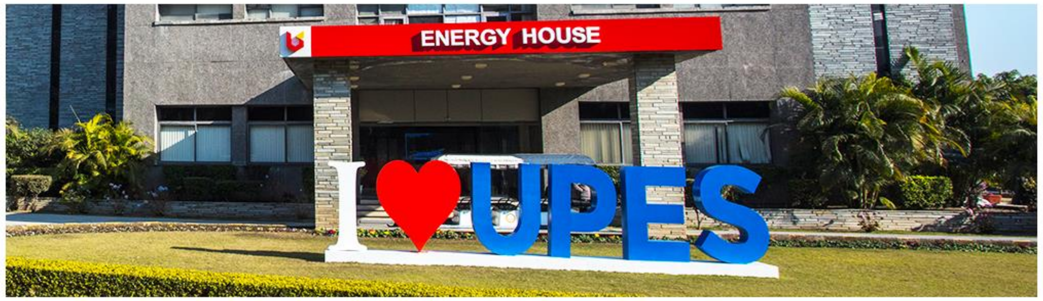


UPES

UNIVERSITY WITH A PURPOSE

Arrays Class

- Arrays class contains various methods for manipulating arrays (such as sorting and searching).
- This class also contains a static factory that allows arrays to be viewed as lists.
- The methods in this class all throw a `NullPointerException`, if the specified array reference is null, except where noted.

Methods of Arrays Class

1. public static int binarySearch(char[] a,int fromIndex,int toIndex, char key)
2. Public static int[] copyof(int[] original, int newLength)
3. static boolean equals(int[] a, int[] a2)
4. static void   fill(int[] a, int val)
5. Static int hashCode(int[] a)
6. static void   sort(int[] a)
7. static String toString(int[] a)

Source: https://docs.oracle.com/javase/7/docs/api/java/util/Arrays.html

Example: Arrays class

```
// Demonstrate Arrays
import java.util.*;
class ArraysDemo {
public static void main(String args[]) {
// Allocate and initialize array.
int array[] = new int[10];
for(int i = 0; i < 10; i++)
array[i] = -3 * i;
// Display, sort, and display the array.
System.out.print("Original contents:");
display(array);
Arrays.sort(array);
System.out.print("Sorted: ");
display(array);
// Fill and display the array.
Arrays.fill(array, 2, 6, -1);
System.out.print("After fill(): ");
display(array);
// Sort and display the array.
Arrays.sort(array);
System.out.print("After sorting again: ");
display(array);
// Binary search for -9.
System.out.print("The value -9 is at location ");
int index =
Arrays.binarySearch(array, -9);
System.out.println(index);
}
static void display(int array[]) {
for(int i: array)
System.out.print(i + " ");
System.out.println();
}
}
```

Legacy Collection

- Collection framework was not the part of earlier versions of Java. However, certain classes and interfaces were available for similar purposes.
- Collections framework introduced J2SE 1.2, and then the previous original classes and interfaces were reengineered to support the collection interface.
- These classes are known as Legacy classes. All legacy classes and interfaces were redesigned by JDK 5 to support Generics.
- The legacy classes defined by **java.util** are shown below:

Dictionary	Hashtable	Properties	Stack	Vector

- There is one legacy interface called **Enumeration.**

Legacy Collection (cont..)

Enumeration Interface:

`public interface Enumeration<E>` → An object that implements the Enumeration interface generates a series of elements, one at a time. Successive calls to the `nextElement` method return successive elements of the series.

Method Detail:

`Boolean hasMoreElements()`→ Tests if this enumeration contains more elements.

`E nextElement()`→ Returns the next element of this enumeration if this enumeration object has at least one more element to provide.

Legacy Collection (Vector)

Vector class:

- The Vector class implements a growable array of objects similar to ArrayList.
- Like an array, it contains components that can be accessed using an integer index. However, the size of a Vector can grow or shrink as needed to accommodate adding and removing items after the Vector has been created.

Methods Details: For other methods visit Ref.

