The following figure demonstrates the Java Collection framework.

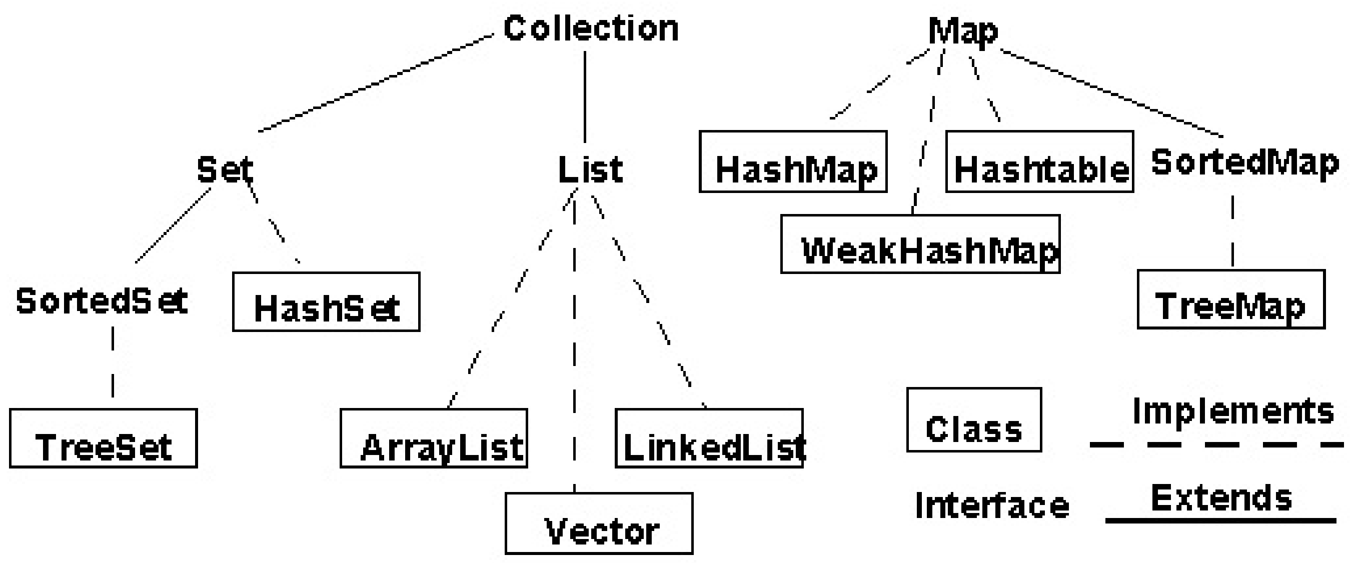


Figure 1: Java Collection Framework

Of these I shall discuss the following :

- ArrayList
- HashTable
- HashMap
- HashSet
- LinkedList

ArrayList in Java

ArrayList is a collection API used for storing elements using dynamic array. The dynamic array is an array in which the array size is not fixed in advance. Therefore, we can change the size of an array at run time using the ArrayList class. When we store elements into ArrayList, depending on the number of elements, the memory is allotted and re-allotted to accommodate all the elements. Every instance of the ArrayList class is allowed to store a set of elements in the list.

Some of the important points about ArrayList class shown below :

- ArrayList is not synchronized.
- ArrayList supports dynamic array which can grow as needed.
- Size of ArrayList can be dynamically increased or decreased.
- ArrayLists are created with initial size.
- In Java, standard arrays are of fixed length. After arrays are created, they cannot grow or shrink means you must know in advance how many elements an array will hold.
- ArrayList can contain duplicate elements.
- ArrayList maintains insertion order of the elements.
- Retrieval is random access because array works at index basis.

ArrayList class extends **AbstractList.** It also implements the **List** interface. It is generic class which has declaration as follows :

