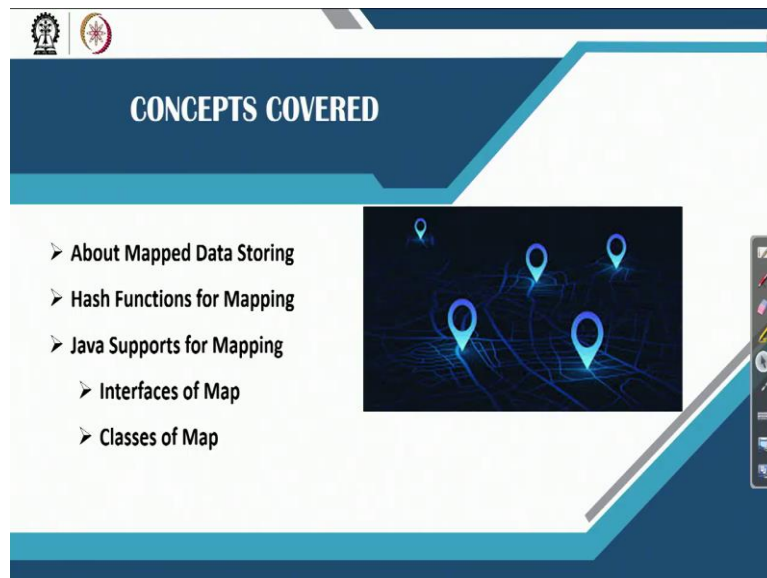


Data Structure and Algorithm Using Java
Professor Debasis Samanta
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur
Lecture 37
Map framework in Java

We have studied most extensively on framework it is called Java collection framework parallel to this java collection framework, there is one very important framework also known, it is called the Map framework. Map framework basically is to serve on data structure it is called a table. Now, table is a very common one form of data and particularly in most of the IT applications, the table is there so, a collection of tables it is basically from what is called a database.

(Refer Slide Time: 01:18)



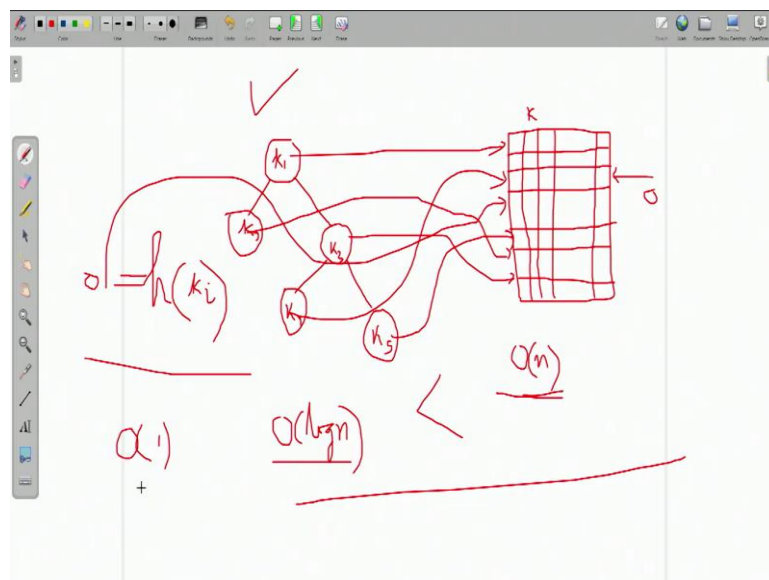
So, today our target is to cover this is very common, but extensively used data structure it is called a table and the map framework in java which basically supports all activities those are related to table structure. So, basically we will try to discuss about, how the java concept there is a map concept actually, that is used to store tables and manipulate tables and behind this manipulation, one important concepts is known, it is called a hash function so, we will try to learn about hashing or hash function and then, the map framework utility though, which is defined in java dot util package will be discussed in details. So, this is basically our plan.

(Refer Slide Time: 02:14)



Now, let us see what is the problem or other issues so, for storing data is concerned and how these issues can be addressed with the concept of what we are going to learn today, it is called a mapping.

(Refer Slide Time: 02:32)



So, I will try to give one idea about, this concept of the storing data now, consider there is a table so, a table is basically is a collection of records. So, we can say there is a number of rows so, each rows is basically is a record for example, there are say large number of students whose data needs to be stored in a database so, we can say a table can be planned for that for each record corresponding to a student's data so, this record is also in our concept of object

oriented programming, it is called objects so, a student object we can say. Now, as you know, each record is defined by a number of attributes so, these basically constitute is called the columns. So, there are maybe any number of columns are there so, these columns basically in the context of object it is basically called a field.

So, an instance of a student's class which basically constitute a record and they are each a field which can be considered as a unique there are maybe more than one fields also by which each record can be uniquely identified so, such a field it is called a primary field or we can say primary key.

So, let this is the field K is the primary field and these are that many objects we can say so, this is basically the third object so, it is an object O so, a table is nothing but a collection of objects and each object is identified by a field or it is called a key that is called a primary key now, so this is basically storing a table in a database, which is the common concept now, what is the problem, let us see.

Suppose, we want to modify a record for a given student so, what we have to do, we have to reach to that row where that student's record is available so, how we can reach so, we have to search the table and then, if records then, that record can be read and then, whatever the modification is required we can do it so, searching is a very common what is called operations, which is required in case of handling table.

Now, you see the searching can be done in many different ways. So, a trivial searching procedure is basically starting from the first record and compare the key of that record which basically our target record and go on until we find the record or we can reach at the end of the table but, this process you know, it is easy simple although but, it is not an efficient particularly when the large number of records are stored in the table so, what is the procedure?

We need one some mechanism by which this searching time can be minimised because, in a table there are a huge number of records accessing is require so, each time searching takes enough time then, definitely it will degrade the performance of the system so, one what is called the indexing mechanism is required.

Now, towards this direction I mean indexing, we can apply binary search tree which we have already studied. Binary search tree is basically an important data structure for the purpose,

what binary search tree does for us? So, for the indexing is concerned it is very simple you know, in case of binary search tree, there is a number of nodes number of nodes and all the nodes are arranged in a particular order is basically satisfying the binary search tree property.

