UNIT 5 : ARRAYS, STRINGS AND VECTORS

UNIT STRUCTURE

5.1 Learning Objectives

5.2 Introduction

5.3 Arrays

- 5.3.1 Declaring Array Variables
- 5.3.2 Creating an Array
- 5.3.3 Initializing Array

5.4 Two-Dimensional Arrays

5.5 Strings

- 5.5.1 String Methods
- 5.5.2 StringBuffer Class

5.6 Vectors

5.7 Let Us Sum Up

5.8 Further Readings

5.9 Answers To Check Your Progress

5.10 Model Questions

5.1 LEARNING OBJECTIVES

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

- declare and initialize arrays for Java programming
- use the method *arraycopy ()*
- define multi-dimensional arrays
- describe strings
- describe vectors

5.2 INTRODUCTION

In the previous unit, we have discussed classes and objects in Java. Array is a linear and homogenous data structure. In this unit, we will discuss the arrays, how the arrays are declared and initialized in Java language. In

addition to this, the concept of the strings and vectors are also covered in this unit in detail. In the next unit, we will explore the concepts related to inheritance and polymorphism.

5.3 ARRAYS

In the basic built-in Java data types, each identifier corresponds to a single value. But when you want to handle a set of values of the same type, identifiers are unable to srore these set of values.In such situation array is essential.

An array is a collection of related data stored in consecutive memory locations under a common variable name. Each variable in the array is called an **array element**. To refer to a particular element in an array you use the array name combined with an integer value of type ***int***, called an **index**. You are already familiar with how to declare an array in C or C++. In Java, it is slightly different from C/C++. There are three steps to create an array in Java :

a) declare a variable to hold array
b) create memory locations
c) put values into the memory locations.

5.3.1 Declaring Array Variables

You are not obliged to create the array itself when you declare the array variable. The array variable is distinct from the array itself. There are two ways to declare the array variables in Java:

data type arrayname[ ];

or

data type[ ] arrayname;

For example,
```
int sum[ ];
float percentage[ ];
int roll_number[ ];
int primes[ ];
```

The above declaration can be written as :

```
int[ ] sum;
```

```
float[ ] percentage;
int[ ] roll_number;
int[ ] primes;
```

Remember that in such types of declarations no memory has been allocated to store the array itself and the number of elements has not been defined.

5.3.2 Creating an Array

Creating an array means allocating memory for it. Java allows us to create arrays using ***new*** operator. The syntax is given below :

arrayname = new type[size];

For example,

```
sum = new int[4];
percentage = new float[10];
roll_number = new float[15];
primes = new int[10];
```

The keyword **new** indicates that you are allocating new memory for the array, and int[4] specifies you want capacity for 4 variables of type int in the array. Since each element in the **sum** array is an integer variable requiring 4 bytes, the whole array will occupy 16 bytes, plus 4 bytes to store the reference to the array.

When an array is created like this, all the array elements are initialized to a default value automatically. Similarly, for the percentage, roll_number and primes array also a fixed size will be allocated. The above declared and created arrays are **one-dimensional array** since each of its elements is referenced using one index.

5.3.3 Initializing Arrays

You can initialize an array with your own values when you declare it, and at the same time determine how many elements it will have. Following the declaration of the array variable, simply add

an equal sign followed by the list of element values enclosed between braces.

For example, **int[ ] primes = {2, 3, 5, 7, 11, 13, 17};**

Here, ***primes*** is an array of 7 elements. The array size is determined by the number of initial values; so no other statement is necessary to define the array.

If you specify initializing values for an array, you must include values for *all* the elements. If you only want to set some of the array elements to values explicitly, you should use an assignment statement for each element.

For example : int[ ] primes = new int[100];

primes[0] = 2;

primes[1] = 3;

The first statement declares and defines an integer array of 100 elements, all of which will be initialized to zero. The two assignment statements then set values for the first two array elements.

You can also initialize an array with an existing array. For example,

int[ ] even = {4, 6, 8, 10};

int[ ] number = even;

Here, both arrays refer to the same set of elements and you can access the elements of the array through either variable name – for example, even[2] refers to the same variable as number[2].