Method	Description
`void addElement(E element)`	adds element to the Vector
`E elementAt(int index)`	returns the element at specified index
`Enumeration elements()`	returns an enumeration of element in vector
`int capacity ()`	Returns the current capacity of this vector.
`public void ensureCapacity( int minCapacity)`	Increases the capacity of this vector, if necessary
`void removeAllElements()`	removes all elements of the Vector

UPES
UNIVERSITY WITH A PURPOSE

Legacy Collection (cont..)

Exp: // To Demonstrate various Vector operations.

```
import java.util.*;
class VectorDemo {
public static void main(String args[]) {
// initial size is 3, increment is 2
Vector<Integer> v = new Vector<Integer>(3, 2);
System.out.println("Initial size: " + v.size());
System.out.println("Initial capacity: " +
v.capacity());
v.addElement(1); v.addElement(2);
v.addElement(3); v.addElement(4);
System.out.println("Capacity after four
additions: " + v.capacity());
v.addElement(5);
System.out.println("Current capacity: " +
v.capacity());
v.addElement(6); v.addElement(7);
System.out.println("Current capacity: " +
v.capacity());
v.addElement(9); v.addElement(10);
System.out.println("Current capacity: " +
v.capacity());
v.addElement(11); v.addElement(12);
System.out.println("First element: " +
v.firstElement());
System.out.println("Last element: " +
v.lastElement());
if(v.contains(3))
System.out.println("Vector contains 3.");
// Enumerate the elements in the vector.
Enumeration<Integer> vEnum =
v.elements();
System.out.println("\nElements in
vector:"); while(vEnum.hasMoreElements())
System.out.print(vEnum.nextElement() + "
");
System.out.println();
}
}
```

Legacy Collection (Stack)

Stack class:

- The Stack class represents a last-in-first-out (LIFO) stack of objects. It extends class Vector with five operations that allow a vector to be treated as a stack.
- When a stack is first created, it contains no items.

Methods Details:

Method	Description
`public E push(E item)`	Pushes an item onto the top of this stack.
`public E pop()`	Removes the object at the top of this stack and returns that object as the value of this function.
`public E peek()`	Looks at the object at the top of this stack without removing it from the stack.
`public boolean empty()`	Tests if this stack is empty.
`public int search(Object o)`	Returns the 1-based position where an object is on this stack.