Now, here each node basically store the key value off a record so, K1 K2 key value of record so, K3 whatever it is you can see K4 and so on so, the number of nodes is basically in the binary search tree is same as the number of records in this table now, K1 is basically you know, if we want to search a particular record having a particular key that has been the primary key then, we can take the confidence of these binary search tree to go fast.

Now, for this for each key it also store one additional information the information regarding actually which is the memory location where that particular record which key value is store in the node so, this means that each node store not only the key values but, in addition to a pointer, pointer to which record it is stored there so, it basically store a pointer so, suppose this is a pointer, which is basically pointing up a record which key value is key five and it is like this and so on so, this pointer is there and like this.

So, this binary search tree not only stored the key values but, also stored the pointers. So, suppose you want to access a particular record whose key value is K3 we can just take where is the K3 is there go, come here and then go to the record so, we can go it now, what are the advantages? Advantage is that the earlier searching which was simple and trivial it took order of n time that mean, n number of comparison that is required in order to find the particular record on the other hand, if we take the binary search tree then, it will take order of $\log n$ and since this order of $\log n$ is less than order of n so, we can say that, this procedure is far far faster than, this procedure that is that trivial linear searching.

Now, here basically you can see we have followed certain concept of indexing or there is a mapping there is an so, this basically the mapping that we have discussed but, there is another mapping this mapping is far far better, than both order of $\log n$ or order of n what is the mapping?

The mapping is that there is a function let this function be h these functions is called hash function and if the target key say key that is, we want to find a record having the K so, let K either i th is record we want to find now, this h function is planned in such a way that if we give the K_i value then, this hK_i will give the address where, this record having the key K_i is

(Refer Slide Time: 11:18)

Concept of mapping

- A Map is a data structure that's designed for **fast lookups**.
- In map, data is stored in **key-object pairs** with every **key being unique**.
 - That is, no two objects should have the same key.
 - Each key maps to an object and hence the name.
 - These pairs are called map entries.

NPTEL Online Certification Courses
IIT Kharagpur

So, let us start about the concept of what is called a hashing concept it is there now, as we see here so, in case of hashing technique, there is a two things that do be in are involved on is that for each object and there is an association with a key value so, key object pairs are there and then key obviously should be unique that means there should not be two objects for which the key values are same, that means two different object is are there then, they should have the different key values are there so, it is basically very good for fast searching and each key is basically maps an object and this mapping can be often by a mapping formula, it is called a hashing concept is there.

(Refer Slide Time: 12:10)

Concept of mapping

Suppose, you want to maintain a large set of objects and you need to access them as fast as possible.

Mapping mechanism in Java allows you do this associating a key to each object. With this, the user's view of storing objects looks like a table below.

Key	Object
k_1	o_1
k_2	o_2
k_3	o_2
k_4	o_5
k_5	o_3
k_6	o_6

NPTEL Online Certification Courses
IIT Kharagpur

So, that is what the concept is that for every object O_i there is a key K_y and they are for each object is mapped by means of K_y , K_i now, that is why this mapping is possible, this mapping is possible by means by a h function it is called a hash function.

(Refer Slide Time: 12:28)

The slide is titled "Map as a mathematical abstraction of function". It contains three bullet points:

- For an object (o), there is a key (k); mathematically $o=h(k)$, where h denotes a function.
- Intuitively, this function h is called a hash function.
- Thus, map data structure models the function abstraction in mathematics. Figure below illustrates a map:

The figure shows two sets, Y and X , represented as irregular shapes. The left set Y is labeled "Key" and contains four black dots. The right set X is labeled "Object" and contains four red dots. Lines connect each dot in Y to a corresponding dot in X . Handwritten in red above the sets are the expressions $f(x)$ and $y = f(x)$. The slide also features a small video inset of a man in the bottom right corner and logos for NPTEL and IIT Kharagpur at the bottom.

Now, this mapping if you see it is very similar to the function for example, y equals to a fx function so, what is that the theme, theme means that our x is basically dependent x is basically x is a independent variable and y is the dependent variable so, for a given value of x , there is a value of y it is not true that for a given value of x there are two values of y it is not the property of function but, for two different values of x they are maybe same values of y it is possible in case of mapping anyway so, this is a usual concept of mapping that is known in mathematics.

So, here mapping means if this is the set x and this is the set y is basically and this mapping is the function effects means how the x can be related to y and $(())(13:31)$ it is there so, this mapping is basically the concept behind a hashing technique that we are going to use for our database or table manipulation.

(Refer Slide Time: 13:44)

Example of maps

Maps are perfect to use for key-value association mapping such as dictionaries. The maps are used to perform lookups by keys or when someone wants to retrieve and update elements by keys.

Examples:

1. Filenames and their contents stored in a secondary storage place.
2. All error codes and their descriptions for a system.
3. Zip codes and delivery destinations in a courier service.
4. A map of managers and employees. Each manager (key) is associated with a list of employees (value) he manages.
5. A map of classes and students. Each class (key) is associated with a list of students (value).

NPTEL Online Certification Courses
IIT Kharagpur

Now, so we have learned about the mapping, mapping has extensively used many application in our usual concept of file structure that is used in computer memory, there is every file name has association to a file object actually which is stored in memory so, there is a mapping from the file name to file which is stored in hard disk or secondary storage and like this the mapping is every error for every error code there is a corresponding error mechanism routine and a for every country or every location there is a PIN code so, there is a mapping from location purchase PIN code and like, like so, there are many examples of mapping which we basically are familiar to.

(Refer Slide Time: 14:30)

Importance of maps

Following are some important characteristics about Map.

1. A map is a mechanism for the fast retrieval of an object.
2. In a map, an entry is characterized with a pair of elements: key and object.
3. Given a key you can find an object. In other words, retrieving an object requires a key to be supplied.
4. The keys must be unique, but the objects may be duplicated.
5. A map cannot contain duplicate keys.
6. Mathematically, map implements an one-to-many and onto association.

