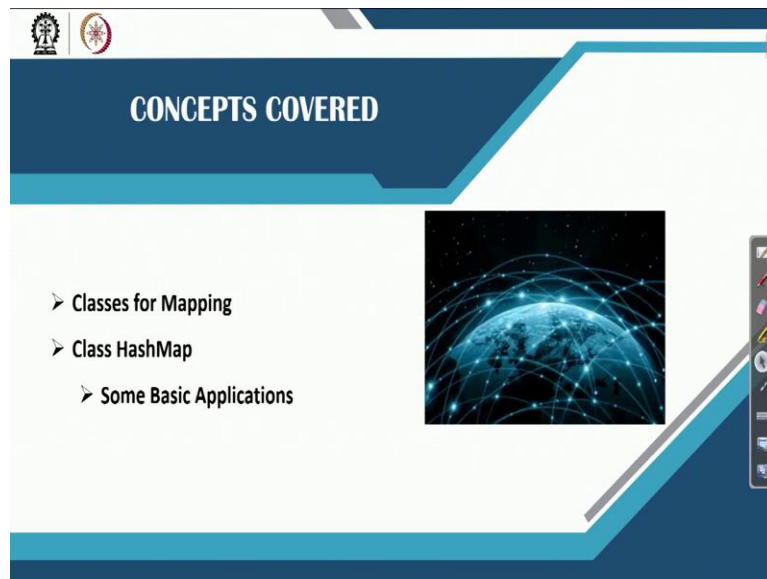


**Data Structures and Algorithm using Java**  
**Professor Debasis Samanta**  
**Department of Computer Science and Engineering,**  
**Indian Institute of Technology, Kharagpur**  
**Lecture 38**  
**Applications of Map (Part-1)**

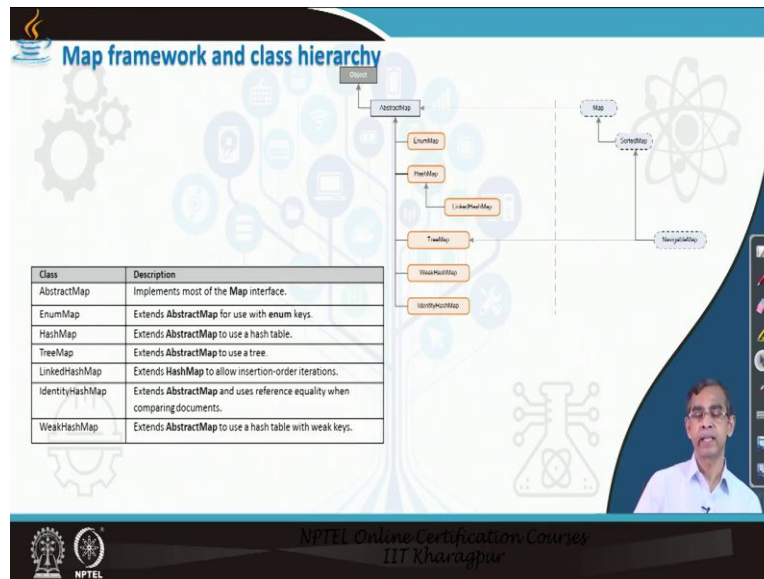
So we are discussing about a map framework. Map framework is used to manipulate tables. Table is a collection of objects and we have discussed about the Java supports and then map framework is there the map framework include many interfaces and classes. So today, in this lecture we will try to cover one important class which is called a hash map class.

(Refer Slide Time: 0:53)



So today we will discuss about the concept of mapping using hash map and then using this hash map how we can create a table of our own and then we can perform many operations on this table. So these are the basically topics that we are going to cover. Then other classes those are defined there in Java map framework will be discussed in the next lecture video.

(Refer Slide Time: 1:19)



So let us start about the map framework for the I mean the recapitulation and you have discussed about many classes are there namely in a map, hash map, then T map and then linked hash map those are the basically useful classes for many many managing tables and today we will discuss about hash map class.

Now so far this class is concerned we have to learn about what are the different constructors those are defined in this class so that we can use them to create the objects here object means the container object or collection object that means tables and then finally we will learn about what are the different methods are there by which this table can be manipulated.

(Refer Slide Time: 2:10)

The slide is titled "Constructors of HashMap class". It features a table with two columns: "Constructor" and "Description".

Constructor	Description
<code>HashMap()</code>	This constructor creates a default hash map.
<code>HashMap(Map&lt;? extends K, ? extends V&gt; m)</code>	The form is to initialize the hash map using the elements of m.
<code>HashMap(int capacity)</code>	The third form initializes the capacity of the hash map to capacity.
<code>HashMap(int capacity, float fillRatio)</code>	The fourth form initializes both the capacity and fill ratio of the hash map by using its arguments. The meaning of capacity and fill ratio is the same as for HashSet, described earlier. The default capacity is 16. The default fill ratio is 0.75.

Below the table, there is a "Note:" section with a bullet point: "There are no class of its own defined in HashMap class."

The slide also includes a small video inset of a man in the bottom right corner and logos for NPTEL and IIT Kharagpur at the bottom.

