UNIT 6: ARRAYS, STRINGS AND VECTORS

UNIT STRUCTURE

6.1 Learning Objectives

6.2 Introduction

6.3 Declaring Arrays

6.4 Creating Arrays

6.5 Initializing Arrays

6.6 Multi-Dimensional Arrays

- 6.6.1 Two-Dimensional Array Declaration
- 6.6.2 Two-Dimensional Array Creation
- 6.6.3 Two-Dimensional Array Initialization

6.7 Arrays Class

6.8 Strings

- 6.8.1 String Class
- 6.8.2 String Arrays
- 6.8.3 String Methods
- 6.8.4 StringBuffer Class

6.9 Vectors

6.10 Let Us Sum Up

6.11 Further Reading

6.12 Answer to Check Your Progress

6.13 Model Questions

6.1 LEARNING OBJECTIVES

After going through this unit, you will be able to:

- learn about declaration, creation and initialization of arrays in Java
- learn about multi-dimensional arrays in Java
- use strings in Java
- learn about vectors in Java

6.2 INTRODUCTION

In the previous unit, we have discussed the implementation of class and declaration of object in Java. Concept of constructor is discussed in this section. We have learned how to pass arguments to methods. Finally, implementation of inheritance in Java is also discussed in this unit.

We already know that an array is a data structure containing similar type of data where data are stored in contiguous memory locations. Each data available in an array is termed as element of the array. In this unit, we are going to discuss about the implementation of arrays and strings in Java. We will also learn about vectors in Java.

6.3 DECLARING ARRAYS

In Java programming, arrays are container objects that store data of same types. For example, an integer array can store only integer values. In Java, arrays are dynamically allocated and the size of an array is fixed at the time of its creation. Each element of an array is accessed by an integer value termed as subscript. Subscript is used within bracket after the name of the array. In Java, subscript of an array is started with 0.

The first step to work with array is the declaration of arrays. In this section, we will learn about one-dimensional arrays. Arrays with only one subscript are termed as one-dimensional arrays. The syntax of one-dimensional array declaration in Java is presented as follows;

Data_Type [ ] Array_Name ;

Or

Data_Type Array_Name [ ] ;

In the above syntax, Data_Type is any valid Java data type and Array_Name is the name of the declared array. For example, let us consider the following array declaration statement in Java.

```
int[ ] Arr ;
```

The above statement will declare an integer array where the name of the array is Arr.

6.4 CREATING ARRAYS

In the previous section, we have learnt to declare a one dimensional array in Java. Now in the next step, we have to create arrays. In Java, arrays are created in memory by using 'new' operator. The syntax of using 'new' operator to create a one dimensional array is presented as follows.

Array_Name = new Data_Type [Array_Size] ;

In the above syntax, Array_Size is the value that denotes the maximum number of data which can be stored in the array. Array_Size can be a constant or an integer variable. For example, let us consider the following Java statement.

```
Arr = new int [30] ;
```

The above statement will create an integer array, Arr that can store at most 30 integer numbers.

6.5 INITIALIZING ARRAYS

After creation of an array in a program, values can be inserted into the array depending upon the requirement of the program. In Java, the syntax to insert a value into a one dimensional array is presented as follows.

Array_Name[SubscriptValue] = Data;

The above statement will insert the value, 'Data' into the array, 'Array_Name' at the subscript, 'SubscriptValue'.

The following Java program will demonstrate the insertion of values in a one dimensional array.

Example: Java program to insert integer values in a one dimensional array.

```
import  java.util.*;
class insertData
{
      public static void main(String []args)
      {
         int Arr[ ];  // 1-D array declaration
         System.out.println("Enter array size:");
```

```
Scanner S = new Scanner(System.in);
int arrSize = S.nextInt();
//One-dimensional array creation in memory
 Arr = new int[arrSize];
// Insertion of integer values into the array
System.out.println("Enter array elements:");
for(int i=0;i<arrSize;i++)
{
 System.out.println("Enter "+(i+1)+"th element:");
 int data = S.nextInt();
 Arr[i] = data;
}
System.out.println("Array elements are:");
//Display array elements
for(int i = 0;i<arrSize; i++)
{
    System.out.print("  "+Arr[i]);
}
S.close();
}
}
```

Output of the program:

```
G:\java program>javac insertData.java

G:\java program>java insertData
Enter array size:
5
Enter array elements:
Enter 1th element:
12
Enter 2th element:
5
Enter 3th element:
8
Enter 4th element:
23
Enter 5th element:
87
Array elements are:
  12  5  8  23  87
```

In Java, creation and initialization of a one dimensional array can be performed at the time of its declaration. The syntax for this operation in Java is presented as follows.

Data_Type [ ] Array_Name={ Data1, Data2 , Data3 , Data4,……..};

Or

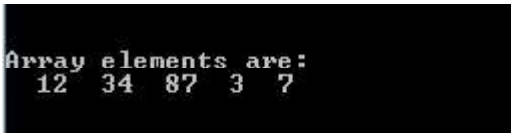
Data_Type Array_Name [ ]={ Data1, Data2, Data3,Data4,……..};

The following Java program will demonstrate the creation and initialization of a one dimensional array at the time of its declaration.

Example: Java program to initialize a one-dimensional array.

```
class initOneDimensionalArray
{
  public static void main(String []args)
   {
      int Arr[ ] = {12,34,87,3,7};
      int N = Arr.length;
     System.out.println("Array elements are:");
     //Display array elements
     for(int i=0;i<N;i++)
       {
        System.out.print("  "+Arr[i]);
       }
    }
}
```

Output of the program:



In the above program, we have used a variable to obtain the size of an array. This variable is 'length'. It is a public final member of array type.

