OBJECT ORIENTED PROGRAMMING USING

Java

Let's explore technology
together to live in the future

Java Hashmap: → Associative arraylist

In the Arraylist chapter, you learned that arrays store items as ordered collection, and you have to access them with index num.
A Hashmap stores items in "key/value" pair you can access them by index of another type.

One object is used as key (index) to another object (value).
It can store different types
String keys and Integer values
String keys and String values

The Hashmap class of Java Collections provi functionality of hash table data structure.
It stores elements in Key/value pairs.
Here keys are unique identifiers used to associate each value on a map.

Create a HashMap:

```
HashMap <K , V> num = new HashMap<K,V>(

import java.util.HashMap;
class Main {
public static void main (String [] args)
{
HashMap<String, Integer> languages = new
                        HashMap < String, Integer>
languages.put ("Java", 8);
languages.put ("JavaScript", 1);
languages.put ("Python", 3);

languages.put ("C++");
System.out.println ("HashMap: "+ languages);

}
```

Basic Operations on Java Hashmap:

Add elements: To add a single element to hashmap, we use put() method of HashMap

```
import java.util.HashMap;
public class Main{
 public static void main (String[]args)

HashMap <String, String> capitaCities = new
                    HashMap<String, String>();

capitalCities.put ("England", "London");
capitalCities.put ("Germany", "Berlin");
capitalCities.put ("Norway", "Oslo");

 System.out.println (capitalCities);
}
```

Access an Item: To access a value in HashMap, use get() method and refer to its key.

```
import java.util.HashMap;
public class Main{
 public static void main (String []args)
{ HashMap<String, String> capitalCities = new
                        HashMap <String,String>(

 capitalCities.put ("England ", "London ");
 capitalCities. put ( "Germany", "Berlin");
 capitalCities. put ("Norway", "Oslo");

 System.out.println (capitalCities.get ("England");

 }
}      London
```

Remove an Item: To remove an item, use the remove() method and refer to the key.

```
 import java.util.HashMap;
 public class Main{
  public static void main (String []args)
{ HashMap <String, String> capitalCities = new
                    HashMap<String,String>();

 capitalCities.put ("England", "London");
 capitalCities.put ("Germany", "Berlin");
 capitalCities.put ("Norway", "Oslo");
 capitalCities.remove( "England ");
System.out.println ( capitalCities);

 }
}
```

To remove all items, use the clear() method:

```
import java.util.HashMap;
public class Main {
 public static void main (String []args)
{ HashMap <String, String> capitalCities = new
                                   HashMap <String, String>();
 capitalCities.put ("England", "London");
 capitalCities.put ("Germany", "Berlin");
 capitalCities.put ("Norway", "Oslo");

 capitalCities.clear ();
 System.out.println (capitalCities);

}}   { }
```

HashMap Size: To find out how many items there are, use the size() method.

```
import java.util.HashMap;
public class Main {
 public static void main (String []args)
{ HashMap <String, String>  capitalCities = new
                              HashMap < String, String>();

 capitalCities.put ("England", "London");
 capitalCities.put ("Germany", "Berlin");

 capitalCities.put ("Norway", "Oslo");

 System.out.println (capitalCities.size ());
}}
```

Access the HashMap: We can also access the keys, values and key/value pairs of HashMap using keySet(), values()

```
import java.util.HashMap;
class Main {
 public static void main (String []args)
{ HashMap < Integer, String > languages =new
                       HashMap <Integer, String>(

 languages.put (1, "Java");
 languages.put (2, "Python");
 languages.put (3, "Javascript");

 System.out.println ("HashMap:"+ languages);

 System.out.println (" Keys:" + languages.
                        keyset());

System.out.println ("Values:"+languages.values()

  }
}
```

```
HashMap : {1 = Java , 2 = Python, 3= Javascript}

Keys: [1, 2, 3]

Values: [ Java, Python , Javascript ]
```

Loop through HashMap:

We can loop through the items of HashMap with for-each loop.

// Print keys

```
import java.util.HashMap;
 public class Main {
  public  static  void  main (String []args)
{
 HashMap<String, String> capitalCities = new
                      HashMap<String,String>();

  capitalCities.put ("England", "London");
  capitaCities.put ("Germany", "Berlin");
  capitalCities.put ("Norway", "Oslo");

  for (String i : capitaCities.keySet())
 {   System.out.println (i);
  }
  for (String j : capitalCities.values())
 {
     System.out.println (j);
  }
```

```
for (String t : capitalCities.keySet)
{
  System.out.println("Key:" + i + "value:"
                          + capitalCities.get(i
}
}
```

```
USA
Norway
England
Germany

Washington DC
Oslo
London
Berlin

Key: Norwg value: Oslo
Key: England value: London
Key: Germany value: Berlin
```

Java HashSet:

A HashSet is a collection of items where every item is unique, and it is found in the java.util package.
The HashSet class of the Java Collections framework provides functionalities of the hash table data structure.