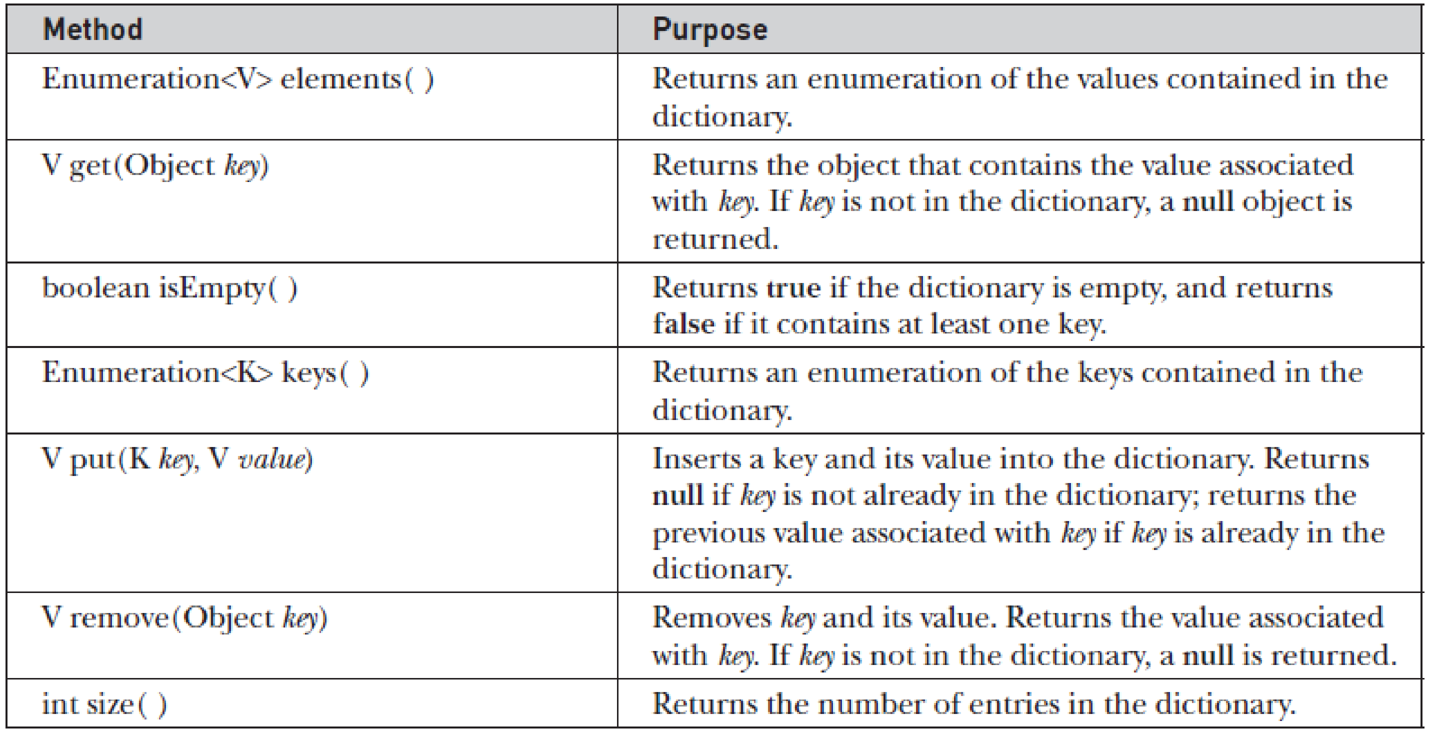
Legacy Collection (cont..)

Exp: // To Demonstrate Stack class operations.

```
import java.util.*;
class StackDemo {
static void showpush(Stack<Integer> st, int
a) {
st.push(a);
System.out.println("push(" + a + ")");
System.out.println("stack: " + st);
}
static void showpop(Stack<Integer> st) {
System.out.print("pop -> ");
Integer a = st.pop();
System.out.println(a);
System.out.println("stack: " + st);
}
public static void main(String
args[]) {
Stack<Integer> st = new
Stack<Integer>();
System.out.println("stack: " + st);
showpush(st, 42);
showpush(st, 66);
showpush(st, 99);
showpop(st);
showpop(st);
showpop(st);
try {
showpop(st);
} catch (EmptyStackException e) {
System.out.println("empty stack");
}
}
}
```

Legacy Collection (Dictionary)

- The Dictionary class is the abstract parent of any class, such as Hashtable, which maps keys to values.
- Every key and every value is an object. In any one Dictionary object, every key is associated with at most one value.
- Given a Dictionary and a key, the associated element can be looked up. Any non-null object can be used as a key and as a value.



Method	Purpose
Enumeration<V> elements( )	Returns an enumeration of the values contained in the dictionary.
V get(Object *key*)	Returns the object that contains the value associated with *key*. If *key* is not in the dictionary, a null object is returned.
boolean isEmpty( )	Returns **true** if the dictionary is empty, and returns **false** if it contains at least one key.
Enumeration<K> keys( )	Returns an enumeration of the keys contained in the dictionary.
V put(K *key*, V *value*)	Inserts a key and its value into the dictionary. Returns **null** if *key* is not already in the dictionary; returns the previous value associated with *key* if *key* is already in the dictionary.
V remove(Object *key*)	Removes *key* and its value. Returns the value associated with *key*. If *key* is not in the dictionary, a **null** is returned.
int size( )	Returns the number of entries in the dictionary.

