Hashmaps:

Important in terms of java collection

Store data in the form of key-value pair.

It is a class which implements map interface.(Part of map family)

It inherits abstract map class.

It contains only unique elements.

It may have one null key and multiple null values.

It maintains no order.(since it does not store it based on index)

\*\*\*\*\*Hashmap is non synchronized.

🡪It means when we are running hashmap in multi threading env ,hash map can be accessed by more than one thread simultaneously.

When we have to increase performance,we use hashmap,instead of hashtable

Bcz, hashtable is synchronized (not thread safe)🡪only one thread can access a hashtable at a time.

Disadvantages:

Problem with the hashmap:

Consider the e.g.where e1 is the obj and values are:

Key : value

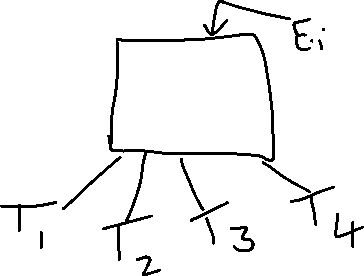
Jansi 29



Raj 33 (Employee obj)

Jansi 29

Raj 33



T1,T2,T3,T4 🡪threads that access e1 of a hasmap simultaneously

Scenario 1:when T1 updates Jansi : 30(instead of 29),T2 also will get the vhanges reflected in it.but it needs the data to be jansi:29.

Scenario 2: when T3 deletes Raj and its value from hashmap,and all other threads can noe have only one value in its hashmap.

**Non- synchronized**: If T1,T2,T3 are accessing simultaneously

**Synchronised:** : If T1,T2,T3 are accessing ,only if T1 releases,T2 can access.

This is called as **Fail - fast Condition**

It results in **Concurrent modification exception**

🡪If any thread modifies the map structure or if it adds or removes the element from hashmap object,this is called fail fast condition or Concurrent modification exception.

How to overcome the exception:

So,we can use hashmap to get the values without updating it in case of multi-threading .

We can also use concurrent hash maps.

Program:

HashMap<String,Integer> h=**new** HashMap <String,Integer>();

h.put("Jansi", 29);

h.put("Raj",33);

h.put("Theju", 3);

System.***out***.println(h.get("Atharva"));

//o/p- null It does not show any array out of bound exception since it does not store based on indexes

System.***out***.println(h);

System.***out***.println("keyssssssssssssss");

//to print all keys and values in hashmap-->use foreach loop with entryset method

**for**(Entry s:h.entrySet())

{

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

//storing Class objects in hash map-->here employee class objects.

HashMap<Integer,Employee> hm=**new** HashMap<Integer,Employee>();

Employee e1=**new** Employee("Lavan",26);

Employee e2=**new** Employee("Vignesh",29);

Employee e3=**new** Employee("Saraswathi",55);

Employee e4=**new** Employee("Lakshmanan",60);

hm.put(1, e1);

hm.put(2, e2);

hm.put(3, e3);

hm.put(4, e4);

//traverse the hashmap,use for loop similar to foreach concept with entryset method

**for**( Entry<Integer,Employee> e:hm.entrySet())

{

**int** key=e.getKey();

Employee emp=e.getValue();

System.***out***.println(key +":" + emp.age + " "+emp.name);

}

///----Employee class code:

**public** **class** Employee {

String name;

**int** age;

**public** Employee(String name, **int** age) {

**this**.name = name;

**this**.age = age;

}

}

Hash table:

Similar to hashmap but it is synchronized.(cannot be used in multi-threading concept).

Stores data in key-value format.

Key will be specified as an object and each and every object inside java has a special identity called hash code.

Hash code:

Whenever we create an object in hash-table ,Java provides a number (object number) for that particular object.It is a 32-bit int number.

<key,value>



<1,”jansi”>



🡪intenally, hash table stores the key in the form of hash code.

<343434334,”jansi”>

Difference between Hashmap and Hashtable

Hashmap Hashtable

🡪Asynchronised 🡪synchronised

🡪used in multi-threading 🡪not used in multi-threading

🡪performance is slow 🡪performance is fast

🡪intrenally,keys are stored in the form of 🡪hash codes are not used here

Hash-codes. To store keys

🡪null values are allowed 🡪null values are not allowed

One null key is allowed strictly,null pointer exception

Multiple null values are allowed will be thrown

🡪unique keys are only allowed.