class ArrayList<E>

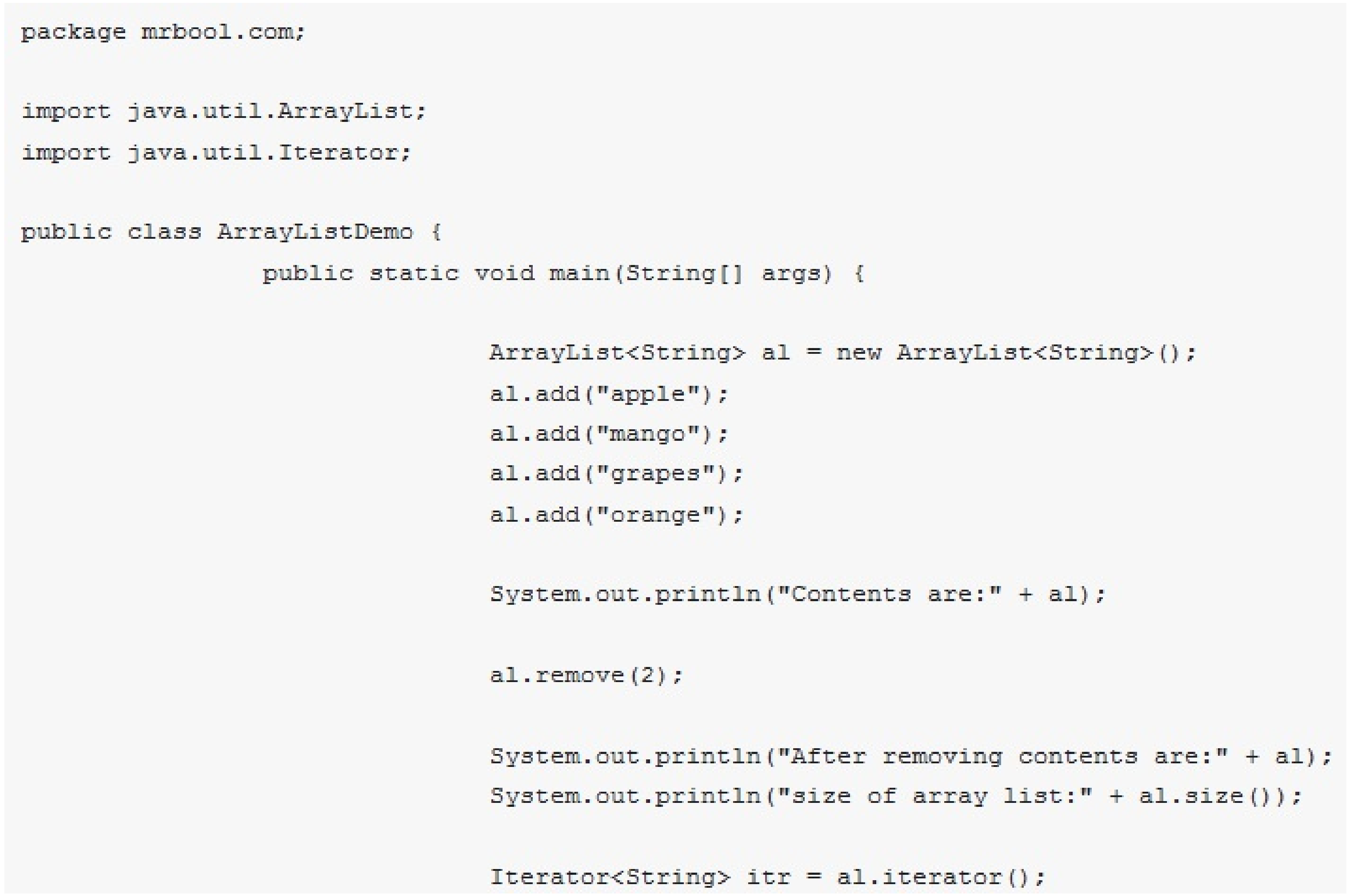
Where, E specifies type of objects to be stored in ArrayList. For example :

```
ArrayList<String> al=new ArrayList<String> ();
```