NPTEL Online Certification Courses
IIT Kharagpur

Now, this concept of mapping is very useful to help us in indexing a table so, is basically mapping is useful for fast retrieval of any object or object where basically represents a record and entry is characterised with pair of elements in our mapping mechanism as you have already mentioned about the record itself and the key value and mapping is basically by virtue of this mapping mechanism or hashing technique is there and as I told you the key must be unique as you know, in case of binary search tree and that is how we have assumed that there is a field for every record for which each record have that distinguishable or discriminated by that key value or field.

So, that is why it is unique and so, duplicate key it is possible but, we should not include that duplicate key is there so, a map should not contain any duplicate keys in that sense so, mathematically the mapping which we are going to use is basically on to on and on to mapping it is called the association.

(Refer Slide Time: 13:44)



The slide features a central title "Hash Functions for Mapping" in blue text. To the left is an icon of a coffee cup with steam. The background is a light blue tree-like structure composed of various technology-related icons such as a gear, a smartphone, a laptop, a Wi-Fi symbol, and a document. In the bottom right corner, there is a small video inset of a man in a light blue shirt. The bottom of the slide has a dark blue footer with the NPTEL logo on the left and the text "NPTEL Online Certification Courses IIT Kharagpur" on the right.

Hash function for mapping

- For the key-object association, a hash function is required.
- This hash function, in fact, generates a code called hash code.
 - That is, with $o=h(k)$, h being the hash function, given an object o , h returns a unique value k .
- In general, this system dependent and for the internal use of the system.

$h(k) = (k, O)$

NPTEL Online Certification Courses
IIT Kharagpur

Now, let us come to the concept of hashing as a mapping function now, so it is basically as I told you, in case of hash function so, there is any for any object there are two pairs key k and O so, these basically if hash function is there so, hk then, it tell that which is the object it is basically so, this concept is there so, there is a hashing technique is for this purpose only now, let us the come to some example so, that we can understand about this hashing function better.

(Refer Slide Time: 16:25)

Hash function for mapping: Examples

A simple hash function is defined as follows.

$$h(k) = k \bmod 19$$

Here, mod is modulo remainder operation. Some examples are:

$$h(123) = 5$$

$$h(99) = 4$$

$$h(2014) = 0$$

$h(24) = 5$

Note that in this example, the hash function generates has codes within the rang $[0, 1, \dots, 18]$ for any integer numbers. Thus, a key k is used to generate a hash code, which determines where in memory the $\langle k, o \rangle$ pair is stored. This is why a map is also sometime referred as a dictionary because of the way it works.

NPTEL Online Certification Courses
IIT Kharagpur

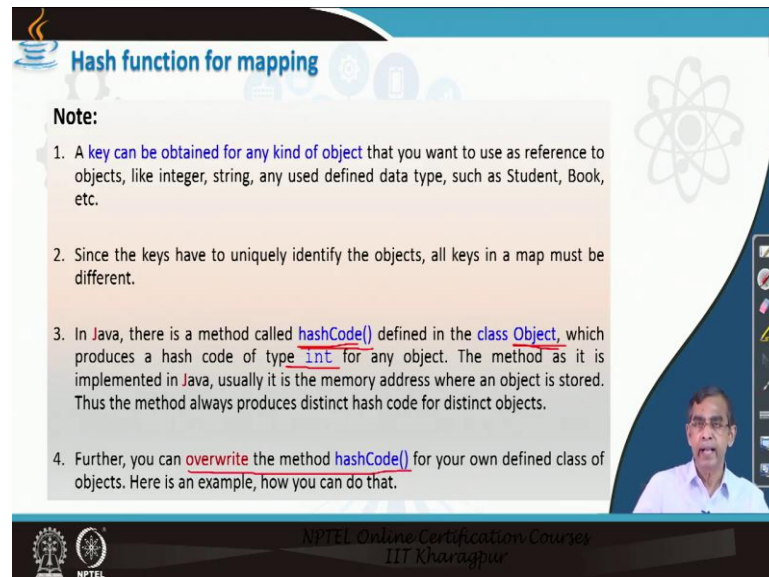
Now, I gave an example of a simple hashing function here is the hashing function let this hash function is defined like this so, you know modular remainder operation for any integer k assuming that k is an integer so, it keeps you, here is some example as we can see so, $h(123)$ if we apply this model function it will be 5 so, hk is equals to 5 in this case, for another

example, h is 99 this gives you 4 and this one so, these are basically we can say address or memory where, the object whose key value k is stored. So, it is an object whose key value 123 it is another object whose key value is 99 it is another object is evaluates 2014.

Now, here judiciously you have to decide this key value so, that any collision if it is there it is avoidable or it can be avoided so, collision if it is there, it should not be (())17:26) it is not desirable for example, I can say suppose, h and 24 so, this is also equals to 5 that means, if there is another object whose key value is 24 they need basically gives the same mapping or memory location that means, the 2 objects whose key values 123 and 24 are located in the same memory location which is not possible, which is not desirable.

So, hashing mapping or hashing functions should be chosen in such a way that for every objects they are the memory location it will return is basically unique and that is the problem which basically needs to be handled very carefully so, this is also a very simple example but, these kind of examples may not work always so, you have to take care about this kind of problem and that is the only issue that we have to think about how this can be solved there.

(Refer Slide Time: 18:32)



Hash function for mapping

Note:

1. A key can be obtained for any kind of object that you want to use as reference to objects, like integer, string, any used defined data type, such as Student, Book, etc.
2. Since the keys have to uniquely identify the objects, all keys in a map must be different.
3. In Java, there is a method called `hashCode()` defined in the class `Object`, which produces a hash code of type `int` for any object. The method as it is implemented in Java, usually it is the memory address where an object is stored. Thus the method always produces distinct hash code for distinct objects.
4. Further, you can **overwrite** the method `hashCode()` for your own defined class of objects. Here is an example, how you can do that.

