**ASSIGNMENT NO – 1**

**TITLE :** Design a system with the help of advance datastructures , collections and generics in java.

**DATE:**

**REMARK :**

**ASSIGNMENT – 1**

**TITLE:** Design a system with the help of advance data structures, collections and generics in Java

**OBJECTIVE :**

A data structure is a specialized format for organizing and storing data. General data structure **types** include the array, the file, the record, the table, the tree, and so on. Any data structure is designed to organize data to suit a specific purpose so that it can be accessed and worked with in appropriate ways.

**PROBLEM STATEMENT :**

Design a system using Hash Table in Java and enhance the system using advanced data structures.

**THEORY :**

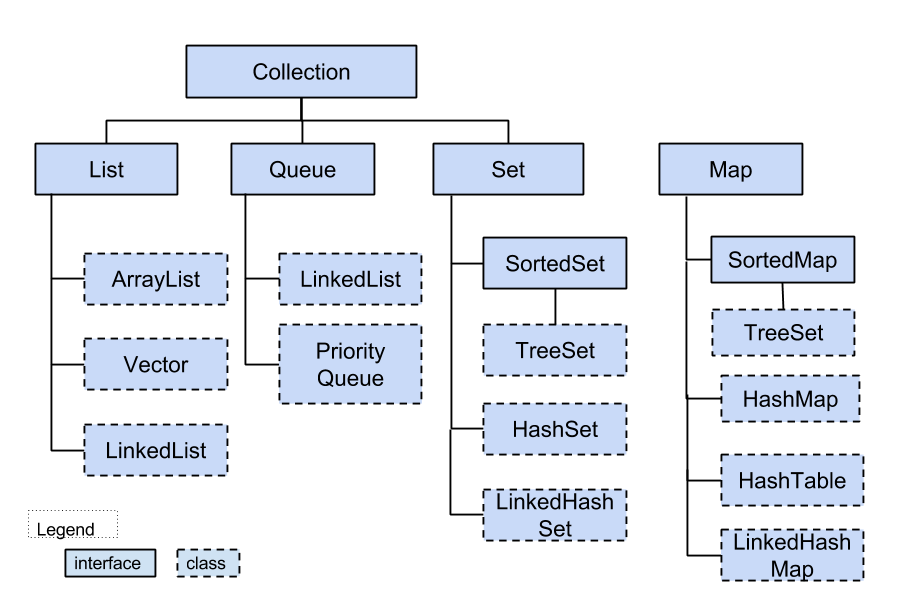
In programming, a ***data structure*** is a particular way of storing and manipulating the internal data of a computer program. There are many data structures, and you probably have used some of them.

For example, an array of int is a data structure that consume n entries in memory, where n is the size of the array. Each entry stores an int and can be accessed or modified via their index.

As the way to store and manipulate internal data mostly determines how much resources and time the application consumed to carry out its tasks, choosing an appropriate data structures can boost the performance of your program and, quite often, reduces implementation efforts.

Therefore, an important part of programming is to determine the right data structures for a given problem. In the coming articles, we will learn how to use and analyze common data structures in Java.

**DIAGRAMATIC REPRESENTATION**

[](http://toolsqa.wpengine.com/wp-content/uploads/2015/04/Collections.png)

ArrayList is a part of [collection framework](https://www.geeksforgeeks.org/collections-in-java-2/) and is present in java.util package. It provides us dynamic arrays in Java. Though, it may be slower than standard arrays but can be helpful in programs where lots of manipulation in the array is needed.

* ArrayList inherits AbstractList class and implements List interface.
* ArrayList is initialized by a size, however the size can increase if collection grows or shrunk if objects are removed from the collection.
* Java ArrayList allows us to randomly access the list.

|  |  |
| --- | --- |
| **Sr.No.** | **Constructor & Description** |
| 1 | **ArrayList( )**  This constructor builds an empty array list. |
| 2 | **ArrayList(Collection c)**  This constructor builds an array list that is initialized with the elements of the collection **c**. |
| 3 | **ArrayList(int capacity)**  This constructor builds an array list that has the specified initial capacity. The capacity is the size of the underlying array that is used to store the elements. The capacity grows automatically as elements are added to an array list |

This class implements a hash table, which maps keys to values. Any non-null object can be used as a key or as a value.  
To successfully store and retrieve objects from a hashtable, the objects used as keys must implement the hashCode method and the equals method.