Example: Java program to sort the elements in a one-dimensional integer array using Selection sort algorithm

```
class sortOneDimensionalArray
{
   public static void main(String []args)
   {
     // One-dimensional array initialization
    int Arr[]={5,19,12,34,87,3,7,89,1,33,32,11,};
    int min,i,j,index;
    int N = Arr.length;
    System.out.println("\n\n");
    System.out.println("\n Before sorting
    array elements are:");
     //Display array elements
     for(i=0;i<N;i++)
     {
           System.out.print("  "+Arr[i]);
     }
     // Sorting using Selection sort algorithm
     for(i=0;i<N-1;i++)
     {
         min = Arr[i];
         index = i;
         for(j=i+1;j<N;j++)
         {
             if(Arr[j]<min)
             {
                 min = Arr[j];
                 index=j;
             }
          }
          if(index!=i)
          {
             Arr[index] = Arr[i];
```

```
            Arr[i] = min;
           }
         }
System.out.println("\n After sorting array
elements are:");
 //Display array elements
   for(i = 0;i<N;i++)
   {
       System.out.print("  "+Arr[i]);
   }
    System.out.print("\n\n");
  }
}
```

Output of the program:

```
Before sorting array elements are:
 5  19  12  34  87  3  7  89  1  33  32  11
After sorting array elements are:
 1  3  5  7  11  12  19  32  33  34  87  89
```

6.6 MULTI-DIMENSIONAL ARRAYS

In the previous sections, we have learnt the use of one dimensional arrays in Java. So we already know that, in case of a one dimensional array, only one subscript is available. Now if we declare an array with multiple subscripts than it is termed as multi-dimensional array. So an N-dimensional array possesses N number of subscripts. In Java, a multi-dimensional array can also be referred as arrays of arrays.

6.6.1 Two-Dimensional Array Declaration

Two-dimensional array is an example of multi-dimensional arrays where two subscripts are required to access an array element. Two-dimensional arrays are used to store tables of data. In a two-dimensional array, the first subscript is used to refer the row number and second subscript is used to refer the column number of a table

of data. The syntax to declare a two-dimensional array in Java is presented as follows.

Data_Type [ ][ ] Array_Name ;

Or

Data_Type Array_Name [ ][ ] ;

For example, let us consider the following array declaration statement in Java.

```
int  [ ][ ] ArrTwo ;
```

The above statement will declare a two-dimensional array, 'ArrTwo' that can store integer values.

6.6.2 Two-Dimensional Array Creation

After declaration of a two-dimensional array, it can be created in memory by using the following syntax in Java.

```
Array_Name = new Data_Type [ Array_Size1 ][ Array_Size2 ] ;
```

In the above syntax, Array_Size1 is the maximum number of rows and Array_Size2 is the maximum number of columns that can be possible in the two dimensional array, Array_Name. So, the total number of array elements that can be stored in Array_Name is (Array_Size1 × Array_Size2). For example, let us consider the following Java statement.

```
ArrTwo = new int [10][40] ;
```

The above statement will create a two-dimensional integer array, ArrTwo that can store at most 10 × 40 = 400 integer numbers.

6.6.3 Two-Dimensional Array Initialization

After creation of a two dimensional array, initial values can be stored by using the following syntax in Java.

Array_Name[SubscriptValue1][SubscriptValue2] = Data1;

The above syntax is used to store Data1 into the cell of the two dimensional array, 'Array_Name' whose first subscript is

SubscriptValue1 and second subscript is SubscriptValue2.

The following Java program will demonstrate the insertion of values in a two dimensional array.

Example: Java program to insert integer data into a two-dimensional array

```
import java.util.*;
class insertDataTwoDArray
{
   public static void main(String []args)
   {
      int arrTwo[][]; //2D-array declaration
     System.out.println("Enter number of rows:");
      Scanner S = new Scanner(System.in);
      int rowNo = S.nextInt();
   System.out.println("Enter number of columns:");
       int colNo = S.nextInt();
      //Two-dimensional array creation in memory
       arrTwo = new int[rowNo][colNo];
  System.out.println("\n\nEnter array elements:");
       //Insertion of integer values into the
       //2D-array
       for(int i=0;i<rowNo;i++)
       {
         for(int j=0;j<colNo;j++)
         {
              System.out.println("Enter
              ("+i+","+j+")th element:");
              int data= S.nextInt();
              arrTwo[i][j]=data;
         }
       }
     System.out.println("\nArray elements are:");
//Display array elements available in the two-
dimensional array
for(int i=0;i<rowNo;i++)
```

```
{
     for(int j=0;j<colNo;j++)
     {
          System.out.print("  "+arrTwo[i][j]);
     }
      System.out.println("\n");
}
    S.close();
   }
}
```

Output of the program:

```
Enter number of rows:
2
Enter number of columns:
3


Enter array elements:
Enter (0,0)th element:
12
Enter (0,1)th element:
6
Enter (0,2)th element:
8
Enter (1,0)th element:
43
Enter (1,1)th element:
22
Enter (1,2)th element:
78

Array elements are:
  12  6  8

  43  22  78
```

In Java, creation and initialization of a two-dimensional array can also be performed at the time of its declaration. For example, let us consider the following statements in Java to initialize two dimensional arrays at the time of its declaration.

```
int  ArrTwo1[  ][  ] ={ {2, 5, 3, 9}, {8, 7, 1, 0},
{7, 3, 5, 1}};
int [ ][ ] ArrTwo2={
                    {2, 5, 3, 9},
                    {8, 7, 1, 0},
                    {7, 3, 5, 1}
                };
```

The following Java program will demonstrate the creation

and initialization of a two-dimensional array at the time of its declaration.

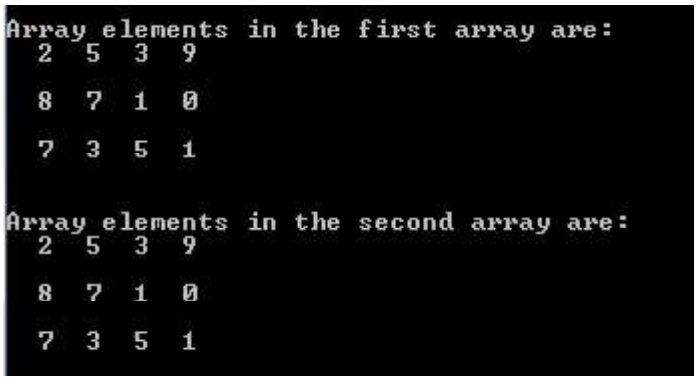
Example: Java program to initialize two-dimensional array

```
class initTwoDArray
{
   public static void main(String []args)
   {
      //Initialization of the first
      //two-dimensional array
       int  arrTwo1[ ][ ] =  { {2, 5, 3, 9},
                               {8, 7, 1, 0},
                               {7, 3, 5, 1}};
//Initialization of the second two-dimensional array
       int  [ ][ ]arrTwo2 =  {
                                    {2, 5, 3, 9},
                                    {8, 7, 1, 0},
                                    {7, 3, 5, 1}
                                 };
System.out.println("\nArray elements in the first
array are:");
//Display array elements available in the first
two-dimensional array
   for(int i=0;i<3;i++)
   {
        for(int j=0;j<4;j++)
        {
            System.out.print("  "+arrTwo1[i][j]);
        }
         System.out.println("\n");
    }
 System.out.println("\nArray elements in the second
array are:");
//Display array elements available in the second
//two-dimensional array
for(int i=0;i<3;i++)
 {
```

```
    for(int j=0;j<4;j++)
    {
       System.out.print("  "+arrTwo2[i][j]);
  }
  System.out.println("\n");
 }

}
}
```