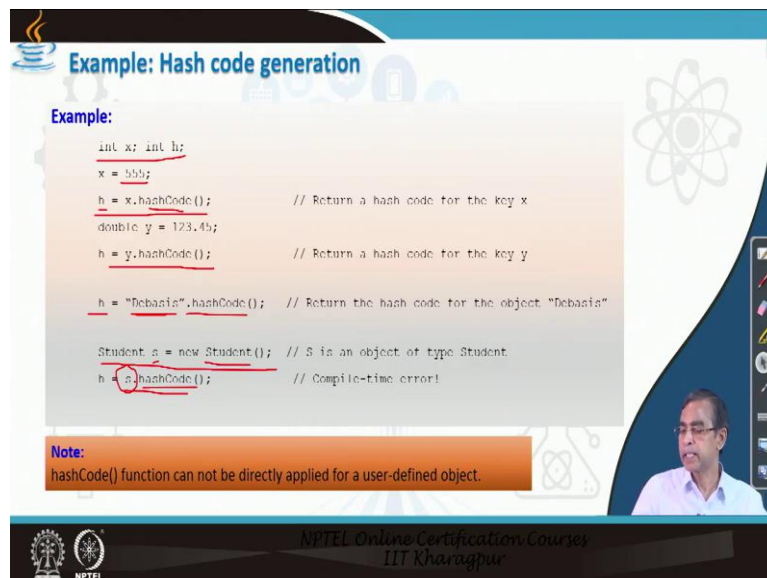
NPTEL Online Certification Courses
IIT Kharagpur

Now, in order to resolve these issues, there is a concept this concept is called hash code hash code is one method which is defined in the object class, the class object is a super class of all objects, which is defined in java dot lang package this means, that any objects that you define basically is a child of the parent class of objects so, it all objects that you define by any class declaration like say student, person, employee whatever it is, they are basically the child class of object so, hash code is defined in the object which is the default method and so, as any

object is a child class of object so, by virtue of inheritance this hash code method is also available to this one.

Now so, this hash code basically gives say an integer value maybe long value it basically essentially a memory address where a particular object is stored so, this hash code therefore is a very clever solution that is defined by java developer which basically if you give a key value for any object or set of key values to this hash function then, it will calculate a unique address for that object and however, you can also override the hash code method in your class declaration we will see exactly how such a overriding of method is possible or overriding the hash code that means, you can define your own hash code also and usually that is more desirable for the programming point of view.

(Refer Slide Time: 20:16)



The slide is titled "Example: Hash code generation" and features a Java code editor with several lines of code. The code includes variable declarations for integers and doubles, and calls to the `hashCode()` method. A comment indicates that the last line, `h = s.hashCode();`, results in a "Compile-time error!". A note at the bottom states: "Note: hashCode() function can not be directly applied for a user-defined object." The slide also includes the NPTEL logo and the text "NPTEL Online Certification Courses IIT Kharagpur".

```
int x; int h;  
x = 555;  
h = x.hashCode(); // Return a hash code for the key x  
double y = 123.45;  
h = y.hashCode(); // Return a hash code for the key y  
  
h = "Debasis".hashCode(); // Return the hash code for the object "Debasis"  
  
Student s = new Student(); // S is an object of type Student  
h = s.hashCode(); // Compile-time error!
```

Note:
hashCode() function can not be directly applied for a user-defined object.

Now, let us consider an example for this so, here x and h are the two integer variables declared suppose x is 555 then, here h and then hash code if we call for this integer it basically return what is a hash value so, it basically it will return for some mechanism it is there, which mechanism? That is basically, it is internal to the java development we should not think about how they convert an integer value to hash value so, this hash value it will return you can write the programme and you can call this method for any variable or any objects and you can find it.

Now, here double y and if you call the hash code for this y, it will basically give a unique value corresponding to the hash code of y so, it is basically just a hx by the hash mapping we can say now, for a given string also this is a hash code so, usually we use a password or login

id so, computer convert those password or login id into hash code and that hash code is basically a one way function given the hash code it is very difficult to come back to the original for example so, this is the string for which the hash code that h will return and from this given we will not be able to come to the what is our original input string for which the hash code is h.

Now, as another example, let us consider this so, h is an object of class student now, (())(21:55) hash code we can obtain in what is that hash code of this object h but, this is possible if we define our own hash code for this object otherwise called this code is basically sometimes give you a compile time error and it is avoidable because, it will try to give that default hash code which is basically compile time does not accept it so, here basically you need to define your own hash code for the newly defined class h for example here.

(Refer Slide Time: 22:27)

Example: Overriding hashCode() method for a user-defined class

```
/* This program illustrates how the hashCode() can be rewritten to return a hash code for a user-defined object. */  
  
import java.util.*;  
  
class Person {  
    String firstName;  
    String lastName;  
    int age;  
  
    Person (String f, String l, int a) {  
        firstName = f;  
        lastName = l;  
        age = a;  
    }  
  
    public int hashCode() {  
        return (firstName.hashCode() * 7 + lastName.hashCode() + 11 * age.hashCode());  
    }  
  
    public static void main (String args[]) {  
        Person p = new Person("Vijay", "Raj", 21);  
        int k = p.hashCode();  
        System.out.println("hash code " + k);  
    }  
}
```

NPTEL Online Certification Courses
IIT Kharagpur

Now, let us see how such a hash code can be overridden and that is a programme you can consider this programme for an example, this programme is basically declare a class the name of the class is passion it has three fields, the one is first name last name and age and here is a constructor of this class to initialise objects and here basically we are overriding hash code how we are overriding hash code and this is the one method that we have decided, how we have decided it is up to the programmers ability.

So, here we decide that what is the hash code up string first name multiplied by 5 plus that hash code of string last name multiplied by 7 plus the hash code of integer h multiplied by 11

and then it will return a very complex on numbers which is basically considered the hash code of a particular object which basically we creates.

So, for example, here an object is created say p and having these are the different values for the objects and then it will return the hash code k for this object and you can print this as code if you print this hash code usually you can see it is in a hexadecimal string that it can return so, this way we can define the hash code of our own.