The following program demonstrates the initialization of one dimensional array :

//Program 5.1 : **To find the average of given list of numbers.**

```
class  Average
     {
          public  static  void  main(String  args[])
               {
      double nums[] ={50.1, 51.2, 52.3, 53.4, 54.5, 55.6, 56.7, 57.8,
```

```
                    58.9, 59.0};
     double result = 0;
     int i;
     for(i=0; i<10; i++)
     result = result + nums[i];
     System.out.println("Average is " + result / 10);
}
     }
```

Output : Average is 54.95

Array Length : You can refer to the length of the array using **length**, a data member of the array object. In our array example i.e. the array **sum,** its length can be assigned to an another variable as follows:

int size = sum.length

The following program 2 & 3 demonstrates the use of length object for computing the largest and the smallest number from a given list of numbers.

//Program 5.2 : **To find the smallest and largest number from a given list of number.**

```
class FindNumber
{
     public static void main(String[] args)
     {
          //array of 10 numbers
     int numbers[ ] = new int[]{12,83,50,18,22,72,41,29,43,21};
          //assign first element of an array to largest and smallest
          int smallest = numbers[0];
          int largetst = numbers[0];
          for(int i=1; i< numbers.length; i++)
               {
                    if(numbers[i] > largetst)
```

```
                    largetst = numbers[i];
                else if (numbers[i] < smallest)
                    smallest = numbers[i];
            }
        System.out.println("Largest Number is  : " + largetst);
        System.out.println("Smallest Number is : " + smallest);
        }
}
```

Output : Largest Number is : 83
Smallest Number is : 12

//Program 5.3 : **To check whether a given number is palindrome or not.**

```
class Palindrome
{
    public static void main(String[] args)
    {
        //array of numbers
         int numbers[ ] = new int[]{2332,42,11,223,24};
        //iterate through the numbers
          for(int i=0; i < numbers.length; i++)
           {
            int number = numbers[i];
            int reversedNumber  = 0;
            int temp=0;
            //reverse the number
            while(number > 0)
            {
                temp = number % 10;
                number = number / 10;
                reversedNumber = reversedNumber * 10 + temp;
           }
                if(numbers[i] == reversedNumber)
```

```
            System.out.println(numbers[i] + " is a palindrome number");
           else
          System.out.println(numbers[i] + " is not a palindrome number");
              }
       }
   }
```

Output :

2332 is a palindrome number
42 is not a palindrome number
11 is a palindrome number
223 is not a palindrome number
24 is not a palindrome number



Palindrome: A palindrome is a sequence of character that is identical from both side i.e., from left to right and right to left. For e.g., madam , 121, 52125 etc.

Reusing Array Variables : You can use an array variable to reference different arrays at different points in your program. Suppose you have declared and defined the variable primes as before:

int[ ] primes = new int[10]; // Allocate an array of 10 integer elements

This produces an array of 10 elements of type *int*. Perhaps you want the array variable primes to refer to a larger array, with 50 elements say. You would simply write :

primes = new int[50]; // Allocate an array of 50 integer elements

Now the variable primes refer to a new array of values of type *int* that is entirely separate from the original. When this statement is executed, the previous array of 10 elements is discarded, along with all the data values you may have stored in it. The variable primes can now only be used to reference elements of the new array.

5.4 TWO-DIMENSIONAL ARRAYS

A table organized in rows and columns is a very effective means for communicating many different types of information. In Java, we represent table as two-dimensional arrays. The general syntax for allocating a two dimensional array is :

type [ ][ ] arrayname = new type[rows][cols];

For example, int [ ][ ] matrix = new int [3][5];

This creates a two dimensional matrix having 3 rows and 5 columns that can store 15 integer values. We are already familiar with C or C++ how to access the elements of a two dimensional array. In Java also, the same techniques are used. For example, to refer to the elements at the second column of the third row we will indicate it as:

matrix [2][1]

We can also initialize a two dimensional array as shown below :

```
int [ ] [ ] matrix = {
                     { 10, 5,  –3, 9, 2 },
                     {  1 , 0, 14, 5, 6 },
                     { –1, 7,  4,  9,  2 }
                     };
```

There is no limit to the number of dimensions an array can have. We can declare three-dimensional, four-dimensional, and higher dimensional arrays. However, arrays with a dimension higher than 2 are not frequently used in object-oriented languages.