Output of the program:



Example: Java program to implement the addition of two two-dimensional arrays

```
import java.util.*;
class addTwoDArray
{
   public static void main(String []args)
   {
    // Two-dimensional array declaration
    int arrTwo1[][],arrTwo2[][],arrResult[][];
    System.out.println("\nEnter number of rows:");
    Scanner S= new Scanner(System.in);
    int rowNo=S.nextInt();
  System.out.println("\nEnter number of columns:");
   int colNo=S.nextInt();
  //First two-dimensional array creation in memory
   arrTwo1 = new int[rowNo][colNo];
  //Second two-dimensional array creation in memory
arrTwo2 = new int[rowNo][colNo];
```

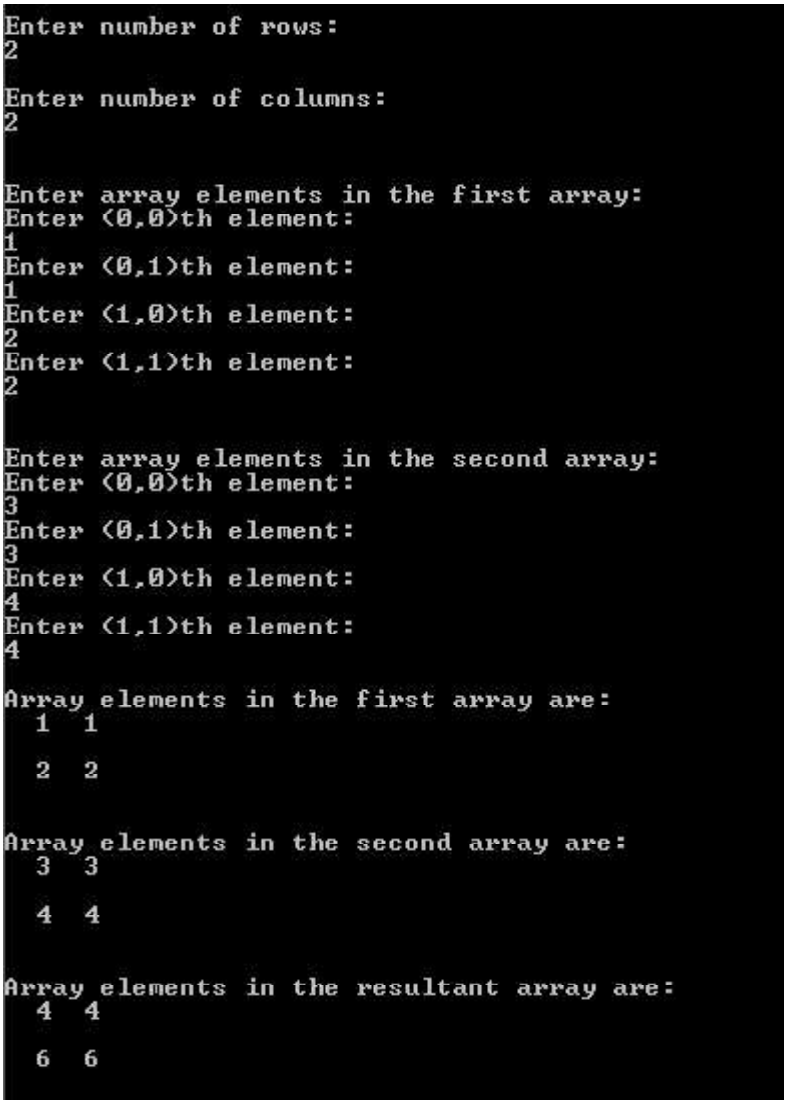
```
//Resultant two-dimensional array creation in memory
 arrResult = new int[rowNo][colNo];
System.out.println("\n\nEnter array elements in the
first array:");
// Insertion of integer values into the first two-
dimensional array
for(int i=0;i<rowNo;i++)
{
     for(int j=0;j<colNo;j++)
     {
        System.out.println("Enter ("+i+","+j+")th
       element:");
       int data= S.nextInt();
       arrTwo1[i][j]=data;
    }
}
System.out.println("\n\nEnter array elements in
the second array:");
// Insertion of integer values into the
//second two-dimensional array
for(int i=0;i<rowNo;i++)
{
     for(int j=0;j<colNo;j++)
     {
        System.out.println("Enter ("+i+","+j+")th
       element:");
       int data= S.nextInt();
       arrTwo2[i][j]=data;
     }
}
// Addition of arrTwo1 and arrTwo2
for(int i=0;i<rowNo;i++)
   {
     for(int j=0;j<colNo;j++)
      {
          arrResult[i][j]=arrTwo1[i][j]+arrTwo2[i][j];
```

```
        }
      }
    System.out.println("\nArray elements in the first
     array are:");
     //Display array elements available in the first
     //two-dimensional array
     for(int i=0;i<rowNo;i++)
      {
         for(int j=0;j<colNo;j++)
         { System.out.print("  "+arrTwo1[i][j]);
            }
           System.out.println("\n");
      }
System.out.println("\nArray elements in the second
array are:");
//Display array elements available in the second
//two-dimensional array
for(int i=0;i<rowNo;i++)
{
   for(int j=0;j<colNo;j++){
   System.out.print("  "+arrTwo2[i][j]);
}
System.out.println("\n");
}
System.out.println("\nArray elements in the
resultant array are:");
//After addition, display array elements available
in the resultant two-dimensional array

 for(int i=0;i<rowNo;i++)
 {
   for(int j=0;j<colNo;j++)
    {
      System.out.print("  "+arrResult[i][j]);
    }
```

```
        System.out.println("\n");
            }
            S.close();
          }
}
```

Output of the program:



6.7 ARRAYS CLASS

'Arrays' is a class available in the Java Collections framework under java. util package. This class contains different static methods to manipulate

arrays in Java. Array of primitive data types or objects may be passed as argument to these methods. These methods may require other arguments also as per the requirement and data type of these arguments must be same with the data type of the array if these are not array index. If some index of the array is required to pass as argument to these methods then it must be an integer value which is less than or equal to the size of the array, otherwise, an exception will be thrown by the corresponding method. 'Arrays' class is inherited from the 'Object' class. Examples of some useful methods provided by Arrays class are presented in the following table.