Now, here is basically is the intelligent aptitude that needs to be applied how we can define hash code for all objects, which should be unique for every objects that you have to deal with now so, there are certain theory behind it, those theories are you want to skip if you are interested, you can search many literature regarding has code generation you can find many ideas from there so, there is not a single idea rather many ideas to generate the hash code it is there.

(Refer Slide Time 24:30)



So, we have learned about the concept of hashing and hash code generation and how this hash code basically gives a unique representation for a given key value of an object where key value can be only a single attributes or it can be multiple attributes as we have planned in the last examples by our first name last name and age are considered as an attributes to generate the objects. So, whatever it is not an issue so, key value is not necessary that is a single value, it can be a composite value what I want to mention categorically.

Now, let us come to the concept of so, these are the things that we have discussed about from the theoretical point of view in the conceptual level but, many things are made easy what the java developer has done for their programmers so, that we can have it so, that is the map framework for which we can have all those things automatically we do not have to bother about, how this hash code, how it can be generated, how it can be used, and whatever it is there now, let us see what is the map support which is there in the java framework

(Refer Slide Time: 25:28)

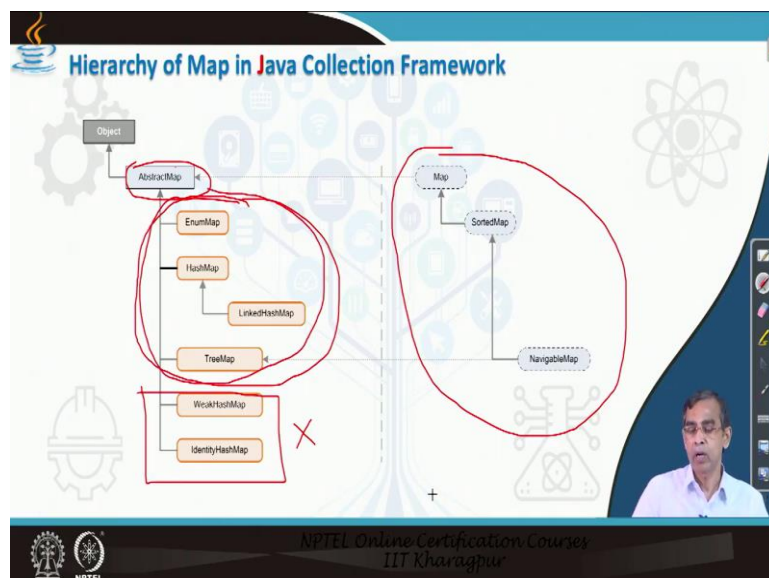
Map facility in Java Collection Framework

- Java introduces the concept of **Map**, which is another member in the Java Collection Framework.
- In Java, a **Map** is an object that maps keys to values, or is a collection of key-value pairs.
- It models the function abstraction in mathematics.

NPTEL Online Certification Courses
IIT Kharagpur

Now, java consists of two framework the collection framework and map framework, map framework is basically to deal with these tables are mapping that is there.

(Refer Slide Time: 25:41)



Now, the map framework if which is the class hierarchy that is there, it has this class hierarchy there are a number of interfaces defined in this framework, the interfaces are mentioned here, namely map, sorted map and the navigable map, those are the interfaces which basically have that design, any class you can design, if you want to implement a particular interface so, interface has the declaration of all methods or fields or any constant in them, there is no method defined there.

Now, so there are certain methods which basically you can use in your programme to create the object in order to manipulate your tables by virtue by means of your mapping mechanism the now important the classes which basically we consider are there now, there are two classes, these two classes are basically for the java developers use only so, we will not consider all those things in there.

Now, so, there are four classes one is called inner map, hash map, linked hash map and Tmap this basically these classes allow you to create a collection this collection is basically table and automatically the indexing and everything will be taken care of by this now, using these classes you can create objects and for this object you can call many method how to add new object into this table, how to remove, how to access, how to search so, many operation those are usually require for the fastest access from the table.

Now, here again in Enum Map easy basically is extends the abstract map, abstract map is abstract class actually, that hash map is also extend abstract map linked hash map is basically extend the hash map and Tmap again extend the abstract map so, this means the again, you see abstract map basically implements map and then sorted map is basically extends the map, navigable map again extend the sort map so, all methods which are defined in map is basically a by means of inheritance also included there in addition to the methods in map they have their own method defined in sorted map and everything, all the methods which are defining these interface are implemented in all these classes.

So, these all these classes are implements all the methods which have did here so, in our subsequent few slides, we will try to understand about what are the interface competitions, what are the methods are there so, that we can use them in our programme regarding programming using map will discuss in the next video lectures today, we will just have the familiarity about the composition of this map interfaces and map classes it is too much also

but, I will try to give a brief summary about all those things so, that we can have a quick look about this.

(Refer Slide Time: 28:30)

The slide features a central title "Map Interfaces" in blue. To the left is an icon of a coffee cup with steam. To the right is a stylized tree diagram with various icons on its branches. The background is light blue with faint icons of gears, a hard hat, and a molecular structure. A small video inset of the speaker is in the bottom right corner. The footer contains the NPTEL logo and the text "NPTEL Online Certification Courses IIT Kharagpur".

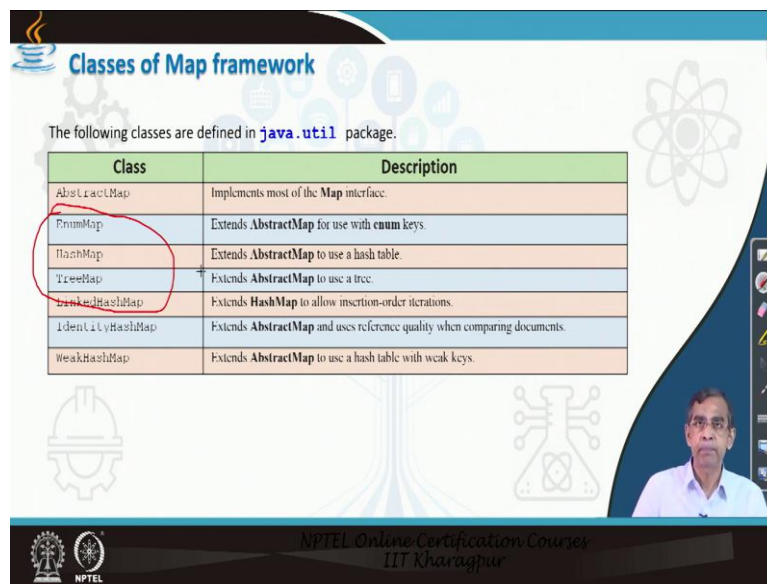
The slide is titled "Interfaces of Map framework" and includes a small Java logo. Below the title, it states: "The following are the interfaces declared in `java.util` package." A table with two columns, "Interface" and "Description", lists the following:

Interface	Description
Map	Maps unique keys to values. The interface is generic and it is defined as interface <code>Map<K, V></code> , where <code>K</code> specifies the type of keys, and <code>V</code> specifies the type of values.
Map.Entry	Describes an element (a key/value pair) in a map. This is an inner class of <code>Map</code> .
NavigableMap	Extends <code>SortedMap</code> to handle the retrieval of entries based on closest match searches.
SortedMap	Extends <code>Map</code> so that the keys are maintained in ascending order.

The slide also features the same decorative elements as the first slide, including a video inset of the speaker and the NPTEL footer.

Now, let us have the map interfaces we have told you that, there are three map interfaces one is map another is sorted map and another is a navigable map in addition to this, there is also on another inner class is define it is called a map dot entry which is within the map interface it is there so, all these map interfaces basically is a generic or what is called a type, the type is of k and v, where k is all generic type and v is of an index so, k can be any type v can be any type and v can be any type of object k can be any type of object it is the concept is there, so, is the pair.

(Refer Slide Time: 29:14)



The following classes are defined in `java.util` package.

Class	Description
<code>AbstractMap</code>	Implements most of the <code>Map</code> interface.
<code>EnumMap</code>	Extends <code>AbstractMap</code> for use with <code>enum</code> keys.
<code>HashMap</code>	Extends <code>AbstractMap</code> to use a hash table.
<code>TreeMap</code>	Extends <code>AbstractMap</code> to use a tree.
<code>LinkedHashMap</code>	Extends <code>HashMap</code> to allow insertion-order iterations.
<code>IdentityHashMap</code>	Extends <code>AbstractMap</code> and uses reference equality when comparing documents.
<code>WeakHashMap</code>	Extends <code>AbstractMap</code> to use a hash table with weak keys.

NPTEL Online Certification Courses
IIT Kharagpur

Now, there are so, these are the different interfaces that we have discussed in addition to the interfaces, there are many classes as you have already mentioned, these are the important classes which are basically used in programming namely Enum Map, the hash map, linked hash map and then Tmap.

(Refer Slide Time: 29:38)




A Quick Tour for Interfaces

NPTEL Online Certification Courses
IIT Kharagpur

Methods of Map interface

Method	Description
<code>void clear()</code>	Removes all key/value pairs from the invoking map.
default <code>V compute(K k, BiFunction<super K, ? super V, ? extends V> func)</code>	Calls <i>func</i> to construct an <i>cv</i> value. If <i>func</i> returns non- null , the new key/value pair is added to the map, any pre existing pairing is removed, and the new value is returned. If <i>func</i> returns null , any pre existing pairing is removed, and null is returned.
default <code>V computeIfAbsent(K k, Function<super K, ? Extends V> func)</code>	Returns the value associated with the key <i>k</i> . Otherwise, the value is constructed through a call to <i>func</i> and the pairing is entered into the map and the constructed value is returned. If no value can be constructed, null is returned.
default <code>V computeIfPresent(K k, BiFunction< superK, ? Super V, ? Extends V> func)</code>	If <i>k</i> is in the map, a new value is constructed through a call to <i>func</i> and the new value replaces the old value in the map. In this case, the new value is returned. If the value returned by <i>func</i> is null , the existing key and value are removed from the map and null is returned.
<code>boolean containsKey(Object k)</code>	Returns true if the invoking map contains <i>k</i> as a key. Otherwise, returns false .

Continued...



Now, let us have a quick tour to all interfaces that mean we will try to give an summary of what are the different methods are defined there and those methods are ultimately defined in the classes namely, Enum Map hash map, linked hash map, Tmap etc. So, we will just import familiarity will see, what are the different methods are the details discussion of the method, it is beyond the scope of this lecture and further detail discussion on you I should recommend you to check the oracle websites that is the document where the detailed documentation of each interfaces with examples are available.

So, there are many methods as we see here the compute method is basically how a key for a given key the hash value can be computed and is a compute if absent, there is a if present function and then contents key that mean whether, a key is available in the collection or not so, those are the methods are there.

(Refer Slide Time: 30:41)

Method	Description
<code>boolean containsValue(Object v)</code>	Returns true if the map contains <i>v</i> as a value. Otherwise, returns false .
<code>Set<Map.Entry<K, V>> entrySet()</code>	Returns a Set that contains the entries in the map. The set contains objects of type Map.Entry . Thus, this method provides a set-view of the invoking map.
<code>boolean equals(Object obj)</code>	Returns true if <i>obj</i> is a Map and contains the same entries. Otherwise, returns false .
<code>default void forEach(BiConsumer<? Super K, ? Super V> action)</code>	Executes <i>action</i> on each element in the invoking map. ConcurrentModificationException will be thrown if an element is removed during the process.
<code>V get(Object k)</code>	Returns the value associated with the key <i>k</i> . Returns Null if the key is not found.
<code>default V get(Object k, V defVal)</code>	Returns the value associated with <i>k</i> if it is in the map. Otherwise, <i>defVal</i> is returned.
<code>int hashCode()</code>	Returns the hash code for the invoking map.
<code>boolean isEmpty()</code>	Returns true if the invoking map is empty. Otherwise, returns false .

Continued...

NPTEL Online Certification Courses
IIT Kharagpur

Now, there are a few more methods also it is defined there here, also I have included here contents value it is basically if a particular object is present in the table or not equals whether two objects are same or not for each basically for the traversal here get is a very important get means, we have to access an elements.