So this is basically our learning objective. Now there are four constructors define in this hash map class I have listed this for constructor this table, and this is a default constructor. So it basically create and table with a default size it is a eleven and then there is another constructor which can be created given a map is already known.

So if you pass an input as a map then a new map will be created using these constructors. Now, this constructor is basically you can specify what will be the size of the table that you want to store sometimes it is required for your better performance that we can specify the size so that memory will be occupied for you at one go. So this is the capacity that you can mention.

Now, another constructor is there in addition to the capacity we can say the field ratio. Now this collection can grow dynamically like other collection object that we have studied while we are discussing about Java collection framework. So, the field the field ratio say that if already filled for the quota that is the capacity is defined then for automatically growing so how much extra that thing is there.

So it basically mention here that it basically field ratio, we will tell you that if it is already occupied by the capacity that you have mentioned then the default capacity is the basically is the sixteen. I told that 11 it is not that 11 to 16 and then default field ratio is 0.75 that means whenever 15th or 16th records are already filled into the table then it automatically increase the size by 75 percent of this total size of the table.

So if it is hundred at the moment, then extra seventy five will be added into the stable so 100 75 will be the new size or new capacity of the table at the next going. So these are the different constructors are there, now let us see how the table objects can be created using these constructors.

Now before another important thing that you should note that there is no any special method which is defined in hash map, all the methods which are defined they are in abstract map that abstract map is basically abstract class which basically is two is inherits all the methods those are they are in map interface and then hash map implements all the methods those are basically declared in the map interface is basically. So there is no extra method other than the methods those are there in the map interface basically.

(Refer Slide Time: 5:02)

**Example 38.1: Creating a Map**

```
/* This example creates a map container and add objects into it using the
put() method. */

import java.util.*;

class HashMapCreation {
    public static void main(String args[]) {
        // Create a hash map object as a container.
        HashMap<Double, String> hmap = new HashMap<Double, String>();

        // Put elements in the map container
        hmap.put(300.0, "OK");
        hmap.put(303.0, "See Other");
        hmap.put(404.0, "Not found");
        hmap.put(500.0, "Internal Server Error");

        System.out.println(hmap); // Printing the container
    }
}
```

Handwritten red annotations: A diagram of a table with four rows and one column, with the label  $h(k)$  written above it. A red circle is drawn around the `HashMap` class name in the code.

Now, let us see how we can create the hashmap objects and what is the syntax and then programming for this purpose. Now, let us consider this is a simple program this program is basically gives an idea about how a hashmap can be created. Hashmap actually is a table that you can say right and now hashmap just a table it should be defined with two generic class or objects right as the arguments.

So here, double is the one argument and string is there. So this basically is the defined to all key that means all key are of type double and here all objects are of type string.

So there is basically mapping between the key values to the object and then this basically we call the constructor the default constructors for this class that we have to create a default table initially whose size is 16. Now here we can add few elements rather objects into the table. Initially when we create this collection it is empty because we do not pass any initialization elements into this collection so it is initially empty. Now we can add one objects at a time and it is an example that I can show you so for adding objects into the table the method that you can consider is called the put method.

So put method should be called for this collection it is called the H map with two elements values one is called the what is the key value and what is the object. So here okay is objects and then key value is 200 and like this. So what we have done is there four objects are added into the container H map. H map a basically hashmap collection or rather a table. So what we can say that a table consists of four objects okay see other, not found, internal server error and these are the key values are there.

Now, internally this hash map is basically generate hashcode for each objects whose key values is given there and mapped to this object so it will basically store here is basically in the table where the records are stored here, here the object like sting objects and then H is basically given key then it will tell in which in which location the key object this object key is stored.

So this is basically the object whose key value is K. So this is this way it is basically the one-to-one what is called in mapping and this is the fastest mapping that is possible compared to the binary search tree or trivial mapping that we have discussed in the last class so these basically key values are used to generate a hash code and then it basically used to tell that in which memory location a particular object is stored there.

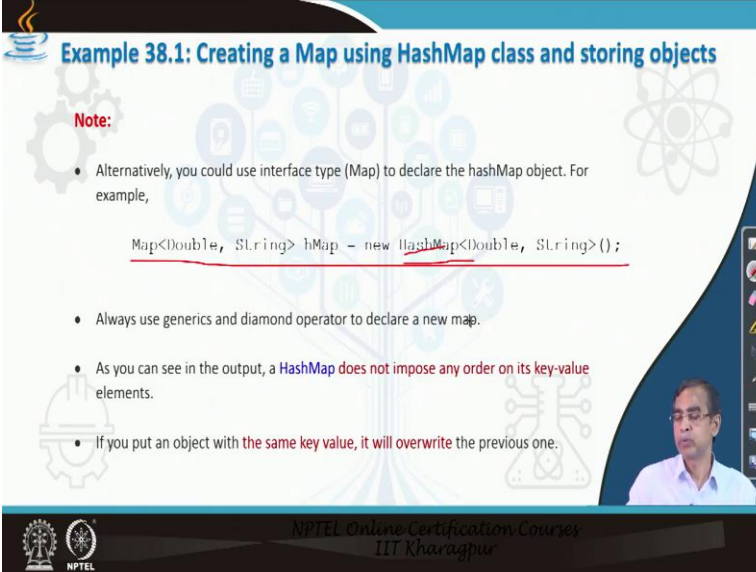
So this is the concept that it is followed here and we have generated a hash map collection is a table containing few objects it is there. Now, here again I can print using a simple print method by which all the objects can be printed you can run this program you can you can see how the table can be printed using this print method print simple println method as it is meant some here.