Function Prototype	**Job of the function**
static int binarySearch(char chArr[ ], char c)	A particular character value is searched using Binary search algorithm in an array.
static int binarySearch(char chArr[ ], int startIn, int endIn, char c)	A particular character value is searched using Binary search algorithm in a particular range of a character array.
static int compare(char chArr1[ ], char chArr2[ ])	Two character arrays are compared lexicographically.
static int compare(char chArr1[ ], int startIn1, int endIn1, char chArr2[ ] , int startIn2, int endIn2)	Particular ranges of two character arrays are compared lexicographically.
static char chArr1[ ] copyOf(char chArr2[ ], int Ln)	Copies a character array with a particular length into another character array.
static char chArr1[ ] copyOfRange (char chArr2[ ] , int startIn, int endIn)	Copies a particular range of a character array into another character array.
static boolean equals(char chArr1[ ], char chArr2[ ])	Checks whether two characters arrays are equal or not equal. If equal it returns true.

static boolean equals(char chArr1[ ], int startIn1, int endIn1, char chArr2[ ] , int startIn2, int endIn2)	Checks whether particular ranges of two characters arrays are equal or not equal. If equal it returns true.
static void fill(char chArr[ ] , char c)	A particular character is assigned to each array element of a character array.
static void fill(char chArr[ ] , int startIn, int endIn, char c)	A particular character is assigned to each array element of a particular range in a character array.
static int hashCode(char chArr[ ])	A hash code is returned based on the elements of a character array.
static int mismatch (char chArr1[ ] , char chArr2[ ])	Returns the index of the first mismatch between two character arrays. If there is no mismatch found then return -1.
static int mismatch(char chArr1[ ] , int startIn1, int endIn1, char chArr2[ ] , int startIn2, int endIn2)	Checks for the first mismatch between two character arrays in particular ranges and if mismatch is found then return the relative index. If no mismatch is available then return -1.
static void sort (char chArr[ ])	A character is sorted in ascending numerical order.
static void sort(char chArr[ ] , int startIn, int endIn)	A particular range of a character array is sorted in ascending order.
static String toString(char chArr[ ])	String representation for the contents of a character array is returned.

Table 6.1: Examples of Arrays Class Methods

Following Java programs will demonstrate the use of some Arrays class methods.

Example: Java program to copy a character array into another character array and then to sort it using Arrays class methods.

```
class copySortArray{
public static void main(String []args)
{
   int i;
   char [] chArr1 =
   {'A','G','K','M','D','O','F','H','U','B'};
   int N=chArr1.length;
   char [] chArr2 = Arrays.copyOf(chArr1, 10);
   System.out.println("\n\nArray elements in the
   original character array are: \n");
   for(i=0;i<N;i++)
    {
        System.out.print("  "+chArr1[i]);
    }
   System.out.println("\nArray elements in
   the copied character array are: \n");
   for(i=0;i<N;i++)
    {
      System.out.print("  "+chArr2[i]);
    }
   Arrays.sort(chArr2);
   System.out.println("\nAfter sorting,
   array elements in the copied character
   array are:\n");
   for(i=0;i<N;i++)
    {
             System.out.print("  "+chArr2[i]);
    }
```

```
    }
}
```

Output of the program:

```
Array elements in the original character array are:

  A  G  K  M  D  O  F  H  U  B
Array elements in the copied character array are:

  A  G  K  M  D  O  F  H  U  B
After sorting, array elements in the copied character array are:

  A  B  D  F  G  H  K  M  O  U
```

In the above program, 'char [ ] chArr2 = Arrays.copyOf(chArr1, 10);' is used to copy the character array 'chArr1' into the character array 'chArr2' where the argument represents the number of elements to be copied and that is 10 in this case. Again, 'Arrays.sort(chArr2);' is used to sort the array elements that are available in 'chArr2'.

6.8 STRINGS

String is a collection of character values. So strings can be stored and manipulated using character arrays. In Java, two classes are available to handle strings efficiently. These two classes are String and StringBuffer.

6.8.1 String Class

In Java, strings can be declared as objects of class String. Let us consider the following Java statements to declare and create a string.

```
String string1;
string1 = new String ("Welcome to KKHSOU");
```

In the first statement, string1 is declared as the object of String class. In the second statement, a string, "Welcome to KKHSOU" is created in memory by using new operator. The above statements can also be written as:

```
String string1 = new String ("Welcome to KKHSOU");
```

Some of the commonly used constructors of String class are presented in the following table.

Constructor	Job of the Constructor
String()	This constructor creates String object with empty character sequence.
String(char chArr [ ])	This constructor creates a string with the character sequence that is available in a particular character array.
String(char chArr [ ] , int startInd, int count)	This constructor creates a string with the character sequence that is available in a subarray of a particular character array.
String(String original)	This constructor creates a string object that contains a string which is same with the string object passed as argument.
String(StringBuffer strBuff)	This constructor creates a string object that contains a character sequence which is the content of a particular string buffer.

Table 6.2: Some Commonly Used Constructors of String Class

A simple Java program is presented below to demonstrate the declaration and creation of a string using String class.

Example: Java program to create a string using String class and to display it.

```
class createString
{
   public static void main(String args[])
  {
     // String object declaration
     String string1;
     // Creation of a string in memory
     string1 = new String("Welcome to KKHSOU");
     System.out.println( "\nThe string is:");
     System.out.println(string1);
  }
}
```

Output of the program:

```
The string is:
Welcome to KKHSOU
```

6.8.2 String Arrays

String arrays are the arrays where the array elements are the objects of String class. A string array is used to store multiple strings. Let us consider the following Java statement to create a string array.