Example:
Create a HashSet object called cars that will store Strings.

```
import java.util.HashSet;
HashSet<String> cars = new HashSet<String>();
```

HashSet Methods

-Add Items : The HashSet class has many useful methods. For example to add items to it use the **add()** method.

```
import java.util.HashSet;
public class Main {
 public static void main (String []args)

HashSet<String> cars = new HashSet<String>();
  cars.add ("Volvo");
  cars.add ("Ford");
  cars.add ("BMW");
  cars.add ("Mazda");

System.out.println (cars);
}
```

2- Check if an item Exists:

To check whether an item exists in HashSet use the contains() method.

```
import java.util.HashSet;
public class Main {
 public static void main (String[] args)
{
 HashSet<String> cars = new HashSet<String>(
  cars.add ("Volvo");
  cars.add ("Ford");
  cars.add ("BMW");
 System.out.println (cars.contains("Ford"));
}}
```

```
true
```

3- Remove an Item:

To remove an item, use the remove() method.

```
import java.util.HashSet;
public class Main {
 public static void main (String [] args)
{
 HashSet <String> cars =new HashSet<String>(
  cars.add ("Volvo");
  cars.add ("BMW");
  cars.add ("Ford");
  cars.remove ("Volvo");
 System.out.println (" cars);
}}
```

```
[Ford, BMW]
```

Loop through a HashSet:

Loop through the items of HashSet with a for-each loop.

```
import java.util.HashSet;
public class Main{
 public static void main (String [] args)
{
{ Hashset <String> cars = new HashSet<String>();
  cars.add ("Volvo");
  cars.add ("BMW");
  cars.add ("Ford");
  cars.add ("Mazda");
 for (String i : cars)
{
{ System.out.println (i);
}}
```

```
Volvo
Mazda
Ford
BMW
```

Java Iterator:
An iterator is an object that can be used to loop through collections like ArrayList and Hashset.
It is called iterator bcz "iterating" is the technical term for looping.

Getting an Iterator:
The iterator() method can be used to get an Iterator for any Collection.

```
import java.util.Arraylist;
import java.util.Iterator;
public class Main {
 public static void main (String[] args)
{ Arraylist <string> cars = new Arraylist<String>(

   cars.add ("Volvo");
   cars.add ("Ford");
   cars.add ("BMW");
   cars.add ("Mazda");

 Iterator <String> it = cars.iterator();
```
$\underbrace{\text{iterator}}_{\text{loop of } x}$

```
  System.out.println (it.next());
```
$\underbrace{\text{it.next()}}_{\text{prints first element in this case}}$

```
  } Volvo
}
```

"it" - object which has iteration of x

Loop through a Collection:

To loop through a collection, use the hasNext() and next() methods of Iterator:

```
 import java.util.ArrayList;
 import java.util.Iterator;
 public class Main {
  public static void main (String[]args)
{ ArrayList<String> cars = new ArrayList<String>();

  cars.add ("Volvo");
  cars.add ("BMW");
  cars.add ("Ford");

  Iterator<String> it = cars.iterator();

  while (it.hasNext())
 {
  System.out.println (it.next());
  }
 }
```

```
Volvo
BMW
Ford
Mazda
```

Removing Items from a Collection:

Iterators are designed to easily Change the collections that they loop through. The remove() method can remove Items from a collection while looping.

```
import java.util.ArrayList;
import java.util.Iterator;
public class Main {
 public static void main (String [] args)
{
 Arraylist <Integer> numbers = new ArrayList
                              <Integer>();
  numbers.add (12);
  numbers.add (8);
  numbers.add (2);
  numbers.add (23);

Iterator<Integer> it = numbers.iterator ();

 while (it.hasNext())
{
   Integer  i = it.next();

  if (i<10)
 {
    it.remove();
 }
}
  System.out.println (numbers);
 }
}
```

Note:

Trying to remove items using a for loop or a for-each loop wouldn't work bcz collection is changing size at same tim that code is trying to loop

```
import  java.util.Iterator;
import  java.util.Arraylist;
public  class  Main{
 public  static  void  main (String[]args)
{ Arraylist<String> name = new ArrayList<String>();

   name.add ("ali");
   name.add ("ahmed");

Iterator<String> it = name. iterator();

  int  y=0;
  while  (it.hasNext());
{  System.out.println (it.next());
   if (y==2)
   { it. remove(y);}
     y++;
}
```