So, get method have the two different version that means, given a k whether, how we can get the object from the table and then, hash code is a default method it is basically there which is basically written the hash code and whether the collection is empty or not that needs to be checked. So, these are the different methods by which we can simply know about the collection, a collection is in this case is a table.

(Refer Slide Time: 31:29)

Method	Description
default V <u>putIfAbsent</u> (K k, V v)	Inserts the key-value pair into the invoking map if this pairing is not already present or if the existing value is null . Returns the old value. The null value is returned when previous mapping exists, or the value is null .
V <u>remove</u> (Object k)	Removes the entry whose key equals <i>k</i> .
default boolean <u>remove</u> (Object k, Object v)	If the key-value pair specified by <i>k</i> and <i>v</i> is in the invoking map, it is removed and true is returned. Otherwise, false is returned.
default boolean <u>replace</u> (K k, V oldV, V newV)	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 true is returned. Otherwise, false is returned.
default V <u>replace</u> (K k, V v)	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, null is returned.
default void <u>replaceAll</u> (BiFunction<? Super K, ? Super V, ? Extends V> func)	Executes <i>func</i> on each element of the invoking map, replacing the element with the result returned by <i>func</i> . A ConcurrentModificationException will be thrown if an element is removed during the process.
int <u>size</u> ()	Returns the number of key/value pairs in the map.
Collection<V> <u>values</u> ()	Returns a collection containing the values in the map. This method provides a collection view of the values in the map.

So, there are so many methods which are defined there here, the put if absent the get method is basically to access it put is basically to store it so, there is a different method and then, put version is there and remove is basically to delete. So, there are many remove method is there replaces basically modification that means, if you want to modify certain key values or objects replace all if you want to modify all and then, size basically what is the size of that tables and then, value is basically all elements in the tables can be obtained using this method now, we will learn about the use of these methods while we discuss the programming.

(Refer Slide Time: 32:09)

Summary: Interfaces in Map

- Maps revolve around two basic operations: get() and put().
- To put a value into a map, use put(), specifying the key and the value.
- To obtain a value, call get(), passing the key as an argument. The value is returned.
- To obtain a collection-view of a map the entrySet() method can be used.
- To obtain a collection-view of the keys, use keySet().
- To get a collection-view of the values, use the method values().

Note: The Map interface is not a sub type of the Collection interface.

So, mainly so far the programme is concerned these are the methods we have to learn little bit sincerely so, that we can understand about it, when we will go for programming then, we will discuss about all these methods in details, the get and put method and then, entry set basically it will basically give you the set form of the collection or tables key set is basically set of all keys those are there in the table and values basically all elements those are there in the tables means, all records we can say so, these are the important methods basically we shall use in our programming or any programmer should use all those methods in their programming practice.

(Refer Slide Time: 32:49)

The SortedMap interface

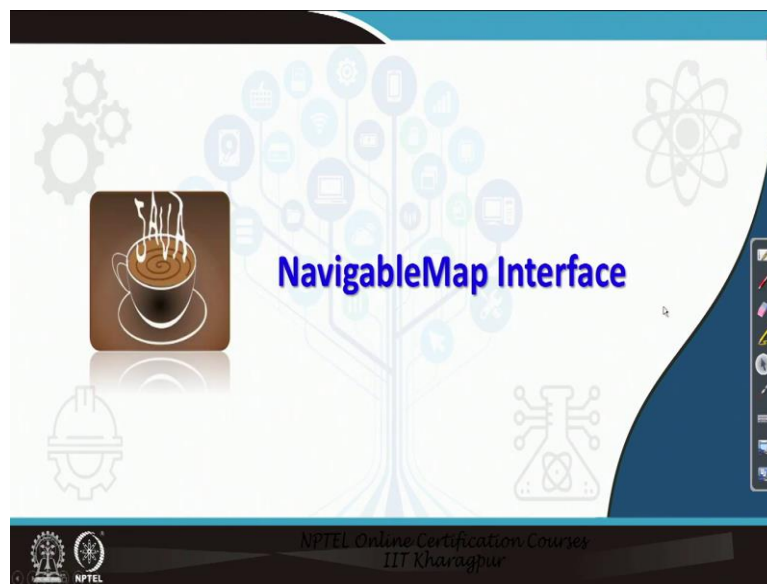
- The **SortedMap** interface extends **Map**. It ensures that the entries are maintained in ascending order based on the keys. **SortedMap** is generic like the interface **Map**. The methods defined in the **SortedMap** interface are listed below.

Method	Description
<code>Comparator<? Super K> comparator()</code>	Returns the invoking sorted map's comparator. If natural ordering is used for the invoking map, null is returned.
<code>K firstKey()</code>	Returns the first key in the invoking map.
<code>SortedMap<K, V> headMap(K end)</code>	Returns a sorted map for those map entries with keys that are less than <i>end</i> .
<code>K lastKey()</code>	Returns the last key in the invoking map.
<code>SortedMap<K, V> subMap(K start, K end)</code>	Returns a map containing those entries with keys that are greater than or equal to <i>start</i> and less than <i>end</i> .
<code>SortedMap<K, V> tailMap(K start)</code>	Returns a map containing those entries with keys that are greater than or equal to <i>start</i> .

NPTEL Online Certification Courses
IIT Kharagpur

Now, so like this map interface, there is sorted map interface also similar to the interface there are many methods which are declared there for example, comparator, head map this is the first elements the last element then, sub map subset of the table tail map the last few elements of the table given the starting values of key so many things are there and here sorted map is basically the difference between the map is that in case of map all the tables objects are not necessarily in sorted order but, in case of sorted map, it is basically sorted as for the key values of each objects so, this is the sorted map.

(Refer Slide Time: 33:32)



The NavigableMap Interface

- The **NavigableMap** interface extends **SortedMap** and declares the behavior of a map that supports the retrieval of entries based on the closest match to a given key or keys. The **NavigableMap** is also a generic interface like the **SortedMap** and **Map** interfaces. The methods defined in the **NavigableMap** interface are listed below.