So this is a simple program and it is easy actually you do not have to bother about how hashing takes place, what is the mechanism it is followed there everything is taken care by the hashmap

class only you just go on adding and you just call the different methods to sharp your purpose your task actually.

Now, let us consider a few more example illustrating few more methods which are important there.

(Refer Slide Time: 8:56)



**Example 38.1: Creating a Map using HashMap class and storing objects**

**Note:**

- Alternatively, you could use interface type (Map) to declare the hashMap object. For example,  

```
Map<Double, String> hMap = new HashMap<Double, String>();
```
- Always use generics and diamond operator to declare a new map.
- As you can see in the output, a **HashMap does not impose any order on its key-value elements.**
- If you put an object with **the same key value, it will overwrite** the previous one.

NPTEL Online Certification Courses  
IIT Kharagpur

Now in this figure I just want to tell about any type of what is called a map table can be created by calling this map interface as a method this is also an alternative way you can do it also there earlier you can replace the constructor using this also it will work that means here it is not necessary to write hashmap you can like this one.

Actually this is the same procedure that you can call for other that is a other map container, for example T map you can write map this one here only T map can be replaced or it is a linked hash map or navigable map whatever it is there, basically they are same their methods are different maybe but the idea of programming is basically same and that is why a unique idea it is their.

Only difference is that hashmaps stores the different objects not necessary in sorted order where as sorted map store all the objects in the sorted order of their key pairs and link hash map is basically follow the link structure, tree map follow the T structure these are the different internal what is called the arrangements.

We do not really as a programmer point of view do not have to bother anything about how they are taking there but it is the different what is called the procedures the different concept the different mechanisms are used to mean to facilitate the different what is called a way something is very fast something is not so accurate whatever it is there. So those things are different but at the beginners level only knowing the different ways to create the tables is important that is what we are basically trying to do it in these video lectures.

(Refer Slide Time: 10:46)

**Example 38.2: Adding object with duplicate values**

```
/* This example creates a map container using the put() method and with duplicate
entries. */

import java.util.*;

class HashMapDuplicateDemo {
    public static void main(String args[]) {
        // Create a hash map object as a container.
        HashMap<Integer, String> hMap = new HashMap<Integer, String>();

        // Put elements to the map container with duplicates
        hMap.put(200, "Ok");
        hMap.put(300, "Ok");
        hMap.put(404, "Not found");
        hMap.put(300, "Internal Server Error");
        hMap.put(303, "Invalid entry");
        hMap.put(101, "See other");

        System.out.println(hMap); // Printing the container
    }
}
```

Now, let us see other examples so that we can learn little bit further few things here and this is another example which basically what will happen if we if you attempt to add a new objects but having the same key values as you know it should not allow the duplicate key values to be there in the table so in fact if you do that it will not report any error or any exception rather it will overwrite the initial entries by the new entries, so if you add any duplicate values the previous elements will be overwritten.

Now here is an example that you can run and you can check this program and this example basically hashmap we duplicate entries and again we create the container here the container has the two attributes. So key will be defined by a type integer and object of type string and here we add objects like this one, so this is the key value these objects 200, 300.

Now here you can see we add one more objects whose key value is same as this one then what will happen, then then this this basically will be overwritten by this one so this entry will not be



more no more rather you will obtain this one so if you have the print then you will see that this will not be printed rather all these things will be printed will be displayed on the screen.

So this basically duplicate key values if you add an object with duplicate key values or same key values then the last entry will exist all other previous entity will be I mean the previous entry with the same key value will be overwritten or replaced. So this is the idea about that how the table can be created.

(Refer Slide Time: 12:39)

**Example 38.3: Copying a Map into other**

```
/* This example creates a map container that copies elements from an existing map. */
import java.util.*;

class HashMapCopyDemo {
    public static void main(String args[]) {
        // Create a hash map object as a container.
        Map<String, Double> hMap1 = new HashMap<>();
        // put elements to the map container
        hMap1.put("John Doe", new Double(133.33));
        hMap1.put("Tom Smith", new Double(123.22));
        hMap1.put("Jane Baker", new Double(1978.60));
        hMap1.put("Tod Hall", new Double(99.22));
        hMap1.put("Ralph Smith", new Double(-19.60));
        System.out.println(hMap1); // printing the container
        // create a copy of a HashMap to hMap2
        Map<String, Double> hMap2 = new HashMap<>(hMap1);
        // Add data into hMap2
        hMap2.put("Robin Smith", new Double(623.22));
        hMap2.put("Dolor Swamy", new Double(10.60));
        System.out.println(hMap2); // Printing the container
    }
}
```

Now, let us see other things and how we can remove or you can access what are the different other operations can be done. Now, this is the one example who is basically gives an idea about how we can add already existing on map tables into the new tables. Now here this example basically making a copy of a table into another table like.