String stringArr [ ] = new String [10];

In the above statement, stringArr is a string array that can store at most 10 numbers of strings.

The following Java program demonstrates the creation of a string array.

Example: Java program to create an array of strings and to display it.

```
import java.util.*;
class createStringArray{
public static void main(String args[])
{
    // Creation of string array
    String  stringArr[ ] = new String [ 5 ];
    int i;
    System.out.println("\nEnter five strings:");
    for(i=0;i<5;i++)
    {
   Scanner scr= new Scanner(System.in);
   System.out.print("Enter "+(i+1)+"th string: ");
   stringArr[i]= scr.nextLine(); //Input strings
   }
  System.out.println( "\nEntered strings are:");
  for(i=0;i<5;i++)
  {
        System.out.println(stringArr[i]);
  }
 }
}
```

Output of the program:



In Java, two strings can be concatenated by using '+' operator. The following Java program demonstrates this use of '+'operator.

Example: Java program to concatenate two strings by using '+'operator.

```
class concatenateString
{
      public static void main(String args[])
      {
       // String object declaration
        String string1, string2, string3;
        // Creation of the first string
        string1 = new String ("Welcome to ");
        // Creation of the second string
        string2 = new String ("KKHSOU");
       // Concatenation of string1 and string2
        string3 = string1 + string2;
       System.out.println( "\nThe first
        string is:\n");
       System.out.println(string1);
       System.out.println( "\nThe second
        string is:\n");
       System.out.println(string2);
       System.out.println("\nAfter
      concatenation, the resultant string is:\n");
```

```
        System.out.println(string3);
          }
      }
```

```
The first string is:
Welcome to
The second string is:
KKHSOU
After concatenation,the resultant string is:
Welcome to KKHSOU
```

6.8.3 String Methods

In Java, various methods are provided in String class to work with strings. Some important methods of String class are presented in the following table.

Function Prototype	**Job of the function**
char charAt(int Ind)	Returns the character available at a particular index in a string.
int compareTo(String string2)	Two strings are compared lexicographically.
int compareToIgnoreCase(String string2)	Two strings are compared lexicographically by ignoring case of the characters.
String concat(String string2)	Concatenates one string to the end of another string.
boolean contains(CharSequence chSeq)	Search for a particular sequence of characters in a string and if found then return true.
boolean contentEquals(CharSequence chSeq)	A particular sequence of characters is compared with a string.
boolean contentEquals(StringBuffer stBuff)	A particular StringBuffer is compared with a string.
boolean endsWith(String su)	Checks whether a string is end with a particular suffix.
boolean equals(String string2)	Returns true if one string is equal to another string.
boolean equalsIgnoreCase(String string2)	Returns true if one string is equal to another string by ignoring case of characters.

void getChars(int start, int end, char chArr [ ] , int chStart)	Characters from a string are copied to a character array.
int hashCode()	Hash code for a string is estimated and returned.
int indexOf(int c)	Returns the index of the first occurrence of a particular character in a string.
int indexOf(int c, int startInd)	Searches a particular character in a string, starting from a particular index and return the index of the first occurrence of the character.
int indexOf(String subStr)	Searches a particular substring within a string and return the index of the first occurrence of the substring.
int indexOf(String subStr, int startInd)	Searches a particular substring within a string, starting from a particular index and return the index of the first occurrence of the substring.
boolean isBlank()	If a string is empty then it returns true.
boolean isEmpty()	If the length of a string is 0 then it returns true.
int lastIndexOf(int c)	Returns the index of a string where a particular character is available for the last time.
int lastIndexOf(int c, int startInd)	Returns the index of a string where a particular character is available for the last time and in this case searching operation is performed in backward direction starting from a particular index.
int lastIndexOf(String subStr)	Returns the index of a string from where a particular substring is available for the last time.
int lastIndexOf(String subStr, int startInd)	Returns the index of a string from where a particular substring is available for the last time and in this case searching operation is performed in backward direction starting from a particular index.
int length()	Returns the length of a string.
String repeat(int num)	Returns a string where the string value is the concatenation of a string repeated N times. N is passed as argument to this method.
String replace(char existChar, char inChar)	Returns a string after replacing all occurrences of an existing character in a string with a new character.

String replaceAll(String reg, String inSubStr)	Replaces all substrings that match a particular regular expression in a string with an input substring.
String replaceFirst(String reg, String inSubStr)	Replaces the first substring that matches a particular regular expression in a string with an input string.
boolean startsWith(String pre)	Returns true if a string starts with a particular prefix.
String strip()	All leading and trailing white spaces of a string are removed and then the string is returned.
Char [ ] toCharArray()	Converts a string to a character array and the new array is returned.
String toLowerCase()	Converts all characters available in a string to lower case.
String toUpperCase()	Converts all characters available in a string to upper case.
static String valueOf(char ch)	String representation of a character is returned.
static String valueOf(char chArr[])	String representation of a character array is returned.

Table 6.3: Some Important Methods of String Class

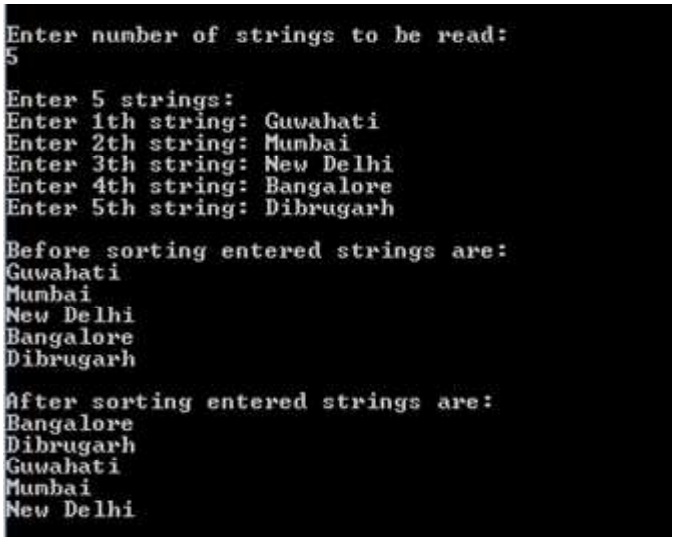
The following program demonstrates the use of some String class methods.

