# PRACTICAL 5

**Aim**: To understand Array constructs and Vector ClassinJava programming Language

**Prerequisite:**

* Knowledge of basic constructs in Java programming language.

**Outcome:** After successful completion of this experiment students will be able to,

* Declare, create and initialize arrays
* Understand the usage of Array as data structure in different programming tasks.
* Use various utility functions predefined in the Array class
* Understand and use Vector class and various methods to manipulate the vector elements

**Theory:**

**Declare, create and initialize Arrays:**

An array is a group of like-typed variables that are referred to by a common name.Arrays in Java work differently than they do in C/C++. Following are some important point about Java arrays.

* In Java all arrays are dynamically allocated.(discussed below)
* Since arrays are objects in Java, we can find their length using member length. This is different from C/C++ where we find length using sizeof.
* A Java array variable can also be declared like other variables with [] after the data type.
* The variables in the array are ordered and each have an index beginning from 0.
* Java array can be also be used as a static field, a local variable or a method parameter.
* The size of an array must be specified by an int value and not long or short.

**One-Dimensional Arrays:**   
The general form of a one-dimensional array declaration is

type var-name[];

OR

type[] var-name;

An array declaration has two components: the type and the name. *type* declares the element type of the array.

When an array is declared, only a reference of array is created. To actually create or give memory to array, you create an array like this:The general form of new as it applies to one-dimensional arrays appears as follows:

var-name = new type [size];

Here, type specifies the type of data being allocated, size specifies the number of elements in the array, and var-name is the name of array variable that is linked to the array. That is, to use new to allocate an array, you must specify the type and number of elements to allocate.

In a situation, where the size of the array and variables of array are already known, array literals can be used.

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

**Multi-Dimensional Arrays:**

Multidimensional arrays are arrays of arrays with each element of the array holding the reference of other array. These are also known as [Jagged Arrays](https://www.geeksforgeeks.org/jagged-array-in-java/). A multidimensional array is created by appending one set of square brackets ([]) per dimension.

int[][] intArray = new int[10][20]; //a 2D array or matrix

int[][][] intArray = new int[10][20][10]; //a 3D array

**Few Overloaded functions in Array class:**

**For Sorting:**

double [] numbers = {6.0, 4.4, 1.9, 2.9, 3.4, 3.5};

java.util.Arrays.sort(numbers);

 char [] chars = {'a', 'A', '4', 'F', 'D', 'P'};

java.util.Arrays.sort(chars);

**For Searching:**

int[] list = {2, 4, 7, 10, 11, 45, 50, 59, 60, 66, 69, 70, 79};

System.out.println("Index is " + java.util.Arrays.binarySearch(list, 11));

 char [] chars = {'a', 'c', 'g', 'x', 'y', 'z'};

System.out.println("Index is " + java.util.Arrays.binarySearch(chars, 't'));

**Checking for equality:**

static booleanequals (char [] a1, char [] a2)

**Copying one array to the other:**

public static char [] copyOf(char [] original, intnewLength)

**Vectors:**

Vector is a data structure that is used to store a collection of elements. Elements can be of all primitive types like int, float, Object, etc. Vectors are dynamic in nature and accordingly, grow or shrink as per the requirement.

Vector Class in Java is found in the java.util package.Vector class is a child class of the AbstractList class and implements the List interface. Therefore, we can use all the methods of the List interface.When a Vector is created, it has a certain capacity to store elements that can be defined initially. This capacity is dynamic in nature and can be increased or decreased.By definition, Vectors are synchronized, which implies that at a time, only one thread is able to access the code while other threads have to wait. Due to this, Vectors are slower in performance as they acquire a lock on a thread.

**Operations in Java Vectors**

The most common operations include:

* Adding elements to the vector.
* Iterating over the elements we added in the vector.
* Replacing elements at certain indexes in the vector.
* Removing elements from the vector.