Now this example basically create a hash map let it be H map one this is the initial table and initially it is empty and then with these few statements we add few objects into it where you can note that key is of type string and then object is of type double and these are the different objects that we have added here and here you can note that we cannot write the throat here we have to create the object because this is the object it is required here.

Now so these are the different what is called entries are made into the initially empty table it is there. Now so this table is created we can print to see exactly how the table is created then you just took a create another table let it be h map two but we want to create this new table taking all



the entries that was already having in the h map one so what we can do is that when we use this constructor then we can pass the previous map as an argument to this one.

So this means that H map two will include all the elements which are they are in H map one and it will automatically decide its size depending on what is the size that is given as an argument and then the new you can add again few more entries into H map one or you can add few entries in h map two as you have declared here.

So this addition will go to the H map table mark not come to the h map two but here this addition will go to the h map two that means that is the map the container will grow it is there. So this example tell that how a table can be created having another table at the hand.

(Refer Slide Time: 14:52)

**Example 38.4: Retrieving objects from a Map container**

```
/* This example illustrates how an object stored in a map framework can be accessed
using get(). */

import java.util.*;

class HashMapAccessDemo {
    public static void main(String args[]) {
        // create a hash map object as a container.
        Map<String, Double> hm = new HashMap<>();
        // put elements to the map container.
        hm.put("John Doe", new Double(4334.33));
        hm.put("Tom Smith", new Double(123.22));
        hm.put("Jane Baker", new Double(1376.80));
        hm.put("Todd Hall", new Double(99.22));
        hm.put("Ralph Smith", new Double(-19.00));
        // deposit 1000 into John Doe's account.
        double balance = hm.get("John Doe");
        hm.put("John Doe", new Double(balance + 1000));

        System.out.println("John Doe's current balance: " + hm.get("John Doe"));
    }
}
```

NPTEL Online Certification Courses  
IIT Kharagpur

Now let us consider few more operations related to a table. We have understood about how that different objects can be inserted into the table now here again we will see exactly how a particular objects can be accessed from the table so for this purpose there is a method called get method which is basically more useful and here is an example that you can check it.

So this basically create a map name of the map is here H map and this is the objects are added into the table and here we just see you call the get method. So get and then the get method can be called either giving a key values or giving object. So in this case we call this giving a key value because in this case this is the key value and then it will return what is the object in this case this

is the object, so it will return the this basically suppose the balance for each employee or each person and these return there.

So we can again put method what basically we can modify it also or you can change it because here if we call put method for the same object (())(16:02) so what it happen is that we it will basically overridden by this with a new value. So balance plus thousand that means this will be incremented by thousand and the new value will be added into this one. So this basically put method can also be used as a replacement method also.

So this is an example you can say this mainly the get method by which you can access a particular record that is there in the table if does not have any table, then it will just return null. Now again if you can pass the key value also we have in this case we have passed the key value a alternately we can pass the object also and then you can call this get in that case it will return that a key value also a vice versa. So this method has the many overloading operations it is there.

(Refer Slide Time: 17:03)

The following methods are used for removing objects from a Map.

Methods	Description
<code>V remove(Object k)</code>	Removes the entry whose key equals <i>k</i> .
<code>default boolean remove(Object k, Object v)</code>	If the key/value pair specified by <i>k</i> and <i>v</i> is in the invoking map, it is removed and <b>true</b> is returned. Otherwise, <b>false</b> is returned.
<code>default boolean replace(K k, V oldValue, V newValue)</code>	If the key/value pair specified by <i>k</i> and <i>oldV</i> is in the invoking map, the value is replaced by <i>newV</i> and <b>true</b> is returned. Otherwise, <b>false</b> is returned.
<code>default V replace(K k, V v)</code>	If the key specified by <i>k</i> is in the invoking map, its value is set to <i>v</i> and the previous value is returned. Otherwise, <b>null</b> is returned.

NPTEL Online Certification Courses  
IIT Kharagpur

Now let us consider another example this example related to how we can delete a particular object from a entry and regarding this removal of objects from a map there are many methods are known and here we have listed few methods actually the remove method. Here this method can be called if we pass the object let the object be K. So given an object as an argument if we call this method then that particular object will be removed from the table. Now remove both object K and object V is basically removed in between that K and V will be removed.

So it is basically the mass removal we can say and it basically return true if the removal operation is successful in this case it will return the object if the if it removed successfully particular key value it is given there and replace also it is we have learnt about how using put also it can be done alternatively you can use the replace method also by which you can pass the key values and then old value and the new value that means if the old value is there matches then it will be replaced by new value provided that key match is there and it returned true or false depending on whether this replace is successful or not.

There is another also these are default that is also it can return by key by the V value that mean a particular key and B has to be passed, so that the place can replace these objects and then in return what is the object that it has replaced.

So they are different what is called the molyformic polymorphic methods which you can use in order to perform the removal operations I will give a very simple example to understand about it for other you can repeat the same with a different calling that invoking the different methods in your program as a practice.

