 6-8: ");
16      System.out.println(leagueTable.subMap(6,9));
17    }
18  }
```

```
$ java TreeMapExample
The first team: Celtic
The top-3 teams:
{1=Celtic, 2=Rangers, 3=Aberdeen}
The teams in positions 6-8:
{6=St. Johnstone, 7=St. Mirren, 8=Hibernian}
$
```

Custom comparators

- For sorting, it is important that the values of the elements can be compared with each other
  - For `String` and `Integer` objects this is trivial, but what about user defined classes?
- Custom **`compare(e1,e2)`** method can be used by implementing **`Comparator<E>`** interface for any element **`E`**
  - Return value is larger than 0, if `e1` is larger than `e2`; smaller than 0, if `e2` is larger than `e1`; and 0, if `e1` and `e2` are equal.
  - The implementation of **`Comparator<E>`** can be passed as a parameter to the constructor of sortable collections or maps.

Custom comparator example (1)

```
1   import java.util.* ;
2   class Card {
3     public enum Face {Two, Three, Four, Five, Six, Seven,
4                       Eight, Nine, Ten, Jack, Queen, King, Ace}
5     public enum Suit {Clubs, Diamonds, Spades, Hearts}
...   ...
16  }
17  class CardComparator implements Comparator<Card> {
18    @Override
19    public int compare(Card c1, Card c2) {
20      if(c1.getFace()==c2.getFace()) {
21        return c1.getSuit().ordinal()-c2.getSuit().ordinal();
22      } else {
23        return c1.getFace().ordinal()-c2.getFace().ordinal();
24      }
25    }
26  }
```

Similar to the deck of cards example yesterday

Order of playing cards depends primarily on the face, secondarily on the suit

Custom comparator example (2)

```
27  public class CustomComparatorExample {
28    public static void main(String[] args) {
29      CardComparator comparator = new CardComparator();
30      SortedSet<Card> cards = new TreeSet<>(comparator);
31      for(Card.Suit suit: Card.Suit.values()) {
32        for(Card.Face face: Card.Face.values()) {
33          cards.add(new Card(face, suit));
34        }
35      }
36      System.out.println("The first card: " + cards.first());
37      System.out.println("The last card: " + cards.last());
38      Card c1 = new Card(Card.Face.Three, Card.Suit.Clubs);
39      Card c2 = new Card(Card.Face.Three, Card.Suit.Hearts);
40      System.out.printf((comparator.compare(c1,c2) > 0 ?
41        "%s > %s\n" : "%s < %s\n"), c1.toString(),c2.toString());
42    }
43  }
```

TreeSet for sorted playing cards

Comparison of individual cards

```
$ java CustomCompratorExample
The first card: Two of Clubs
The last card: Ace of Hearts
Three of Clubs < Three of Hearts
$
```

UNIVERSITY OF ABERDEEN

Summary

- Queues are collections where elements are typically inserted and removed in first-in-first-out manner
  - Queues are useful for “waiting room” type of functionality
- Sets are collections of elements with no duplicate elements
- Maps are elements organised as key-value pairs with no duplicate keys (however, duplicate values are allowed)
  - Maps and sets allow efficient processing of sorted data
- To use sorting algorithms for objects of user-defined classes, custom comparator must be implemented for comparing the objects

Questions, comments?