**Few Vector Methods in Java**

|  |  |  |
| --- | --- | --- |
| **SN** | **Method** | **Description** |
| 1) | [add()](https://www.javatpoint.com/java-vector-add-method) | It is used to append the specified element in the given vector. |
| 2) | [addAll()](https://www.javatpoint.com/java-vector-addall-method) | It is used to append all of the elements in the specified collection to the end of this Vector. |
| 3) | [addElement()](https://www.javatpoint.com/java-vector-addelement-method) | It is used to append the specified component to the end of this vector. It increases the vector size by one. |
| 4) | [capacity()](https://www.javatpoint.com/java-vector-capacity-method) | It is used to get the current capacity of this vector. |
| 5) | [clear()](https://www.javatpoint.com/java-vector-clear-method) | It is used to delete all of the elements from this vector. |
| 6) | [clone()](https://www.javatpoint.com/java-vector-clone-method) | It returns a clone of this vector. |
| 7) | [contains()](https://www.javatpoint.com/java-vector-contains-method) | It returns true if the vector contains the specified element. |
| 8) | [containsAll()](https://www.javatpoint.com/java-vector-containsall-method) | It returns true if the vector contains all of the elements in the specified collection. |
| 9) | [copyInto()](https://www.javatpoint.com/java-vector-copyinto-method) | It is used to copy the components of the vector into the specified array. |
| 10) | [elementAt()](https://www.javatpoint.com/java-vector-elementat-method) | It is used to get the component at the specified index. |
| 11) | [elements()](https://www.javatpoint.com/java-vector-elements-method) | It returns an enumeration of the components of a vector. |
| 12) | [ensureCapacity()](https://www.javatpoint.com/java-vector-ensurecapacity-method) | It is used to increase the capacity of the vector which is in use, if necessary. It ensures that the vector can hold at least the number of components specified by the minimum capacity argument. |
| 13) | [equals()](https://www.javatpoint.com/java-vector-equals-method) | It is used to compare the specified object with the vector for equality. |
| 14) | [firstElement()](https://www.javatpoint.com/java-vector-firstelement-method) | It is used to get the first component of the vector. |
| 15) | [forEach()](https://www.javatpoint.com/java-vector-foreach-method) | It is used to perform the given action for each element of the Iterable until all elements have been processed or the action throws an exception. |
| 16) | [get()](https://www.javatpoint.com/java-vector-get-method) | It is used to get an element at the specified position in the vector. |
| 17) | [hashCode()](https://www.javatpoint.com/java-vector-hashcode-method) | It is used to get the hash code value of a vector. |
| 18) | [indexOf()](https://www.javatpoint.com/java-vector-indexof-method) | It is used to get the index of the first occurrence of the specified element in the vector. It returns -1 if the vector does not contain the element. |
| 19) | [insertElementAt()](https://www.javatpoint.com/java-vector-insertelementat-method) | It is used to insert the specified object as a component in the given vector at the specified index. |
| 20) | [isEmpty()](https://www.javatpoint.com/java-vector-isempty-method) | It is used to check if this vector has no components. |
| 21) | [iterator()](https://www.javatpoint.com/java-vector-iterator-method) | It is used to get an iterator over the elements in the list in proper sequence. |
| 22) | [lastElement()](https://www.javatpoint.com/java-vector-lastelement-method) | It is used to get the last component of the vector. |
| 23) | [lastIndexOf()](https://www.javatpoint.com/java-vector-lastindexof-method) | It is used to get the index of the last occurrence of the specified element in the vector. It returns -1 if the vector does not contain the element. |
| 25) | [remove()](https://www.javatpoint.com/java-vector-remove-method) | It is used to remove the specified element from the vector. If the vector does not contain the element, it is unchanged. |
| 26) | [removeAll()](https://www.javatpoint.com/java-vector-removeall-method) | It is used to delete all the elements from the vector that are present in the specified collection. |
| 27) | [removeAllElements()](https://www.javatpoint.com/java-vector-removeallelements-method) | It is used to remove all elements from the vector and set the size of the vector to zero. |
| 28) | [removeElement()](https://www.javatpoint.com/java-vector-removeelement-method) | It is used to remove the first (lowest-indexed) occurrence of the argument from the vector. |
| 29) | [removeElementAt()](https://www.javatpoint.com/java-vector-removeelementat-method) | It is used to delete the component at the specified index. |

(TO BE COMPLETED BY STUDENTS)

|  |  |
| --- | --- |
| Roll No. K005 | Name: Jal Bafana |
| Class: Btech. Cyber Security (Sem-4) | Batch: K1 |
| Date of Practical: 24.02.2025 | Date of Submission: 24.02.2025 |
| Grade: |  |

1. Write a program (with series of functions) that reads student scores, gets the best score, and then assigns grades based on the following scheme:

Grade is B if score is >= best - 20;

Grade is C if score is >= best - 30;

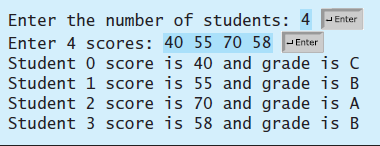
Grade is D if score is >= best - 40;

Grade is F otherwise.