(Refer Slide Time: 19:09)

The slide is titled "Removing objects from a Map container" and features a blue header with a coffee cup icon. A "Note:" section contains three bullet points explaining the `remove(Object key)`, `remove(Object key, Object value)`, and `replace(K key, V value)` methods. The slide also includes a small video inset of a man in the bottom right corner and a footer with the NPTEL logo and text "NPTEL Online Certification Courses IIT Kharagpur".

**Removing objects from a Map container**

**Note:**

- The `remove(Object key)` method removes the mapping for a key from the map if it is present (we care about only the key, and the value does not matter). This method returns the value to which the map previously associated the key, or null if the map doesn't contain mapping for the key.
- Similarly, the `remove(Object key, Object value)` method removes the mapping of a specified key and specified value, and returns true if the value was removed. This method is useful in case we really care about the key and value to be removed.
- The `replace(K key, V value)` method replaces the entry for the specified key only if it is currently mapping to some value. This method returns the previous value associated with the specified key.

NPTEL Online Certification Courses  
IIT Kharagpur

Now let us see an example how it can be used it okay so these are the basically usefulness of this error I have given a little discussion about different methods how they work actually and what is the difference between one with or two other because they are all replaced or removed method like.

(Refer Slide Time: 19:35)

**Example 38.5: Removing objects from a Map container**

```
/* There are two methods, namely remove() and replace() that you can use for remove an
entry in a map framework. The following example illustrates how an object stored in a
map framework can be removed. */

import java.util.*;

class hashMapDemo {
    public static void main(String args[]) {
        // Create a hash map object as a container.
        Map<String, Double> hm = new HashMap<>();

        hm.put("John Doe", new Double(134.54));
        hm.put("Tom Smith", new Double(133.77));
        hm.put("Jane Baker", new Double(1378.80));
        hm.put("Rod Hall", new Double(99.22));
        hm.put("Ralph Smith", new Double(-5.11));

        Double val = hm.remove("Jane Baker");
        if (val != null) {System.out.println("Removed value: " + val);}
        System.out.println(hm);
        hm.remove("Rod Hall", 99.22);
        System.out.println(hm);
        hm.replace("Ralph Smith", 545.67);
        System.out.println(hm);
    }
}
```

NPTEL Online Certification Courses  
IIT Kharagpur

Now let us see some example so that we can understand the use of removal method in a particular program and this basically the program illustrates how the remove and replace method can be called for a map container. Now here we first create a map and here the type or the key value of type string and then object is of type double and here are the basically usual method we have used basically to create the container or initialize the container we can say.

And here the remove method we have called and we pass the key here key is the Jane beaker. Now if it finds it basically returns the Double that is basically objects it is there so okay so in that case the double is the object it is there. So if it is found it is there otherwise it will return null. Now, here if b not equals to null then it basically gives a message that you can that you can just practice from this program you can check it here we call the remove method for other one instance this is the key value and also we passed it. So if both are matches, then only the this method will successful to remove it.

For example here, you see this is basically matching so it will there if you change this value by some other value and call this you will see that it is not possible. So in that case it will return null actually. And so these are the simple example here is another example here this basically it will give you a failure case of removal because it does not match this one it although matches this one it does not match. So whenever you call this method with two arguments both are to be masked then the operation will be successful.

So this example tell how the remove method is removal operation can be done from table the replace method also you can call and accordingly you can practice then other different versions of the replace and even method also can be practiced in the program you please try to do yourself.

(Refer Slide Time: 21:50)

**Accessing objects from a Map container**

Methods	Description
<code>V get (Object k)</code>	Returns the value associated with the key <code>k</code> . Returns <b>Null</b> if the key is not found.
<code>default V getOrDefault (Object k, V defVal)</code>	Returns the value associated with <code>k</code> if it is in the map. Otherwise, <code>defVal</code> is returned.
<code>int hashCode ()</code>	Returns the hash code for the invoking map.
<code>boolean isEmpty ()</code>	Returns <b>true</b> if the invoking map is empty. Otherwise, returns <b>false</b> .
<code>boolean containsValue (Object v)</code>	Returns <b>true</b> if the map contains <code>v</code> as a value. Otherwise, returns <b>false</b> .
<code>int size ()</code>	Returns the number of key/value pairs in the map.
<code>Collection&lt;V&gt; values ()</code>	Returns a collection containing the values in the map. This method provides a collection-view of the values in the map.

NPTEL Online Certification Courses  
IIT Kharagpur

Now we will give that how a collection can be viewed. It can be viewed as a set and that different methods are they are again to access a particular element as I said that get method is there by which you can retrieve a particular objects. There is basically key value if it is given then it will return the object is basically get or default. So whatever the value it is the end and then default value if it is there if it matches then it will return.

The hash code we have defined is empty is basically to check that whether collection is empty or not whether particular table contents a particular value which is passed as argument to take it it can be used this method. Size is basically to indicate that what is the present number of elements or total number of objects stored in the table that size method can be called for that purpose and then collection basically it gives you to see all the elements which are stored there in these objects.

Now let us quickly have some examples about the application of these methods that is called the view of a collections or map container.