Method	Description
<code>Map.Entry<K, V> ceilingEntry(K obj)</code>	Searches the map for the smallest key k such that $k \geq obj$. If such a key is found, its entry is returned. Otherwise, null is returned.
<code>K ceilingKey(K obj)</code>	Searches the map for the smallest key k such that $k \geq obj$. If such a key is found, it is returned. Otherwise, null is returned.
<code>NavigableSet<K> descendingKeySet()</code>	Returns a NavigableSet that contains the keys in the invoking map in reverse order. Thus, it returns a reverse set-view of the keys. The resulting set is backed by the map.
<code>NavigableMap<K, V> descendingMap()</code>	Returns a NavigableMap that is the reverse of the invoking map. The resulting map is backed by the invoking map.
<code>Map.Entry<K, V> firstEntry()</code>	Returns the first entry in the map. This is the entry with the least key.

Continued...

And navigable map is basically the idea about the closest key value if it is given there, if it does not match perfectly, then it will try to find the record or objects which are the closest match so, that is why the navigable map has been implemented there and it has many methods there is basically ceiling entry that means, it is some key values which is greater than some key value which is basically a ceiling key, ceiling entry and descending key set that mean it will give the set of values in descending order descending map it is all objects in descending order first entry which is a fast object it is there.

(Refer Slide Time: 34:13)

Method	Description
<code>Map.Entry<K, V> floorEntry(K obj)</code>	Searches the map for the largest key k such that $k < obj$. If such a key is found, its entry is returned. Otherwise, <code>null</code> is returned.
<code>K floorKey(K obj)</code>	Searches the map for the largest key k such that $k < obj$. If such a key is found, it is returned. Otherwise, <code>null</code> is returned.
<code>NavigableMap<K, V> headMap(K upperBound, Boolean incl)</code>	Returns a <code>NavigableMap</code> that includes all entries from the invoking map that have keys that are less than <code>upperBound</code> . If <code>incl</code> is <code>true</code> , the map element equal to <code>upperBound</code> is included. The resulting map is backed by the invoking map.
<code>Map.Entry<K, V> higherEntry(K obj)</code>	Searches the set for the largest key k such that $k > obj$. If such a key is found, its entry is returned. Otherwise, <code>null</code> is returned.
<code>K higherKey(K obj)</code>	Searches the set for the largest key k such that $k > obj$. If such a key is found, it is returned. Otherwise, <code>null</code> is returned.

Continued...

There are a few more methods also it is there like that floor entry as it is opposite to the ceiling entry below to a certain values the key and then floor key is all the objects all the key values basically certain value layer, which is basically the object that is there means, below objects or is basically a lower bound upper bound like so, there is also hate map giving the upper bound and then Boolean that means, whether will include all or not then higher entry higher key.

So, those are the methods basically we will discuss whenever we discuss programming in details, illustrating how these methods work, and then you will be able to understand but, this is for the summary only these are the methods have available for you so, that you can use it.

(Refer Slide Time: 35:03)

The Map classes

There are several classes to implement the map interfaces. All classes extends the `AbstractMap` class, which in turns implements the `Map` interface. This implies that all the methods in the `Map` interface are mostly defined in them in addition to some of their own methods. In the following, a brief description of each class followed by the constructors and methods are briefly discussed.

- Creating a map container
- Storing objects into a map container
- Retrieving objects from a map container
- Removing objects from a map container
- Collection view of a map container
- Bulk operations on a Map container

NPTEL Online Certification Courses
IIT Kharagpur

And there is so, in these are the different methods and likewise the map classes are there, the map classes are ultimately that you can create an object that is a collection object in your programme and then, it basically you can manipulate this programme and again I repeat a collection according to this map is just a table a collection according to the java collection framework that we have discussed about vector is a collection, array is a collection, array list is a collection, T is a collection, linked list is a collection those are different collection that we have discussed but, here all these map basically gives a new form of collection, this is called a tables actually and the table collection is characterised two sets on the other hand, all these collection are characters only one type generic type so, there is that difference.

And now, so far this map classes is concerned so, how a map classes it is also called container, how a collection or it is call rather we can say the container, a container can be created, how container can be sorted, how some objects in a container can be retrieved, how some objects in a container can be removed, how some container can be viewed and how the bulk operation copy from a set to another set all these things can be done or a table can be copied from another table or two table can be marched and so many things are there.

So, these are the basically activities or operations that we will be able to achieve using the different classes which is defined in the map framework and these are called a map classes and so far the different map classes are concerned I have already mentioned that four map classes are there.

(Refer Slide Time: 36:46)

Map classes

Class	Description
AbstractMap	Implements most of the Map interface.
EnumMap	Extends AbstractMap for use with enum keys.
HashMap	Extends AbstractMap to use a hash table.
TreeMap	Extends AbstractMap to use a tree.
LinkedHashMap	Extends HashMap to allow insertion-order iterations.
IdentityHashMap	Extends AbstractMap and uses reference equality when comparing documents.
WeakHashMap	Extends AbstractMap to use a hash table with weak keys.

The slide also features a class hierarchy diagram on the right side, showing the relationships between these classes. A small video inset of the presenter is visible in the bottom right corner of the slide.

So, those map classes are basically in Enum Map, the hash map, Tmap, linked hash map these two maps are not used for the programming purpose so they are not included in our discussion so, this is basically overall the map framework. Now, there are different methods truly the methods those are already learned about interfaces they have been defined all these interfaces so, there is no extra method in each classes actually to be studied further so, however only thing that needs to be understood about that, how the different collections or container objects can be created so, constructor needs to be there.

(Refer Slide Time: 37:25)

REFERENCES

- <https://cse.iitkgp.ac.in/~dsamanta/javads/index.html>
- <https://docs.oracle.com/javase/tutorial/>

So, in our few, in our next few slides, we will try to understand about how the different constructors can be created that will be discussed in our next video lectures, where we will discuss about the different classes and the different constructors, which are basically used to create the objects and using those objects we can manipulate different tables. Thank you.