The following figure 5.1 depicts the representation of the matrix array in memory :

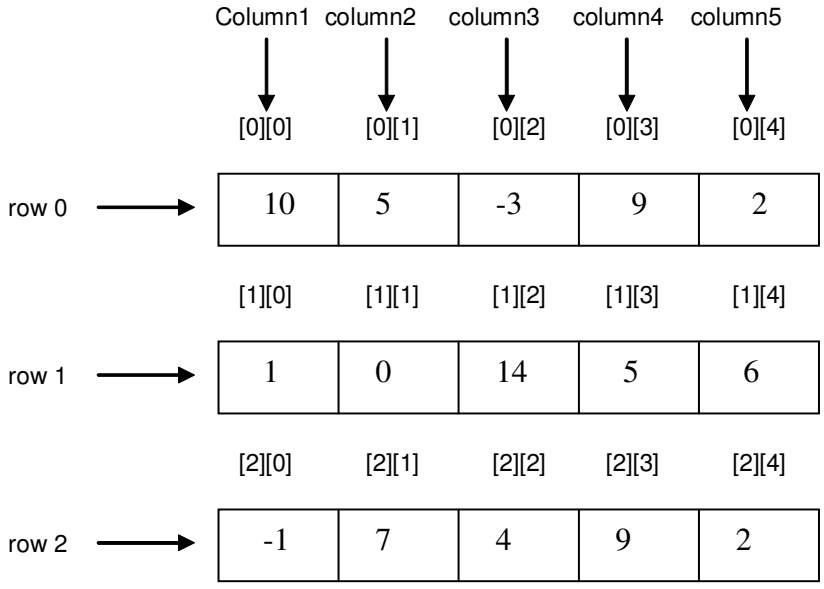


Figure 5.1 : Representation of matrix array

The following program demonstrate the use of two dimensional array in Java programming. Program 5.4 will simply display the initialized list of elements as output.

//Program 5.4 : **To display the elements of a two dimensional array in matrix form.**

```
class Matrix
{
     public static void main(String args[])
     {
          int mat[ ][ ] ={
                         { 10, 5, -3, 9, 2 },
                         { 1, 0, 14, 5, 6 },
                         { -1, 7, 4, 9, 2 },
                    };
          int i, j;
          for(i=0; i<3; i++)
          {
               for(j=0; j<5; j++)
               System.out.print(mat[i][j] + " ");
               System.out.println();
          }
     }
}
```

Output :

```
 10,   5, –3,   9,   2
  1,   0, 14,   5,   6
 –1,   7,   4,   9,   2
```

Multidimensional arrays are actually arrays of arrays. It means we can create or allocate the sub arrays (columns). It is possible to create sub arrays of different lengths which will look like a triangle. For example, the following program creates a two dimensional array in which the sizes of the second dimension are unequal or the array looks like a triangle :

//Program 5.5 : **To display the elements of a two dimensional matrix in triangular form.**

```
class  TriangularArray
   {
      public  static  void  main(String  args[])
         {
            int  arr[][]  =  new  int[4][];
            int  i;
            for(i=0; i<4; i++)//Creates triangular array
            arr[i]  =  new  int[i+1];
            int  j, k  =  0;
            for(i=0;  i<4;  i++)// Assigns the elements
            for(j=0;  j<i+1;  j++)
            {
                  arr[i][j]  =  k;
                  k++;
            }
            for(i=0;  i<4;  i++)// Display the elements
            {
               for(j=0;  j<i+1;  j++)
               System.out.print(arr[i][j]  +  "  ");
               System.out.println();
            }
         }
   }
```

Output :

```
0
1  2
3  4  5
6  7  8  9
```



CHECK YOUR PROGRESS

Q1. Choose the correct answer from the following :

i) Which of the following statements about arrays are true?