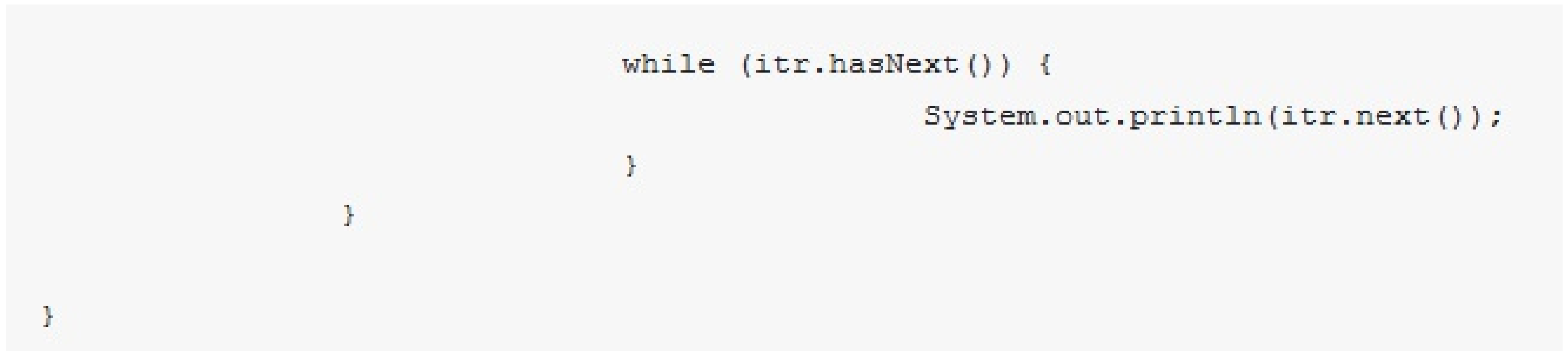
ArrayList has following constructors:

- ArrayList (): It builds an empty array list.
- ArrayList (Collection c): It builds an array list which is initialized with elements of the collection c.
- ArrayList (int capacity): It builds array list which has specified initial capacity used to store the elements. It grows automatically when elements are added to the array list.

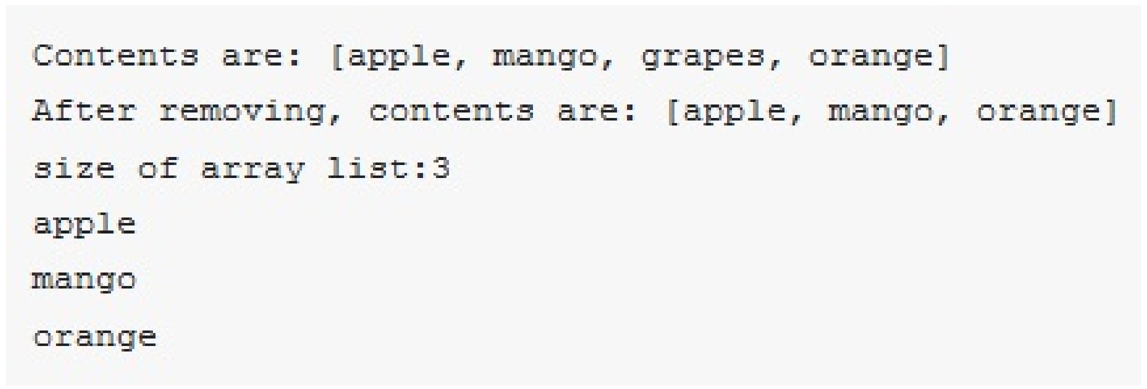
Method	Description
void add(int position, element obj)	It inserts specified element at the specified position in the ArrayList.
boolean add(element obj)	It appends specified element to the end of the ArrayList.
boolean addAll(Collection c)	It appends all the elements of the collection to the end of the ArrayList.
element remove(int position)	It removes specified element at the specified position in the ArrayList.
boolean remove(object obj)	It removes first occurrence of specified element obj from the ArrayList.
void clear()	It removes all the elements from the ArrayList.
boolean contains(Object o)	It returns true if ArrayList contains the specified element .
object get(int position)	It returns the element at the specified position in the ArrayList.
int indexOf(Object o)	It returns first occurrence of the specified element in the list or -1 if element not found in the list.
int lastIndexOf(Object o)	It returns the last occurrence of the specified element in the list or -1 if the element is not found in the list.
int size()	It returns the number of elements in the list.



```java
package mrbool.com;

import java.util.ArrayList;
import java.util.Iterator;

public class ArrayListDemo {
                public static void main(String[] args) {

                                ArrayList<String> al = new ArrayList<String>();
                                al.add("apple");
                                al.add("mango");
                                al.add("grapes");
                                al.add("orange");

                                System.out.println("Contents are:" + al);

                                al.remove(2);

                                System.out.println("After removing contents are:" + al);
                                System.out.println("size of array list:" + al.size());

                                Iterator<String> itr = al.iterator();
```