Example: Java program to input strings into an array and to sort it using Bubble sort algorithm.

```
import java.util.*;
class sortStringArray
{
  public static void main(String args[])
  {
  System.out.println("\n\nEnter number of strings
   to be read:");
   Scanner S = new Scanner(System.in);
   int stringNo = S.nextInt();
   String temp=new String();
   // Creation of string array
  String  stringArr[ ] = new String [ stringNo ];
    int i,j;
```

```
System.out.println("\nEnter"+stringNo+" strings:");
for(i=0;i<stringNo;i++){
   Scanner scr= new Scanner(System.in);
  System.out.print("Enter "+(i+1)+"th string: ");
   stringArr[i]= scr.nextLine(); //Input strings
 }
   System.out.println( "\nBefore sorting
   entered strings are:");
for(i=0;i<stringNo;i++){
   System.out.println(stringArr[i]);
}
 // Sorting using Bubble sort algorithm
 for(i=0;i<stringNo-1;i++){
 for(j=0;j<stringNo-i-1;j++)
 {
  if(stringArr[j].compareTo(stringArr[j+1])>1)
   {
        temp = stringArr[j];
       stringArr[j] = stringArr[j+1];
        stringArr[j+1] = temp;
    }
  }
}
System.out.println( "\nAfter sorting entered s
trings are:");
for(i=0;i<stringNo;i++)
{
   System.out.println(stringArr[i]);
}
}
}
```

Output of the program:



In the above program, the String class method, 'compareTo(String string2)' is used to compare two strings.

Example: Java program to input names of some employees and to find out the names which contain 'Kumar' as middle name.

```
import java.util.*;
class findNames
{
      public static void main(String args[]){
      System.out.println("\n\nEnter the number of
       names to be read:";
      Scanner S = new Scanner(System.in);
      int stringNo = S.nextInt();
      String temp = new String();
      // Creation of string array
   String  stringArr[ ] = new String [ stringNo ];
      int i,j;
      System.out.println("\nEnter "+stringNo+"
      Employees'names:");
      for(i=0;i<stringNo;i++)
        {
          Scanner scr= new Scanner(System.in);
              System.out.print("Enter  "+(i+1)+"th
```

```
Employee name: ");
      stringArr[i]= scr.nextLine(); //Input Employee names
      }
      System.out.println( "\nNames with middile
      name 'Kumar' are:");
      for(i=0;i<stringNo;i++)
      {
         //Display Employee names which contain 'Kumar'
           as middle name
           temp = stringArr[i].toUpperCase();
           //Convert each character to upper case
            if(temp.contains("KUMAR"))
                System.out.println(stringArr[i]);
        }
      }
```

```
Enter the number of names to be read:
5

Enter 5 Employees'names:
Enter 1th Employee name: Bipul Kumar Deka
Enter 2th Employee name: Rinku Moni Nath
Enter 3th Employee name: Ankur Gogoi
Enter 4th Employee name: Rajib Kumar Dubey
Enter 5th Employee name: Pranab Das

Names with middile name 'Kumar' are:
Bipul Kumar Deka
Rajib Kumar Dubey
```

In the above program, the String class method 'toUpperCase()' is used to convert each character of a string into upper case and 'contains(CharSequence chSeq)' is used to check whether the character sequence, 'KUMAR' is available or not available in a string.



CHECK YOUR PROGRESS

Q 1: In Java, subscript of an array is started with _____.

(a) 0

(b) 1

(c) 2

(d) None of the above

Q 2: Which of the following is the correct statement in Java to declare a one-dimensional array?

(a) int arr { };

(b) int [ ] arr;

(c) int ( ) arr;

(d) None of the above

Q 3: Which of the following class is provided in Java to manipulate arrays?

(a) String

(b) Array

(c) Arrays

(d) Both (a) and (b)

Q 4: Which of the following function checks whether two character arrays are equal or not equal?

(a) compare(char chArr1[ ], char chArr2[ ]);

(b) mismatch (char chArr1[ ] , char chArr2[ ]);

(c) equals(char chArr1[ ], char chArr2[ ]);

(d) None of the above

Q 5: Which of the following class is not used to handle strings?

(a) String

(b) StringBuffer

(c) Strings

(d) None of the above

Q 6: Fill in the blanks:

(a) Each element in an array is accessed by an integer value referred as _____.

(b) _____ returns the string representation for the contents of a character array.

(c) hashCode(char chArr[ ]) returns _____ of a character array.

(d) String(char chArr [ ]) constructs a string with _____.

(e) _____returns the character available at a particular index in a string.

6.8.4 StringBuffer Class

One limitation of String class object is that characters of such objects cannot be modified and deleted. Also new characters cannot be added. Some of the String class methods can only return modified new strings but they cannot modify the original strings. So in Java, StringBuffer class is provided to create strings with flexible lengths and the content of these strings can be modified. This means that using StringBuffer class the characters of original strings can be modified and deleted. Also new characters and substrings can be added to the original strings. The constructors of StringBuffer class are presented in the following table.

Constructor	**Job of the Constructor**
StringBuffer( )	This constructor creates an empty string buffer. In this case the initial capacity of the string buffer is 16 characters.
StringBuffer(int strLen)	This constructor also creates an empty string buffer but in this case the initial capacity is passed as argument.
StringBuffer(String initString)	This constructor creates a string buffer with the characters that are available in a particular string.
StringBuffer(CharSequence chSeq)	This constructor creates a string buffer with the characters that are available in a particular CharSequence.

Table 6.4: Constructors of StringBuffer Class

Some important methods of StringBuffer class are presented in the following table:

Function Prototype	**Job of the function**
StringBuffer append(char ch)	Appends a particular character to a string that is stored in a StringBuffer object.