Legacy Collection (Hashtable)

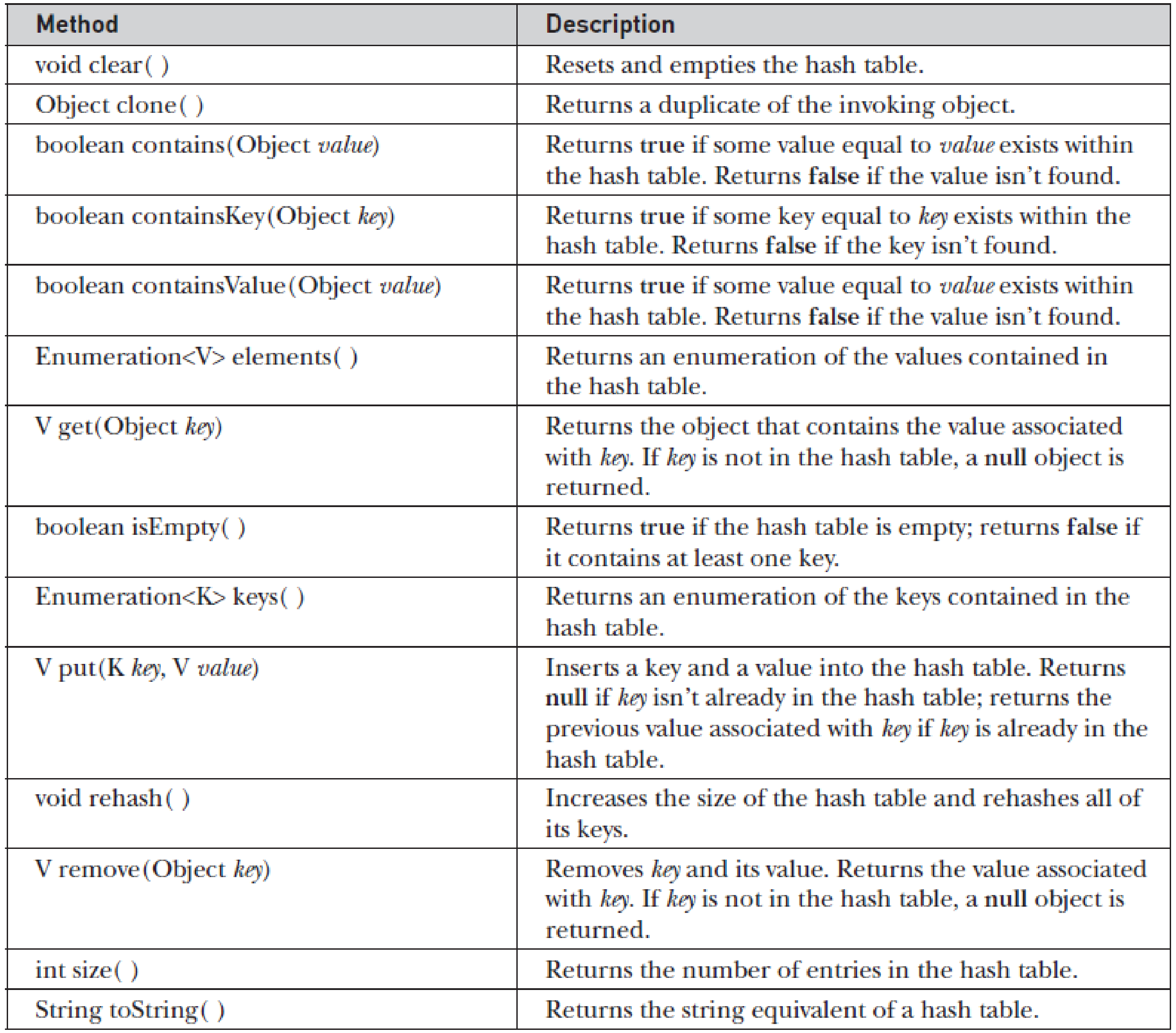
- This class implements a hash table, which maps keys to values. Any non-null object can be used as a key or as a value.
- To successfully store and retrieve objects from a hashtable, the objects used as keys must implement the `hashCode` method and the `equals` method.

Constructors:

`Hashtable()`→ Constructs a new, empty hashtable with a default initial capacity (11) and load factor (0.75).

