**Assignment 4**

What is an array?

Array is a collection of similar types of elements stored in contiguous memory locations.The elements in the array can be accessed randomly by their index numbers.

Types:

One Dimension

Multi Dimensional

//Java Program to illustrate how to declare, instantiate, initialize

//and traverse the Java array.

class Testarray{

public static void main(String args[]){

int a[]=new int[5];//declaration and instantiation

a[0]=10;//initialization

a[1]=20;

a[2]=70;

a[3]=40;

a[4]=50;

//traversing array

for(int i=0;i<a.length;i++)//length is the property of array

System.out.println(a[i]);

}}

Multidimensional Array:

//Java Program to illustrate the use of multidimensional array

class Testarray3{

public static void main(String args[]){

//declaring and initializing 2D array

int arr[][]={{1,2,3},{2,4,5},{4,4,5}};

//printing 2D array

for(int i=0;i<3;i++){

for(int j=0;j<3;j++){

System.out.print(arr[i][j]+" ");

}

System.out.println();

}

}}

Enhanced For Loops:

The Enhanced For Loop Has The Following Characteristics:

* Simplified Iteration Over Collections
* Much Shorter, Clearer, And Safer
* Effective For Arrays
* Simpler When Using Nested Loops
* Iterator Disadvantages Removed

The Foreach Loops:

public class TestArray {

public static void main(String[] args) {

double[] myList = {1.9, 2.9, 3.4, 3.5};

// Print all the array elements

for (double element: myList) {

System.out.println(element);

}

}

}

Array Resizing:

* You cannot resize an array.
* You can use the same reference variable to refer to

an entirely new array, such as:

1. int[] myArray = new int[6];
2. myArray = new int[10];

Copying Arrays:

//original array

int[] myArray = { 1, 2, 3, 4, 5, 6 };

// new larger array

int[] hold = { 10, 9, 8, 7, 6, 5, 4, 3, 2, 1 };

// copy all of the myArray array to the hold

// array, starting with the 0th index

System.arraycopy(myArray, 0, hold, 0,

myArray.length);

public static void arraycopy(Object src, int srcPos, Object

dest, int destPos, int length)

Passing Arrays To Methods:

public static void printArray(int[] array) {

for (int i = 0; i < array.length; i++) {

System.out.print(array[i] + " ");

}

}

Returning An Array From A Method:

public static int[] reverse(int[] list) {

int[] result = new int[list.length];

for (int i = 0, j = result.length - 1; i < list.length;

i++, j--) {

result[j] = list[i];

}

return result;

}

What is collection?

A collection is a single object managing a group of

objects known as its elements.

Set – An unordered collection; no duplicates are

permitted.

List – An ordered collection; duplicates are

permitted.

Set Example:

import java.util.\*;

public class SetExample {

public static void main(String[] args) {

Set set = new HashSet();

set.add("one");

set.add("second");

set.add("3rd");

set.add(new Integer(4));

set.add(new Float(5.0F));

set.add("second"); // duplicate, not added

set.add(new Integer(4)); // duplicate, not added

System.out.println(set);

}

}

List Example:

import java.util.\*

public class ListExample {

public static void main(String[] args) {

List list = new ArrayList();

list.add("one");

list.add("second");

list.add("3rd");

list.add(new Integer(4));

list.add(new Float(5.0F));

list.add("second"); // duplicate, is added

list.add(new Integer(4)); // duplicate, is added

System.out.println(list);

}

}

Map:

• A Map object describes mappings from keys to

values:

• Duplicate keys are not allowed

• One-to-many mappings from keys to values is not

permitted

• The contents of the Map interface can be viewed

and manipulated as collections

• entrySet – Returns a Set of all the key-value pairs.

• keySet – Returns a Set of all the keys in the map.

• values – Returns a Collection of all values in the

map.

import java.util.\*;

public class MapExample {

public static void main(String args[]) {

Map map = new HashMap();

map.put("one","1st");

map.put("second", new Integer(2));

map.put("third","3rd");

// Overwrites the previous assignment

map.put("third“,"III");

// Returns set view of keys

Set set1 = map.keySet();

// Returns Collection view of values

Collection collection = map.values();

// Returns set view of key value mappings

Set set2 = map.entrySet();

System.out.println(set1 + "\n" + collection + "\n" + set2);

}

}

Classic Collections:

• The Vector class, which implements the List

interface.

• The Stack class, which is a subclass of the Vector class

and supports the push, pop, and peek methods.

• The Hashtable class, which implements the Map

interface.

• The Properties class is an extension of Hashtable

that only uses Strings for keys and values.

• Each of these collections has an elements method that

returns an Enumeration object.

Iterators:

* Iteration is the process of retrieving every element in a collection.
* The basic Iterator interface allows you to scan forward through any collection.