StringBuffer append(char chArr [ ])	Appends a particular character array to a string that is stored in a StringBuffer object.
StringBuffer append(String string2)	Appends a particular string to a string that is stored in a StringBuffer object.
int capacity()	Returns the capacity of a string buffer.
char charAt(int ind)	Returns the character available at a particular index from a string stored in a StringBuffer object.
int compareTo(StringBuffer strBuff)	Compares lexicographically two strings stored in two different StringBuffer objects.
StringBuffer delete(int IndBeg, int IndEnd)	Delete the characters available in a particular range of a string.
StringBuffer deleteCharAt(int ind)	Delete the character available at a particular position in a string.
void ensureCapacity(int minCap)	Sets the minimum capacity of a string buffer.
int indexOf(String string2)	Returns the index of the first occurrence of a particular substring in a string.
int indexOf(String string2, int startInd)	Returns the index of the first occurrence of a particular substring in a string where searching is started at a particular index.
StringBuffer insert(int ind, char c)	Inserts a particular character into a string.
StringBuffer insert(int ind, char chArr[ ])	Inserts a particular character array into a string.
StringBuffer insert(int ind, String string2)	Inserts a particular string into a string.
int lastIndexOf(String string2)	Returns the index of the first occurrence of a particular substring in a string.
int lastIndexOf(String string2, int startInd)	Returns the index of the last occurrence of a particular substring in a string where searching is started at a particular index in backward direction.
StringBuffer replace(int indBeg, int indEnd, String string2)	Replaces the characters of a string available at a particular range with the characters of a particular string.
StringBuffer reverse()	Reverses the character sequence in a string.

void setCharAt(int ind, char c)	Sets a particular character to a particular index of a string.
void setLength(int len)	Sets the length of a string.
String substring(int indBeg)	Returns a substring available at a particular index of a string.
String substring(int indBeg, int indEnd)	Returns a substring which contains the characters that are available in a range of a string.

Table 6.5: Some Important Methods of StringBuffer Class

The following program demonstrates the creation of string using StringBuffer class and use of some StringBuffer class methods.

Example: Java program to input two strings using StringBuffer class and to append the second string at the end of the first string using StringBuffer class method.

```
import java.util.*;
public class appendString
{
  public static void main(String[] args)
  {
   StringBuffer strBuffer1 = new StringBuffer("");
   StringBuffer strBuffer2 = new StringBuffer("");
   System.out.print("\nEnter the first string: ");
    Scanner sc = new Scanner(System.in);
    strBuffer1.append(sc.nextLine());
   System.out.print("\nEnter the second string: ");
    strBuffer2.append(sc.nextLine());
    System.out.println("\nThe first string is:");
    System.out.println(strBuffer1);
   System.out.println("\nThe second string is:");
    System.out.println(strBuffer2);
    strBuffer1.append(strBuffer2);
    System.out.println("\nAfter attachment of
    the second string,the first string is:\n");
    System.out.println(strBuffer1);
```

```
        }
    }
```

Output of the program:

```
Enter the first string: Gauhati

Enter the second string: University

The first string is:
Gauhati

The second string is:
University

After attachment of the second string,the first string is:

GauhatiUniversity
```

In the above program, 'append(String string2)' is used to append a particular string at the end of a string that is stored in a StringBuffer object.

6.9 VECTORS

In Java, vectors are the generic dynamic arrays. These arrays can store objects of any types. As vectors are dynamic arrays, these can contain any number of objects. So, the number of objects stored in a vector can be increased or decreased as required by a particular program.

In Java, Vector class is provided in java.util package to implement vectors. It implements the List interface. Vector class provides two data members to efficiently handle storage management. These members are capacity and capacityIncrement. The value of capacity is no less than the vector size. When the size of a vector becomes larger than its capacity then the capacity is automatically increased by the size which is stored in capacityIncrement. capacityIncrement is a protected data member of Vector class. Two more protected members are defined in this class that are elementCount and elementData. The value in elementCount refers the number of elements that are stored in a Vector class object. On the other hand, elementData is an array that holds the components of a vector.

The constructors of Vector class are presented in the following table.

Constructor	Job of the Constructor
Vector()	This constructor creates an empty vector. The initial capacity of this vector is 10 and standard capacity increment is 0.
Vector(int initCap)	This constructor creates an empty vector but in this case the initial capacity of the vector is passed as argument and its capacity increment is zero.
Vector(int initCap, int capInc)	This constructor also creates an empty vector but in this case, both the initial capacity and capacity increment are passed as arguments.
Vector(Collection col)	This constructor constructs a vector which stores the elements of a particular collection.

Table 6.6: Constructors of Vector Class

Some important methods of Vector class are presented in the following table.

Function Prototype	Job of the function
boolean add(S elem)	It appends a particular element to the end of a vector.
void add(int ind, S elem)	It inserts a particular element at a particular index in a vector.
int capacity()	It returns the present capacity of a vector.
void clear()	It deleted all the elements of a vector.
Object clone()	Clone of a vector is returned.
boolean contains(Object elem)	If a vector contains a particular element then it returns true.
boolean containsAll(Collection coll)	If a vector contains all the elements that are available in a particular Collection then it returns true.
void copyInto(Object[] arr)	Components of a vector are copied to a particular array.
void ensureCapacity(int capMin)	It ensures the minimum capacity of a vector and so, if requires it increases the capacity to this minimum value.

boolean equals(Object elem)	Equality of a particular object is checked with a vector.
int hashCode()	It returns the hash code of a vector.
int indexOf(Object elem)	The index of the first occurrence of a particular element in a vector is returned. If the element is not available in the vector then it returns -1.
int indexOf(Object elem, int ind)	The index of the first occurrence of a particular element in a vector is returned and in this case forward searching is performed started with a particular index. If the element is not available in the vector then it returns -1.
void insertElementAt(S ob, int ind)	It adds a particular object to a vector at a particular index.
boolean isEmpty()	It returns true if a vector is empty.
int lastIndexOf(Object elem)	The index of the last occurrence of a particular element in a vector is returned. If the element is not available in the vector then it returns -1.
int lastIndexOf(Object elem, int ind)	The index of the last occurrence of a particular element in a vector is returned and in this case backward searching is performed started with a particular index. If the element is not available in the vector then it returns -1.
boolean remove(Object elem)	It deletes the first occurrence of a particular element from a vector.
void removeAllElements()	It deletes all components of a vector and sets its size to zero.
boolean removeElement(Object elem)	It deletes the first occurrence of a particular element from a vector.
void removeElementAt(int ind)	It removes the component that is available at a particular index in a vector.
protected void removeRange(int startInd, int endInd)	It deletes all the elements that are available in a range of a vector.
void setSize(int value)	It sets the size of a vector to a value that is passed as an argument.
int size()	It gives the total number of components that are available in a vector.
Object[ ] toArray()	It returns an array that contains all the elements of a vector.
String toString()	It gives a string representation of a vector.
void trimToSize()	It reduces the capacity of a vector up to its present size.