(Refer Slide Time: 22:51)

**Example 38.6: Retrieving objects from a Map container**

```
/* The following example illustrates how an object stored in a map framework can be accessed.
For this, you should use the get(). */

import java.util.*;

class hashMapsExample {
    public static void main(String args[]) {
        // Create a hash map object as a container.
        Map<String, Double> hMap = new HashMap<>();

        // Put elements to the map container
        hMap.put("John Doe", new Double(100.00));
        hMap.put("Tom Parker", new Double(100.00));
        hMap.put("Jane Parker", new Double(100.00));
        hMap.put("Bird Hall", new Double(100.00));
        hMap.put("Ralph Smith", new Double(-10.00));

        // Deposit 1000 into John Doe's account.
        double balance = hMap.get("John Doe");
        hMap.put("John Doe", balance + 1000);

        System.out.println("John Doe's current balance: " + hMap.get("John Doe"));
    }
}
```

Now let us consider this is the one simple program. In this program we just see exactly how the different elements or objects can be accessed from a table. Now here we create a table the name of the table H map initially it is empty whose key value of type string and object is of type double.

Now here, we just initialize the table with few objects in it. It is just previous examples there. Now here the get method we pass the key value so it will search the entire table and then search will be very fast. It is not that one by one is just I hash map will take their own mechanism to hit to the table and then get the elements it will return the value that it basically has then John Doe it is there it will return this value so balance will be there.

If you print this balance, you can see this value is written successfully and then the put method it basically replace it basically replace this method and then after replacement you can find this one. So it is basically retrieving come modification that we have discussed let us consider few more example in this line to understand this concept better.



(Refer Slide Time: 24:11)

**Example 38.7: Accessing the status of a map**

```
/* There are two methods, namely size() and isEmpty() that can be used to
check the status of a map container. The following example illustrates how
an object stored in a map framework can be viewed. */

class HashMapDemo {
    public static void main(String args[]) {
        // Create a hash map object as a container.
        HashMap<Integer, String> hmap = new HashMap<Integer, String>();
        // Put elements in the map container
        hmap.put(200, "DOM");
        hmap.put(300, "Soc Other");
        hmap.put(400, "Not Found");
        hmap.put(500, "Internal Server Error");
        // Checking the container
        if (hmap.isEmpty()) {
            System.out.println("Error: The container is empty");
        } else {
            System.out.println(hmap); // printing the container
        }
        // printing the size of the container
        System.out.println("size : " + hmap.size());
    }
}
```

Here is another example, in this example what we can see the different other method like size, isEmpty and those other things it is there. So in this method this method to exercise the different other method you can write this program and continue to call the different methods that those are there in the hashmap class and then check it absolutely it will work.

You have to create the map container and then apply the method and you can see the result, getting the result is very simple just simply put system dot out dot printed println and then whole that container can be printed a particular record can be printed whatever the way you can print you can see the result.

Now here is another example. This example first create a container there is a key pair is a key keys of integer and then object is of type string and this is basically the different way that we can create the table actually. So the table is created and then we can call isEmpty that means the table is created if we call this method before this one we could that isEmpty is true, then it basically true then it will give that empty.

Otherwise it will print the entire collection that mean all the objects those are there will be printed by this by this statement and then size also how many elements are there so size can be printed here.

Then size is basically present size of the container is 4 you can see there and then again you can again call the remove method replace method get method all these things into to learn much



more about it and then again you can call size you can see that size is changing automatically and then it is initial sizes 16 you can go on adding four more method you do not have to bother about size memory you will be automatically allocated in a runtime manner. So it is so good for programmer actually so you just go on doing and many things those are required for the management point of view it is automatically done at the back end you do not have to bother about it.

So it is that is why it is so good efficient and then based mechanism to support the programming and to develop application. Now let us consider few more example in this direction and we will just consider about traversing a collection. We have a very raw method of traversing. Traversing means visiting all the elements which are there in the table.

(Refer Slide Time: 26:58)

**Map iteration**

As a **Map** is not a true collection, **there is no direct method for iterating over a map**. Instead, we can iterate over a map using its collection views. Any Map's implementation has to provide the following four Collection view methods:

Methods	Description
<code>keySet()</code>	Returns a Set view of the keys contained in the map. Hence we can iterate over the keys of the map.
<code>values()</code>	Returns a collection of values contained in the map. Thus we can iterate over values of the map.
<code>entrySet()</code>	Returns a Set view of the mappings contained in this map. Therefore, we can iterate over mappings in the map using the set view.
<code>forEach()</code> and <u>Lambda expression</u>	Additionally, you can use the Lambda expressions and the <code>forEach()</code> statement to view the map container.

NPTEL Online Certification Courses  
IIT Kharagpur

So there are many ways that traversal can be done and mainly the traversal that can be done by means of few methods which are defined in the map interface and ultimately implemented in the map hashmap class. So these are the key set is basically give you all is a set of all keys those are there in the table and then value these basically give you set of all objects those are stored in the table.