The program prompts the user to enter the total number of students, then prompts

the user to enter all of the scores, and concludes by displaying the grades. Here is

a sample run:



import java.util.Scanner;

public class StudentGrades

{

public static void main(String[] args)

{

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the number of students: ");

int numStudents = scanner.nextInt();

int[] scores = new int[numStudents];

System.out.print("Enter " + numStudents + " scores: ");

int bestScore = 0;

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

{

scores[i] = scanner.nextInt();

if (scores[i] > bestScore) {

bestScore = scores[i];

}

}

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

{

char grade = getGrade(scores[i], bestScore);

System.out.println("Student " + i + " score is " + scores[i] + " and grade is " + grade);

}

scanner.close();

}

public static char getGrade(int score, int best)

{

if (score >= best - 10) return 'A';

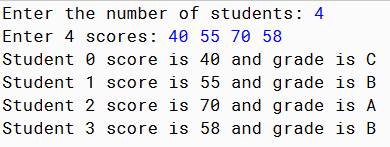
else if (score >= best - 20) return 'B';

else if (score >= best - 30) return 'C';

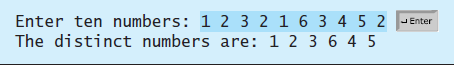
else return 'D';

}

}



1. Write a program (with series of functions) that reads in ten numbers and displays distinct numbers (i.e., if a number appears multiple times, it is displayed only once). *Hint*: Read a number and store it to an array if it is new. If the number is already in the array, ignore it. After the input, the array contains the distinct numbers. Here is the sample run of the program:



import java.util.Scanner;

import java.util.ArrayList;

public class DistinctNumbers

{

public static void main(String[] args)

{

Scanner scanner = new Scanner(System.in);

ArrayList<Integer> distinctNumbers = new ArrayList<>();

System.out.println("Enter 10 numbers: ");

for (int i = 0; i < 10; i++)

{

int num = scanner.nextInt();

if (!distinctNumbers.contains(num)) {

distinctNumbers.add(num);

}

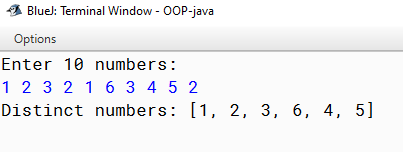
}

System.out.println("Distinct numbers: " + distinctNumbers);

scanner.close();

}

}



1. Write a program to create a Vector that holds the following values.

2, 6, Hello, 7, everyone

Use appropriate methods of Vector class to perform the following task.

1. Iterate through the vector
2. Add element at a specified index
3. Check whether a given element exist in the Vector, if so remove it
4. Replace an item with a new value given
5. Retrieve the first and the last element of the Vector

import java.util.Vector;

public class EgVect

{

public static void main(String[] args)

{

Vector<Object> vector = new Vector<>();

// adding elements to vector

vector.add(2);

vector.add(6);

vector.add("Hello");

vector.add(7);

vector.add("everyone");

// a. iterating

System.out.println("Vector elements:");

for (Object element : vector) {

System.out.print(element + " ");

}

System.out.println();

// b. adding to a specific index

vector.add(2, "NewElement");

System.out.println("After adding at index 2: " + vector);

// c. checking and removing

if (vector.contains("Hello")) {

vector.remove("Hello");

}

System.out.println("After removing 'Hello': " + vector);

// d. replacing with a new value

vector.set(1, 99);

System.out.println("After replacing element at index 1 with 99: " + vector);

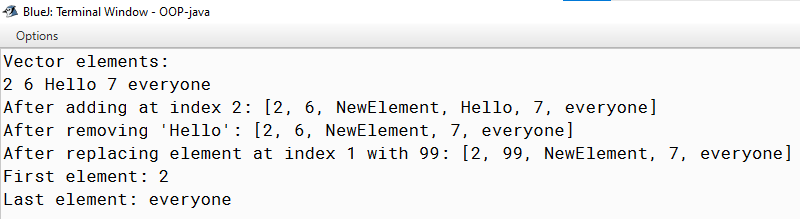
// e. first and last elements

System.out.println("First element: " + vector.firstElement());

System.out.println("Last element: " + vector.lastElement());

}

}



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*