* It is similar to HashMap, but is synchronised.
* Hashtable stores key/value pair in hash table.
* In Hashtable we specify an object that is used as a key, and the value we want to associate to that key. The key is then hashed, and the resulting hash code is used as the index at which the value is stored within the table.

**Constructors:**

* **Hashtable():** This is the default constructor.
* **Hashtable(int size):** This creates a hash table that has initial size specified by size.
* **Hashtable(int size, float fillRatio):** This version creates a hash table that has initial size specified by size and fill ratio specified by fillRatio. **fill ratio:** Basically it determines how full hash table can be before it is resized upward.and its Value lie between 0.0 to 1.0
* **Hashtable(Map m):** This creates a hash table that is initialised with the elements in m.

**Methods:**

1. **void clear() :** method clears the hashtable so that it contains no keys.
2. **Syntax :** public void clear()
3. **Returns :** NA
4. **Exception :** NA
5. **Object clone() :** used to create a shallow copy of this hashtable.
6. **Syntax :** public Object clone()
7. **Returns :**method call returns a clone of the hashtable.
8. **Exception :** NA

|  |
| --- |
| // Java code illustrating clear() and clone() methods  import java.util.\*;  class hashTabledemo  {  public static void main(String[] arg)  {  //creating a hash table  Hashtable h = new Hashtable();  Hashtable h1 = new Hashtable();  h.put(3, "Geeks");  h.put(2, "forGeeks");  h.put(1, "isBest");  // create a clone or shallow copy of hash table h  h1 = (Hashtable)h.clone();  // checking clone h1  System.out.println("values in clone: " + h1);  // clear hash table h  h.clear();  // checking hash table h  System.out.println("after clearing: " + h);  }  } |

1. Output:
2. values in clone: {3=Geeks, 2=forGeeks, 1=isBest}
3. after clearing: {}

**SYNTAX / CODE :**

import java.util.Hashtable;

import java.util.Scanner;

class hash\_table

{

public static void main(String args[])

{

int num = 0,ch;

String cho;

Scanner sc = new Scanner(System.in);

Scanner s = new Scanner(System.in);

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

do

{

System.out.println("Welcome to the Restaurant Sales");

System.out.println("enter 1 to insert details,2 to search,3 to delete,4 to display details,5 exit");

System.out.println("enter your choice");

ch=sc.nextInt();

switch(ch)

{

case 1:

System.out.println("How many records do you wish to insert");

num=sc.nextInt();

System.out.println("enter the item code");         //key

System.out.println("enter the food item");         //value\_1

//System.out.println("enter the cost for the food item");          //value\_2

for(int i=0;i<num;i++)

{

int key=sc.nextInt();

String val=s.nextLine();

h.put(key,val);

}

break;

case 2:

System.out.println("enter the value to be item code to be searched");

int mat=sc.nextInt();

if(h.containsKey(mat))

{

System.out.println("present"+" "+h.get(mat));

}

else

{

System.out.println("not present");

}

break;

case 3:

System.out.println("enter the key you wish to remove");

int del=sc.nextInt();

for(int j=0;j<num;j++)

{

if(h.containsKey(del))

{

h.remove(del);

}

}

System.out.println("elements are");

System.out.println(h);

break;

case 4:

System.out.println("to display the details");

for(int i=0;i<num;i++)

{

System.out.println(h);

}

case 5:

System.out.println("to exit");

break;

default:System.out.println("wrong choice");

}

System.out.println("wish to perform any operation again");

cho=s.next();

}

while(cho.equalsIgnoreCase("yes"));

}

}

**OUTPUT:**

The output for the data structures.

Welcome to the Restaurant Sales

enter 1 to insert details,2 to search,3 to delete,4 to display details,5 exit

enter your choice

1

How many records do you wish to insert

1

enter the item code

enter the food item

12

Noodles

wish to perform any operation again

yes

Welcome to the Restaurant Sales

enter 1 to insert details,2 to search,3 to delete,4 to display details,5 exit

enter your choice

2

enter the value to be item code to be searched

12

present Noodles

wish to perform any operation again

yes

Welcome to the Restaurant Sales

enter 1 to insert details,2 to search,3 to delete,4 to display details,5 exit

enter your choice

4

to display the details

{12=Noodles}

to exit

wish to perform any operation again

n

**PLATFORM:**

Net Beans and JDK8.0.

**CONCLUSION:**

Therefore we are using The HashTable Standard Library to Implement a Restaurant Management System for faster retrieval and access.