So a table stores everything the key values as well as objects. So now key set can be used to get the set of all key values and then empty set is basically written a set view of the mapping that mean it will give you the key and then objects both the things are there and then there is a

traversal called the forest method you can use we have we can use the iterative method also but I tell you method is not defined for the map framework but we can use it little bit customizing this one.

But best idea is that for each and lambda expression by which the each elements each objects in a table can be traversed and can be printed or can be displayed or can be steamed into some other collections or like this one. Now let us have some example, so that we can understand about how these method can be exercised and can be utilized in program.

(Refer Slide Time: 28:20)

**Example 38.8: Map iteration**

```
/* This program illustrates the above here ways of iterating over a map container */
import java.util.*;

class HashMapIterationDemo {
    public static void main(String args[] ) {
        // create a hash map object as a container.
        Map<String, String>mapCountryCodes = new HashMap<>();

        mapCountryCodes.put("11", "USA");
        mapCountryCodes.put("44", "United Kingdom");
        mapCountryCodes.put("33", "France");
        mapCountryCodes.put("81", "Japan");
        mapCountryCodes.put("91", "India");

        // Collection view of keySet()
        Set<String>keyCodes = mapCountryCodes.keySet();
        Iterator<String> iterator = keyCodes.iterator();

        while(iterator.hasNext()) {
            String code = iterator.next();
            String country = mapCountryCodes.get(code);
            System.out.println(code + " -> " + country);
        }
    }
} // Continued to next ...
```

And here is one simple program this program create a map the map is called map country codes is of hash map type. We initially create the table entering some elements which are there and then here you can see set course we basically create a set. Set object we will discuss in details on we will discuss another data structure said in in the next modules.

And so set is basically to store the collection of all the elements at the set, so it is basically set and the name of the collection is called set codes is basically called the key set method for this container map. So map country codes dot key sets. So it basically gives you all the key values those are there in this thing. So namely all these things will be stored in this collection set course if you can give the print system dot out dot println set codes you can see that all these value will be printed.

So whatever be the size of the table it is not an issue you can get at the key value and then you can process you can perform sorting and so many other manipulation that is required you can have it you can store them as an array or array list or whatever the other vector whatever it is there.

Now there is another way of traversing it is called the iterator. So we can just create an iterator in virtue of another method is called the set codes, set code is basically our code that we have created so create an iterator for the set code. Now you can see we can do that iterator not for the table because iterator cannot be used for the table but we can use for other collection here set collection. So we can use it and then we can see. So this basically print all the country codes that is their using this traversal selector. So it is traversing indirectly that container but using the set of a particular elements the same thing also can be applied to the other type of the object also.

So we can set codes other set objects and then we can get that get object and then we can use it. So get object method we have discussed earlier so this one.

(Refer Slide Time: 30:45)

**Example 38.8: Map iteration**

```
// Continued on...  
  
// Collection view using values()  
Collection<String> countries = mapCountryCodes.values();  
for (String country : countries) {  
    System.out.println(country);  
}  
  
// Collection view using entrySet()  
Set<Map.Entry<String, String>> entries =  
mapCountryCodes.entrySet();  
for (Map.Entry<String, String> entry : entries) {  
    String code = entry.getKey();  
    String country = entry.getValue();  
    System.out.println(code + " -> " + country);  
}  
  
// Collection view using Lambda expression  
mapCountryCodes.forEach((code, country) ->  
    System.out.println(code + " -> " + country));  
}
```

NPTEL Online Certification Courses  
IIT Kharagpur

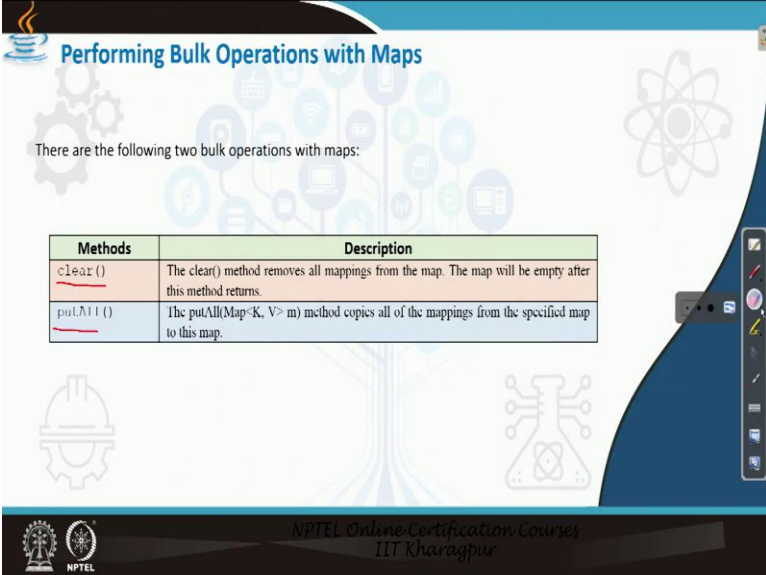
Now let us see an example so that we can explain this concept better and here is the another example that is basically we can have the set of key values and then set of objects also. Now so in order to have the set of object the method get value and set of key if you want to access it then method is get key that is the two methods are there.

