



MAP



Map

Agenda

- 1 **Map Interface**
- 2 **HashMap Class**
- 3 **HashTable Class**
- 4 **TreeMap Class**
- 5 **Properties Class**

Map Interface

- Map is an interface that stores data in the form of key-value pair
- All the keys in the map will be unique
- We can retrieve the value stored in a map by providing the key value
- A Map cannot contain duplicate values
- Each key can map to at most one value
- For basic operations it uses the following methods
 - **put()** - for adding elements
 - **get()** - for retrieving an element
 - **remove()** - to remove an element
 - **size()** - to check the size of the Collection

The HashMap class

- HashMap uses the hashCode value of an object to determine how the object should be stored in the collection
- Hashcode is used again to help locate the object in the collection
- Gives you an unsorted and unordered Map
- Allows only one null key and multiple null values in a collection
- HashMap are not synchronized

```
HashMap<String,Double> hm = new HashMap<String,Double>();  
hm.put("John Doe", new Double(3434.34));  
hm.put("Tom Smith", new Double(123.22));
```

The HashMap class

Map is an object that stores *key/value* pairs.

- Given a key, you can find its value. Keys must be unique, but values may be duplicated.
- The HashMap class provides the primary implementation of the map interface.
- The HashMap class uses a hash table to implement Map interface.
- This allows the execution time of basic operations, such as `get()` and `put()` to be constant.

In the following example, it maps names to account balances.

The HashMap class

```
import java.util.*;

class HashMapDemo {
    public static void main(String args[]) {

        // Create a hash map
        HashMap<String,Double> hm = new HashMap<String,Double>();

        // Put elements to the map
        hm.put("John Doe", new Double(3434.34));
        hm.put("Tom Smith", new Double(123.22));
        hm.put("Jane Baker", new Double(1378.00));
        hm.put("Tod Hall", new Double(99.22));
        hm.put("Ralph Smith", new Double(-19.08));

        // Get a set of the entries
        Set set = hm.entrySet();
```

The HashMap class (Contd.).

// Get an iterator

```
Iterator i = set.iterator();
```

// Display elements

```
while(i.hasNext()) {
```

```
Map.Entry me = (Map.Entry)i.next();
```

```
System.out.println(me.getKey() + ": " + me.getValue()); }
```

// Deposit 1000 into John Doe's account

```
double balance = ((Double)hm.get("John Doe")).doubleValue();
```

```
hm.put("John Doe", new Double(balance + 1000));
```

```
System.out.println("John Doe's new balance: " +
```

```
hm.get("John Doe")); } }
```


The HashMap class (Contd.).

The output of the program is

Ralph Smith: -19.08

Tom Smith: 123.22

John Doe: 3434.34

Tod Hall: 99.22

Jane Baker: 1378.0

John Doe's new balance: 4434.34

The above program first populates the HashMap object. Then the contents of the map are displayed using a set-view, obtained by calling entrySet(). The keys & values are displayed by calling getKey() and getValue() methods of the Map.Entry interface.

***Note:** TreeMap instead of HashMap would have given a sorted output.*

The Hashtable Class

- Part of java.util package
- It implements Map interface and extends Dictionary Class
- It can contain only unique elements
- The key cannot have a null value
- It is a synchronized class

```
Hashtable<String,Double> balance = new Hashtable<String,Double>();  
balance.put("Arun", new Double(3434.34));  
balance.put("Radha", new Double(123.22));
```

Hashtable Example

```
import java.util.*;

class HashtableDemo {
    public static void main(String args[]) {
        // Create a hash map
        Hashtable<String,Double> balance = new
        Hashtable<String,Double>();
        Enumeration names;
        String str;
        double bal;
        balance.put("Arun", new Double(3434.34));
        balance.put("Radha", new Double(123.22));
        balance.put("Ram", new Double(99.22));
    }
}
```

Hashtable Example (Contd.).

```
// Show all balances in hash table.  
names = balance.keys();  
while(names.hasMoreElements()) {  
    str = (String) names.nextElement();  
    System.out.println(str + ": " +  
        balance.get(str));  
    }  
System.out.println(); } }
```

TreeMap

- Implements Map interface
- Provides efficient means of storing key/value pairs in sorted order
- Allows rapid retrieval
- Guarantees that its elements will be sorted in ascending key order
- The Key cannot be null but it can contain multiple null values

TreeMap Example

```
import java.util.*;

class TreeMapDemo{
    public static void main(String arg[]){
        TreeMap tm = new TreeMap();
        tm.put("Suresh",new Double(15357.75));
        tm.put("Meenu",new Float(18345.50));
        tm.put("Viren",new Integer(20000));
        tm.put("Avinash",new Double(19900.25));
        tm.put("Priya",new Integer(12000));
        tm.put("Zakir",new Float(16500.90));
        tm.put("Nirav",new Double(22000));
        tm.put("Jayesh",new Integer(15000));
        tm.put("Poorva","Zero");
    }
}
```

TreeMap Example (Contd.).

```
Set salary = tm.entrySet();
    Iterator it = salary.iterator();
    while(it.hasNext()){
        Map.Entry e = (Map.Entry) it.next();
        System.out.println(e.getKey()+" : "+" is "+e.getValue());
    } }
```

Properties

- Extends Hashtable.
- Used to maintain lists of key value pairs in which both the key and the value are Strings
- Useful method

//Used to print all the system properties

```
Properties p=System.getProperties();  
p.list(System.out);
```

//Used to get a system property user.name

```
System.out.println(p.getProperty("user.name"));
```


Example

```
import java.util.*;
class mysysproperties
{
public static void main(String arg[])
{

Properties p=System.getProperties();
p.list(System.out);
System.out.println(p.getProperty("user.name"));
}
}
```

Quiz

```
1. TreeSet map = new TreeSet();  
   map.add("one");  
   map.add("two");  
   map.add("three");  
   map.add("one");  
   map.add("four");  
   Iterator it = map.iterator();  
   while (it.hasNext() ) {  
       System.out.print( it.next() + " " );  
   }
```

- a. Compilation fails
- b. four three two one
- c. one two three four
- d. four one three two

Quiz (Contd.).

```
2. public static void before() {  
    Set set = new TreeSet();  
    set.add("2");  
    set.add(3);  
    set.add("1");  
    Iterator it = set.iterator();  
    while (it.hasNext())  
        System.out.print(it.next() + " ");  
}
```

The before() method will not compile

Which of the following statements are true?

- a. The before() method will print 1 2
- b. The before() method will print 1 2 3
- c. The before() method will not compile.
- d. The before() method will throw an exception at runtime.

Summary

- In this module, you were able to understand
 - How to work with
 - Map Interface
 - HashMap class
 - Hashtable class
 - TreeMap and
 - Properties class



Thank You