One key cannot contain many values.

e.g program for a hashtable:

Hashtable<Integer,String> h=**new** Hashtable<Integer,String>();

h.put(1, "Jansi");

h.put(2, "Raj");

//create a clone copy /shallow copy of hash-table-->using clone method

Hashtable hclone=**new** Hashtable();

//duplicate object of h

hclone=(Hashtable) h.clone();//typecasting to hashtable

System.***out***.println("original"+h);

System.***out***.println("clone:"+hclone);

//deletinh h object,all datas of original will be cleared,but cloned copy will remain as such

h.clear();

System.***out***.println("\*\*\*\*after clearing\*\*\*\*");

System.***out***.println("original"+h);

System.***out***.println(hclone);

System.***out***.println("xxxxxxxxxxxxxxx");

e.g. prog to print its values in different ways:

Hashtable<String,String> con=**new** Hashtable <String,String>();

con.put("Jansi", "Lakshmanan");

con.put("Rajagopal", "sridhar");

con.put("Thejaswini", "rajagopal");

//Contains method to check whether key or value is present

**if**(con.containsKey("Jansi"))

{

System.***out***.println("Key found");

}

//to print all values from hashtable using enumeration .--returns only values not keys

System.***out***.println("sequential order using enumeration ");

Enumeration e= con.elements();

**while**(e.hasMoreElements())

{

System.***out***.println(e.nextElement());

}

o/p:🡪only values not keys

sequential order using enumeration

sridhar

Lakshmanan

rajagopal

//to print all values using entry set

System.***out***.println("sequential order using entry set ");

**for**(Entry<String,String> e1:con.entrySet())

{

System.***out***.println(e1.getKey() +" " +e1.getValue());

}

o/p:🡪both keys and values

sequential order using entry set

Rajagopal sridhar

Jansi Lakshmanan

Thejaswini rajagopal

//to print all values directly:returns in key=value format

System.***out***.println("non-sequential order");

System.***out***.println(con);

o/p: non-sequential order

{Rajagopal=sridhar, Jansi=Lakshmanan, Thejaswini=rajagopal}

//check hash tables are equal or not using contains method

Set s =con.entrySet();

**if**(con.equals(hclone)) {

System.***out***.println("both are equal");

}

**else**

{

System.***out***.println("both are not equal");

}

//To get the hash code of object

System.***out***.println(con.hashCode());

o/p:

//it allows only unique values -->e.g

con.put("thejaswini", "Raj");

System.***out***.println("unique values check :only if value is changed and key remains same");

System.***out***.println(con);

o/p:🡪checks the unique values based on key,if key is same,then it allows only one entry

{Rajagopal=sridhar, Jansi=Lakshmanan, Thejaswini=Raj}

System.***out***.println("unique values check :both key and value is not changed");

con.put("thejaswini", "Rajagopal");

System.***out***.println(con);

o/p: does not allow duplicate entry

{Rajagopal=sridhar, Jansi=Lakshmanan, Thejaswini=Rajagopal}

//null vlaues check 🡪does not allow null values in both key and values.

System.***out***.println("Null vlaues check");

con.put(**null**, "Rajagopal");

o/p: java.lang.NullPointerException will be thrown

How to use hash maps in selenium?

e.g.We can create a hashmap and store all the user credentials.

Here,admin and customer are key and [rajagopisri@gmail.com\_raj->has](mailto:rajagopisri@gmail.com_raj-%3ehas) both username and pwd connected by an underscore.

HashMap<String, String> h = **new** HashMap<String, String>();

h.put("admin", "rajagopisri@gmail.com\_raj");

h.put("customer", "jansilakshmanan90@gmail.com\_test123");

**return** h;

}

**public** **void** storedatatest() **throws** Exception {

//call the function and get only values of customer credentials key

String username = Hashmap\_selenium\_concept.*storedata*().get("customer");

//split the value using split method and store it in string array where username and pwd will be stored in 0th and 1st index resp.

String credentials[] = username.split("\_");

//get the username based on array index.

driver.findElement(By.*name*("email")).sendKeys(credentials[0]);

driver.findElement(By.*name*("password")).sendKeys(credentials[1]);

driver.findElement(By.*xpath*("//\*[@class='ui fluid large blue submit button']")).click();

Thread.*sleep*(5000);



Map interface in java collections.-->rectangle-interface,circle-class

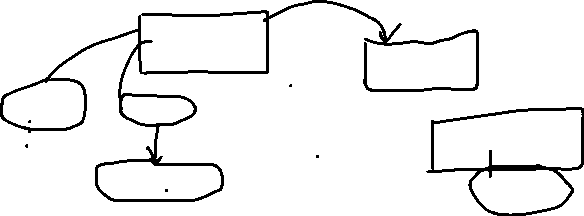
🡪part of collection framework extends



Map interface

Sorted Map Interface

Hashtable Hashmap



extends Navigable Map(interface)

Linked Hashmap Treemap



Examples of map:

1.Error codes

(1.2.1 : “fatal error”)

2.map of .Zipcodes

3.Map of managers and employees

4.Test data storage

5.store some configuration

6.env.variables that we store in properties file can also be stored in map

7.Rest API automation where we can store header

Methods in map:

1.put()-<to insert key,values

2.putall()🡪to add map inside a map

3.get()🡪 to get key,values

4.contains()🡪to check whether key /value is available

5.remove (key)

6.keyset()🡪returns set view which contains all the keys where we can iterate

7.entryset()🡪returns set view containing both keys and values