Now this example basically is a continuation of the previous program we create so it again value this basically gives everything these basically gives set of key values as well as objects and then the for each can be used to traverse all these objects here. So it is indirectly we are visiting the container class then again we can you can contain them as a set also.

We create a set of entries like and then entry set that mean all key values and then object can be stored there and then we can use the for each method to print all the elements those are there. So there is again another for each and this is the basically lambda expression by which the entire collection can be printed.

So there are different ways the table can be traverse we have illustrated and I advise you to run this program with your own right understanding and then see the results you will be better understand how these codes are working for you.

(Refer Slide Time: 32:10)



The slide is titled "Performing Bulk Operations with Maps" and features a background with various icons like gears, a tree, and a molecule. It contains a table with two columns: "Methods" and "Description".

Methods	Description
<code>clear()</code>	The <code>clear()</code> method removes all mappings from the map. The map will be empty after this method returns.
<code>putAll()</code>	The <code>putAll(Map&lt;K, V&gt; m)</code> method copies all of the mappings from the specified map to this map.

At the bottom of the slide, there are logos for NPTEL and IIT Kharagpur, along with the text "NPTEL Online Certification Courses IIT Kharagpur".

There are a few more bulk operation and bulk operation means at a time if we can call a method get key get value in the sense they are bulk operation but few more bulk operation that needs to be discussed it is called the, in this example we have discussed one is called the clear you can understand what it does it basically clean the entire entries in the table and put all these method basically copies of all the mapping from a specified map to this map.

So put all basically we want to make a copy from one map to other map you can use the put all map also it is basically just okay merging you can say marking two tables into one table like.

(Refer Slide Time: 32:59)

The slide displays a Java code snippet for performing bulk operations on maps. The code is as follows:

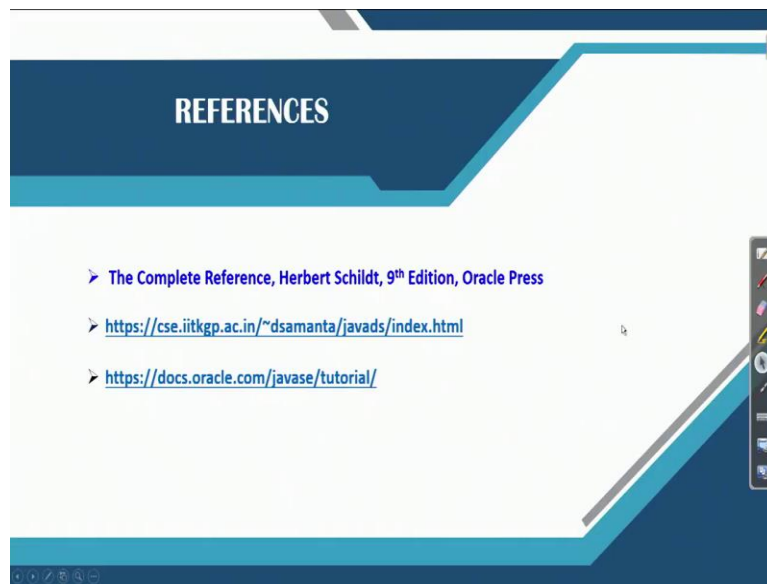
```
/* This program illustrates the above three ways of iterating over a map container */
import java.util.*;
class HashMapOperations {
    public static void main(String args[]) {
        // Create two map object containers.
        Map<Integer, String>countryCodesEU = new HashMap<>();
        countryCodesEU.put(44, "United Kingdom");
        countryCodesEU.put(33, "France");
        countryCodesEU.put(49, "Germany");
        Map<Integer, String>countryCodesWorld = new HashMap<>();
        countryCodesWorld.put(1, "United States");
        countryCodesWorld.put(86, "China");
        countryCodesWorld.put(82, "South Korea");
        System.out.println("Before: " + countryCodesWorld);
        // Merge two containers using putAll()
        countryCodesWorld.putAll(countryCodesEU);
        System.out.println("After: " + countryCodesWorld);
        // Clear one map container
        countryCodesEU.clear();
        System.out.println("Is Map empty? " + countryCodesEU.isEmpty());
    }
}
```

The slide also features the NPTEL logo and the text "NPTEL Online Certification Courses IIT Kharagpur" at the bottom.

Now, let us see some examples so that we can understand this concept better and here is an example who is basically explained the usefulness of put all method and then and then clear method that we have discussed. So this example here we can just create a table enter few objects into it and this basically you see merging. So here the merging operation is takes place how the using put all.

So there is a two one map it is here this map and another map is created so this map will be merged with this map. So new map will be added with this one. And then if we call clear then all entries those are there in this container will be removed permanently and then if you call this method you can see that it is now empty. So this basically is told some bulk operation that can be applied on the hashmap object.

(Refer Slide Time: 33:55)



So this is the topics that we have discussed about we have learn about hash map other the classes that we have to cover our namely T map and then linked hash map and sorted hash map we will discuss in the next video lectures and for your understanding and for reference you can follow the link that I have given here.

So these are the link that you can search and you can get many more materials here and this book also you can consult where good discussion about all those interfaces and classes you can find it. Okay thank you thank you very much.