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

}
```



```
Contents are: [apple, mango, grapes, orange]
After removing, contents are: [apple, mango, orange]
size of array list:3
apple
mango
orange
```

HashTable

A HashTable is an array of the list. It was part of the java.util and is extended in the Dictionary class. HashTable was re-engineered to implement the Map interface. HashTable is similar to HashMap which can store elements in the form of key-value pairs and it is synchronized. It contains unique elements and neither the keys nor the values can be null.

class HashTable<K, V>

Where K specifies type of keys, and V specifies type of values.

We can create HashTable as follows:

```
HashTable<String, Integer> ht=new HashTable<String, Integer> () ;
```

The HashTable contains the following constructors.

- **HashTable ():** It is the default constructor.
- **HashTable (int size):** It creates hash table that has initial size specified by size. (The default size is 11.)
- **HashTable (int size, float fillRatio)**: It creates hash table that has initial size specified by size and fill ratio specified by fillRatio. This ratio must be between 0.0 and 1.0.
- **HashTable (Map m):** It creates hash table that is initialized with elements in m.

Method	Description
void clear()	It resets and removes all the key value pairs from the HashTable.
Object clone()	It returns duplicate of invoking object.
boolean contains (object value)	It returns true if the value that is equal to the value parameter exists and returns false if value is not found
boolean containsKey (object key)	It returns true if the key that is equal to the key parameter exists and returns false if key is not found
boolean containsValue (object value)	It returns true if the value that is equal to value parameter exists and returns false if value is not found
Enumeration elements	It returns enumeration of the values in hash table./td>
V get(Object key)	It returns the object associated with key and it returns null if the key is not in hash table.
boolean isEmpty()	It returns true if there are no key value pairs in the hash table.
Enumeration keys()	It returns enumeration of the keys in hash table.
V put(K key, V value)	It inserts key and value in the hash table.
int size()	It returns number of key value pairs in the hash table.
String to String	It returns string equivalent of hash table.

```
package mrbool.com;

import java.util.Enumeration;
import java.util.Hashtable;

public class HashTableDemo {
                public static void main(String[] args) {
                               Hashtable<String, String> ht = new Hashtable<String, String>()
                               ht.put("player 1", "sachin");
                               ht.put("player 2", "sehwag");
                               ht.put("player 3", "dhoni");

                               Enumeration<String> values = ht.keys();
                               while (values.hasMoreElements()) {
                                               String str = (String) values.nextElement();
                                               System.out.println(str + ":" + ht.get(str));
                               }
```