Table 6.7: Some Important Methods of Vector Class

The following program demonstrates the creation of vector and use of some Vector class methods.

Example: Java program to create an empty vector and append new elements into it. At the end, the program will display the current capacity and size of the vector.

```
import java.util.*;
public class appendElementVector
{
    public static void main(String arg[])
    {
        //Creation of an empty Vector
        Vector vcr = new Vector();
        //Append elements into the vector
        vcr.add('A');
        vcr.add(5);
        vcr.add("KKHSOU");
        vcr.add(23.5);
       //Display elements available in the vector
        System.out.println("\nElements available
        in the vector are:\n");
        System.out.println(vcr);
        System.out.println("\nPresent size of
        the vector is :"+vcr.size());
        System.out.println("\nPresent capacity
        of the vector is :"+vcr.capacity());
     }
    }
```

Output of the program:

```
Elements available in the vector are:
[A, 5, KKHSOU, 23.5]
Present size of the vector is :4
Present capacity of the vector is :10
```

In the above program, 'add(S elem)' is used to append a particular element to the end of a vector. Then 'size()' is used to find out the size of the Vector object and 'capacity()' is used to find out the current capacity of the vector.



CHECK YOUR PROGRESS

Q 1: Which of the following class in Java can be used to create strings with flexible lengths?

(a) String
(b) StringBuffer
(c) Arrays
(d) None of the above

Q 2: Which of the following constructor creates an empty string buffer with an initial capacity of 16 characters?

(a) StringBuffer( )
(b) StringBuffer(int strLen)
(c) StringBuffer(String initString)
(d) None of the above

Q 3: Vector (int initCap) creates an empty vector with initial capacity of _____.

(a) 10
(b) 16
(c) initCap
(d) None of the above

Q 4: ____ returns capacity of a vector.

(a) ensureCapacity(int capMin)
(b) capacity()
(c) size()
(d) None of the above

Q 5: ____reduces the capacity of a vector up to its present size.

(a) trimSize()
(b) reduceSize()

(c)	trimToSize()
(d)	None of the above
Q 6:	State True/False:
(a)	capacity() returns the minimum capacity of a string buffer.
(b)	Strings created by using StringBuffer class cannot be modified.
(c)	In Java, vectors are the generic dynamic arrays.
(d)	Vector class is available in java.util package



6.10 LET US SUM UP

- Array is a data structure that store similar type of data. These data are stored in contiguous memory locations. In Java programming, arrays are container objects that store data of same types. Each element of an array is accessed by an integer value termed as subscript.
- The syntax of one-dimensional array declaration in Java is presented as follows.

  Data_Type [ ] Array_Name ;

  Or

  Data_Type Array_Name [ ] ;
- The syntax of using 'new' operator to create a one dimensional array in memory is presented as follows.

  Array_Name = new Data_Type [Array_Size] ;
- In Java, the syntax to create and initialize a one dimensional array at the time of its declaration is presented as follows.

  Data_Type [ ] Array_Name = { Data1, Data2 , Data3 , Data4,…….};
- Array with multiple subscripts is termed as multi-dimensional array.
- Two-dimensional array is an example of multi-dimensional arrays where two subscripts are required to access an array element.
- The syntax to declare a two-dimensional array in Java is presented as follows.

  Data_Type [ ][ ] Array_Name ;

  Or

  Data_Type Array_Name [ ][ ] ;

- The syntax in Java to create a two dimensional array in memory using 'new' operator is presented as follows.
  Array_Name = new Data_Type [Array_Size1][Array_Size2] ;
- 'Arrays' is a class available in the Java Collections framework under java. util package. This class contains different static methods to manipulate arrays in Java.
- String is a collection of character values. In Java, String and StringBuffer are the two classes used to handle strings efficiently.
- A string array is used to store multiple strings.
- StringBuffer class is provided to create strings with flexible lengths and the content of these strings can be modified.
- In Java, vectors are the generic dynamic arrays. Vector class is provided in java.util package to implement vectors.



6.11 FURTHER READING

1) Balaguruswamy, E. (2014). Programming with Java-A Primer. McGraw-Hill Professionals.
2) Wu, C. T. (2006). An Introduction to object-oriented programming with Java TM. Mcgraw-Hill Incorporated.
3) Schildt, H., & Coward, D. (2014). Java: the complete reference (p. 1312). New York: McGraw-Hill Education.
4) Horstmann, C. S., & Cornell, G. (2001). Core Java 2: Fundamentals (Vol. 1). Prentice Hall Professional.
5) https://docs.oracle.com/javase/tutorial/ Date of Access: 12-04-2022



6.12 ANSWERS TO CHECK YOUR PROGRESS

Ans to Q No 1: (a)

Ans to Q No 2: (b)

Ans to Q No 3: (c)

Ans to Q No 4: (c)

Ans to Q No 5: (c)

Ans to Q No 6: (a) Subscript

Ans to Q No 6: (b) toString(char chArr[ ])

Ans to Q No 7: (c) a hash code

Ans to Q No 8: (d) the character sequence that is available in a particular character array

Ans to Q No 9: (e) charAt (int Ind)

Ans to Q No 10: (b)

Ans to Q No 11: (a)

Ans to Q No 12: (c)

Ans to Q No 13: (b)

Ans to Q No 14: (c)

Ans to Q No 15: (a) False

Ans to Q No 16: (b) False

Ans to Q No 17: (c) True

Ans to Q No 18: (d) True



6.13 MODEL QUESTIONS

Q 1: How a two-dimensional array can be created in memory by using Java statement?

Q 2: What is the purpose of Arrays class?

Q 3: Write a Java program to merge two integer arrays so that the resultant array becomes sorted.

Q 4: How strings can be handled in Java?

Q 5: Write down the different constructors of String class.

Q 6: Write a Java program to check an input string is palindrome or not palindrome.

Q 7: What are the differences between String class and StringBuffer class?

Q 8: What is vector? How vector can be created and manipulated in Java?

Q 9: What is the difference between an array and a vector?

*** ***** ***