`Hashtable(int initialCapacity)`→ Constructs a new, empty hashtable with the specified initial capacity and default load factor (0.75).

Legacy Collection (Hashtable)



Method	Description
void clear( )	Resets and empties the hash table.
Object clone( )	Returns a duplicate of the invoking object.
boolean contains(Object *value*)	Returns **true** if some value equal to *value* exists within the hash table. Returns **false** if the value isn't found.
boolean containsKey(Object *key*)	Returns **true** if some key equal to *key* exists within the hash table. Returns **false** if the key isn't found.
boolean containsValue(Object *value*)	Returns **true** if some value equal to *value* exists within the hash table. Returns **false** if the value isn't found.
Enumeration<V> elements( )	Returns an enumeration of the values contained in the hash table.
V get(Object *key*)	Returns the object that contains the value associated with *key*. If *key* is not in the hash table, a **null** object is returned.
boolean isEmpty( )	Returns **true** if the hash table is empty; returns **false** if it contains at least one key.
Enumeration<K> keys( )	Returns an enumeration of the keys contained in the hash table.
V put(K *key*, V *value*)	Inserts a key and a value into the hash table. Returns **null** if *key* isn't already in the hash table; returns the previous value associated with *key* if *key* is already in the hash table.
void rehash( )	Increases the size of the hash table and rehashes all of its keys.
V remove(Object *key*)	Removes *key* and its value. Returns the value associated with *key*. If *key* is not in the hash table, a **null** object is returned.
int size( )	Returns the number of entries in the hash table.
String toString( )	Returns the string equivalent of a hash table.

Legacy Collection (Hashtable)

Exp: // To Demonstrate Hashtable class operations.

```
import java.util.*;
class HTDemo {
public static void main(String args[]) {
Hashtable<String, Double> balance =
new Hashtable<String, Double>();
Enumeration<String> names;
String str; double bal;
balance.put("John Doe", 3434.34);
balance.put("Tom Smith", 123.22);
balance.put("Jane Baker", 1378.00);
balance.put("Tod Hall", 99.22);
balance.put("Ralph Smith", -19.08);
// Show all balances in hashtable.
names = balance.keys();
while(names.hasMoreElements()) {
str = names.nextElement();
System.out.println(str + ": " +
balance.get(str)); }
System.out.println();
// Deposit 1,000 into John Doe's account.
bal = balance.get("John Doe");
balance.put("John Doe", bal+1000);
System.out.println("John Doe's new
balance: " +
balance.get("John Doe"));
}
}
```

Legacy Collection (Properties)

- ➢ The Properties class represents a persistent set of properties. The Properties can be saved to a stream or loaded from a stream. Each key and its corresponding value in the property list is a string.
- ➢ Because Properties inherits from `Hashtable`, the `put` and `putAll` methods can be applied to a Properties object.

Constructors:

`Properties()`→ Creates an empty property list with no default values.

`Properties(Properties defaults)`→ Creates an empty property list with the specified defaults.

Legacy Collection (Properties)

Method	Description
`String getProperty(String key)`	Returns the value associated with the key.
`String getProperty(String key, String defaultProperty)`	Returns the value associated with the key; defaultProperty
`void list(PrintStream streamOut)`	Sends the property list to the output stream linked to streamOut.
`void list(PrintWriter streamOut)`	Sends the property list to the output stream linked to streamOut.
`void load(InputStream streamIn) throws IOException`	Inputs a property list from the input stream linked to streamIn.
`Enumeration propertyNames( )`	Returns an enumeration of the keys.
`Object setProperty(String key, String value)`	Associates value with the key.
`void store(OutputStream streamOut, String description)`	After writing the string specified by description, the property list is written to the output stream linked to streamOut.

UPES
UNIVERSITY WITH A PURPOSE

References

- https://docs.oracle.com/javase/8/docs/technotes/guides/collections/overview.
- htmlhttps://www.studytonight.com/java/legacy-classes-and-interface.php

UPES

UNIVERSITY WITH A PURPOSE

THANK YOU

UPES

UNIVERSITY WITH A PURPOSE