```
player 3:dhoni
player 2:sehwag
player 1:sachin
```

class HashMap<K, V>

Here, K specifies the type of keys and V specifies the type of values.

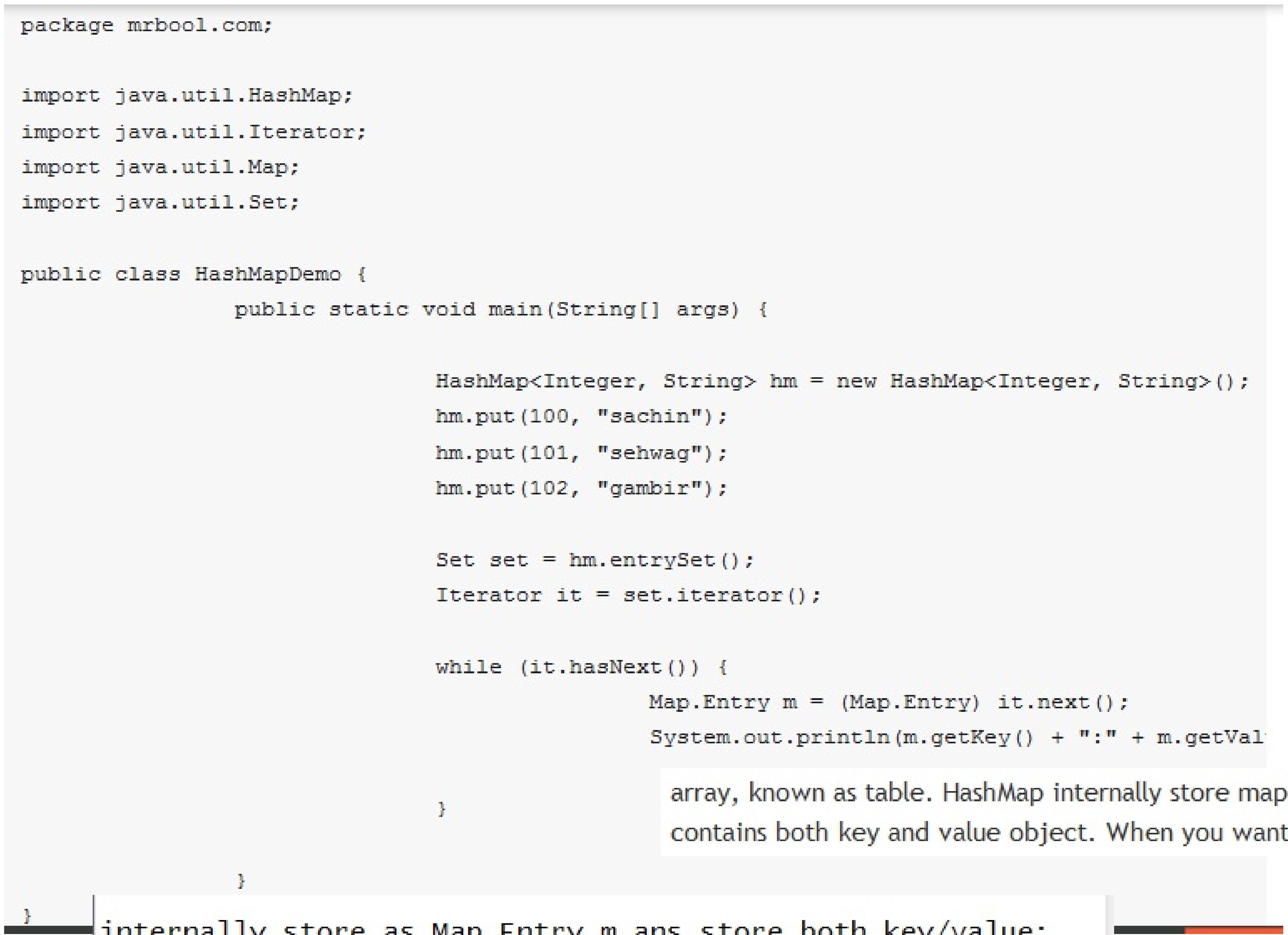
We can create HashMap as follows:

```
HashMap<String, Integer > hm=new HashMap<String, Integer> () ;
```

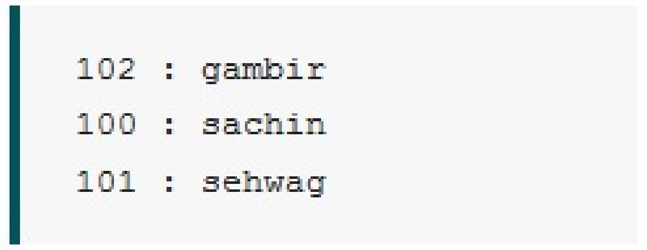
There are four types of constructors in HashMap as shown below :

- **HashMap ()**: It is a default hash map.
- **HashMap (Map m)**: It initializes the hash map by using the elements of m.
- **HashMap (int capacity):** It initializes the capacity of the hash map to capacity. The default initial capacity of HashMap will be 16 and the load factor will be 0.75.Load factor represents at what level HashMap should be doubled.
- **HashMap (int capacity, float fillRatio)**: It initializes both capacity and fill ratio by using its arguments.

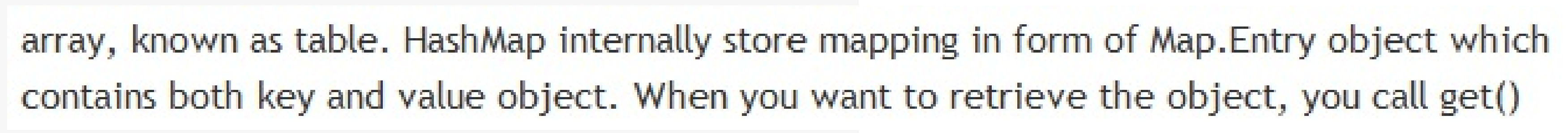
Method	Description
void clear()	It resets and removes all the key value pairs from the HashMap.
Object clone()	It returns duplicate copy of HashMap instance.
boolean containsKey (object key)	It returns true if map contains mapping for the specified key.
boolean containsValue (object value)	It returns true if map maps one or more keys to the specified value.
Set enrtySet()	It returns true if collection of mappings found this map.< /td>
Object get(Object key)	It returns the value if specified key is mapped or null if it contains no mapping for the value.
boolean isEmpty()	It returns true if there are no entries in the hash map.
Set keySet	It returns set of keys in map object.
Object put(Object key, Object value)	it stores key-value pairs into the hash map.
putAll(map m)	Copy all the mappings to another map.
Object remove(object key)	it removes the key and corresponding value from the Hashmap.
int size()	It returns number of key value pairs in the hash map.
Collection values()	It returns collection view of map values.



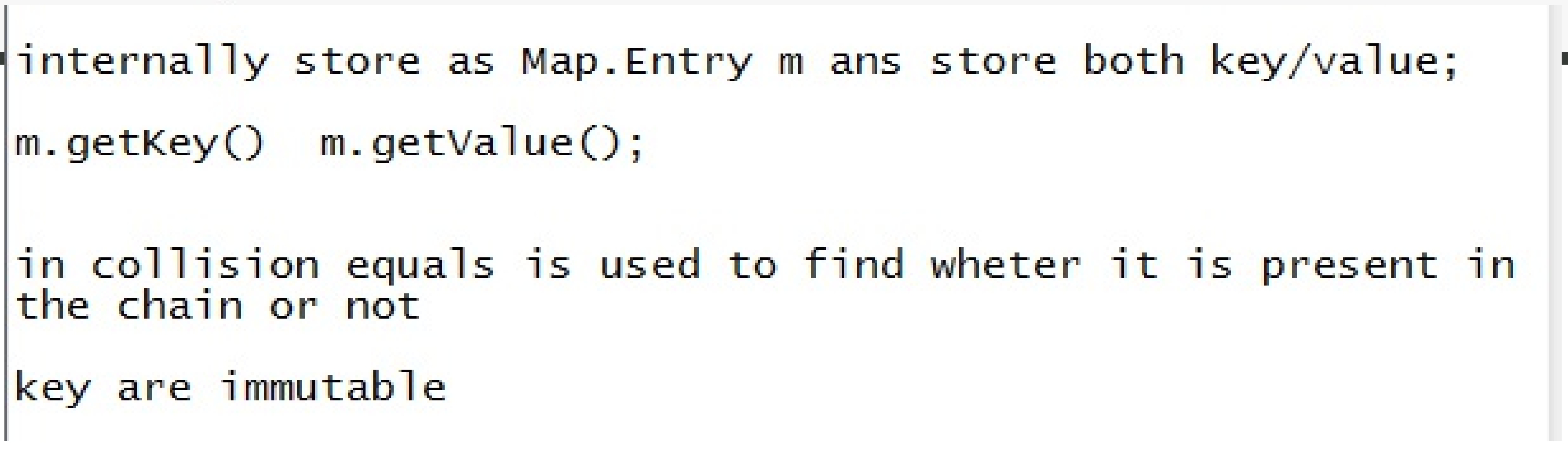
```
package mrbool.com;

import java.util.HashMap;
import java.util.Iterator;
import java.util.Map;
import java.util.Set;

public class HashMapDemo {
                public static void main(String[] args) {

                                HashMap<Integer, String> hm = new HashMap<Integer, String>();
                                hm.put(100, "sachin");
                                hm.put(101, "sehwag");
                                hm.put(102, "gambir");

                                Set set = hm.entrySet();
                                Iterator it = set.iterator();

                                while (it.hasNext()) {
                                                Map.Entry m = (Map.Entry) it.next();
                                                System.out.println(m.getKey() + ":" + m.getVal

                                }

                }
}
```



```
102 : gambir
100 : sachin
101 : sehwag
```



array, known as table. HashMap internally store mapping in form of Map.Entry object which contains both key and value object. When you want to retrieve the object, you call get()



internally store as Map.Entry m ans store both key/value;

m.getKey() m.getValue();

in collision equals is used to find wheter it is present in the chain or not

key are immutable

HashSet

A HashSet represents a set of elements. It is available with collection package and extends **AbstractSet** and implements the **Set** interface. The HashSet class is used to create a collection and store it in a hash table. It does not guarantee the order of elements. It does not allow the duplicate elements to be stored.

We can write HashSet as follows:

class HashSet<T>

Here, T represents the generic type constructor. It represents which types of elements are being stored into the HashSet.

We can create object as follows:

```
HashSet<String> hs=new HashSet<String> ();
```

HashSet contains the following constructors:

- **HashSet ():** It is default hash set.
- **HashSet (Collection c):** It initializes hash set by using elements of c.
- **HashSet (int capacity):** It creates HashSet with initial capacity (The default capacity is 16).
- **HashSet (int capacity, float fillRatio)**: it initializes capacity and fillRatio to grow the capacity after certain elements inserted.

Method	Description
void clear()	It removes all the elements from the HashSet.
Object clone()	It returns duplicate copy of HashSet instance.
boolean contains (object obj)	It returns true if the set contains the specified element.
boolean isEmpty()	It returns true if there are no elements in the hash set.
Object remove(object key)	It removes the element from the HashSet, if it is present. It returns true if element is removed successfully, otherwise false.
int size()	It returns the number of elements present in the HashSet.

```
import java.util.HashSet;
import java.util.Iterator;

public class HashsetDemo {
                public static void main(String[] args) {
                               HashSet<String> hs = new HashSet<String>();
                               hs.add("cricket");
                               hs.add("foorball");
                               hs.add("baseball");
                               hs.add("cricket");
                               System.out.println(hs);

                               System.out.println("Elements using iterator:")
                               Iterator it = hs.iterator();
                               while (it.hasNext()) {
                                               System.out.println(it.next());
                               }

                }

}
```

```
[baseball, cricket, foorball]

Elements using iterator:
baseball
cricket
foorball
```

LinkedList

LinkedList contains group of elements in the nodes. LinkedList extends **AbstractSequentialList** and implements the **List**, **Deque** and **Queue** interfaces.

Following are some of the features of LinkedList class:

- LinkedList is very convenient to store the data.
- It can contain duplicate elements.
- It maintains insertion order.
- It is not synchronized.
- It doesn't support random access for retrieving values.
- It can be used as list, stack or queue.

LinkedList has the following declaration:

class LinkedList<E>

We can create LinkedList for storing String type elements as follows:

```
LinkedList<String> ll=new LinkedList<String> () ;
```

Method	Description
void add(int position, element obj)	It inserts specified element at the specified position in the LinkedList.
boolean add(element obj)	It adds specified element to the end of the LinkedList.
boolean addAll(Collection c)	It appends all the elements of the collection to the end of the LinkedList.
boolean addAll(int index, Collection c)	It inserts all the elements of the collection into the LinkedList, starting at the specified position.
element remove(int position)	It removes specified element at the specified position in the LinkedList.
boolean remove(object obj)	It removes first occurrence of specified element obj from the LinkedList.
void clear()	It removes all the elements from the LinkedList.
boolean contains(Object o)	It returns true if LinkedList contains specified element .
void addFirst(Object o)	It inserts the element at the first position of the linked list.
void addLast(Object o)	It appends the specified element to the end of the linked list.
Object get(int index)	It returns the element at specified position in the LinkedList.

Object getFirst()	It returns the first element in the LinkedList.
Object getLast()	It returns the last element in the LinkedList.
int indexOf(Object o)	It returns the first occurrence of the specified element in the list or -1 if the element is not found in the list.
int lastIndexOf(Object o)	It returns the last occurrence of the specified element in the list or -1 if the element is not found in the list.
int size()	It returns the number of elements in the linked list.
ListIterator listiterator(int index)	It returns a list iterator of the elements in the list starting at specified position in the list.
Object remove(int index)	It removes the element in the list at the specified position.
Object removeFirst()	It removes the first element from the linked list.
Object removeLast()	It removes the last element from the linked list.
Object set(int index, Object element)	It replaces the element at the specified position in the list with the specified element .
Object[] toArray()	It coverts linked list into an array of Object class type. All the elements in the list are stored in correct order.
Object[] toArray(Object [] a)	It returns all the elements in the list are stored in correct order; The run type of returned array is that of the specified array.

```
package mrbool.com;

import java.util.Iterator;
import java.util.LinkedList;

public class LinkedListDemo {
                public static void main(String[] args) {
                                LinkedList<String> ll = new LinkedList<String>();
                                ll.add("India");
                                ll.add("America");
                                ll.add("China");
                                ll.add("India");

                                Iterator it = ll.iterator();
                                while (it.hasNext()) {
                                                System.out.println(it.next());
                                }
                }
}
```

```
India
America
China
India
```