A. Arrays are a group of variables containing values that all have the same type.

B. Elements are located by index or subscript.

C. The length of an array c is determined by the expression c.length();.

D. The zeroth element of array c is specified by c[0].

a) A, C, D. b) A, B, D.

c) C, D. d) A, B, C, D.

ii) Consider the code segment below. Which of the following statements is not true?

int g[];

g = new int[23];

a) The first statement declares an array reference.

b) The second statement creates the array.

c) g is a reference to an array of integers.

d) The value of g[3] is -1.

iii) Which of the following initializer lists would correctly set the elements of array n?

a) int n[] = { 1, 2, 3, 4, 5 };.

b) array n[int] = { 1, 2, 3, 4, 5 };.

c) int n[5] = { 1; 2; 3; 4; 5 };.

d) int n = new int(1, 2, 3, 4, 5);.

iv) Consider the following Java Program :

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

```
        int a[ ] = { 99, 22, 11, 3, 11, 55, 44, 88, 2, -3 };
        int result = 0;
        for ( int i = 0; i < a.length; i++ )
        {
                if ( a[ i ] > 30 )
                result += a[ i ];
        }
          System.out.println( "Result is: " +result );
    }
}
```

The output of this Java program will be:

a) Result is: 280. b) Result is: 154.

c) Result is: 286. d) Result is: 332.

v) Which statement below initializes array items to contain 3 rows and 2 columns?

a) int items[][] = { { 2, 4 }, { 6, 8 }, { 10, 12 } };.

b) int items[][] = { { 2, 6, 10 }, { 4, 8, 12 } };.

c) int items[][] = { 2, 4 }, { 6, 8 }, { 10, 12 };.

d) int items[][] = { 2, 6, 10 }, { 4, 8, 12 };.

vi) 26. For the array in the previous question, what is the value returned by items[1][0]?

a) 4 b) 8 c) 12 d) 6.

vii) Which of the following statements creates a multidimensional array with 3 rows, where the first row contains 1 value, the second row contains 4 items and the final row contains 2 items?

a) int items[][] = { { 1, null, null, null }, { 2, 3, 4, 5}, { 6, 7, null, null } };.

b) int items[][] = { { 1 }, { 2, 3, 4, 5 }, { 6, 7 } };.

c) int items[][] = { { 1 }, { 2, 3, 4, 5 }, { 6, 7}, {});.

d) int items[][] = { { 1 }, { 4 }, { 2 } };.

5.5 STRINGS

Strings are sequence of characters. **I**n Java, strings are objects of the class 'String'. The 'String' and 'StringBuffer' classes are standard class that comes with Java, and they are specifically designed for creating and processing strings. A sequence of characters separated by double quotes are called String constants. Java handles String constants in the same way that other computer languages handle normal strings. Objects of type String are immutable it means that once a String object is created, its contents cannot be altered.

As String is a class, we can create an instance or object and give it a name. For example,

```
String name;
name = new String("KKHSOU");
```

Unlike in other classes, the explicit use of **new** to create an instance or object is *optional* for the String class. So, we can create a new String object in this way :

```
String name;
name = "KKHSOU";
```

or

```
String name = "KKHSOU";
```

Once you have created a **String** object, you can use it any where that a string is allowed. For example, the following statement displays 'KKHSOU' :

```
System.out.println(name);
```

We are already using the ***length*** method which is under the String class to get the length of an array. In case of finding the length of a string also we can use it. For example,

```
int i = name.length();
```

Java defines one operator for **String** objects: **+**. It is used to concatenate two strings. For example, this statement

```
String myString = "I" + " like " + "Java.";
```

results in **myString** containing "I like Java."

The following program demonstrates the above concept :

//Program 5.6 : **To concatenate two strings using the + operator.**

```
class  StringJoin
    {
        public static void main(String args[])
            {
                String str1 = "Krishna Kanta Handique";
                String str2 = "State Open University";
                 // str3 holds the resultant string
                String str3 = str1 + " and " + str2;
                System.out.println("\n First String :" +str1);
                System.out.println("\n Second String :" +str2);
                System.out.println("\n Resultant String :" +str3);
            }
    }
```

Output : First String : Krishna Kanta Handique
Second String : State Open University
Resultant String : Krishna Kanta Handique State Open University

5.5.1 Strings Methods

The String class defines a number of methods that allow us to accomplish a variety of string manipulation tasks. The following table lists some of the commonly used string methods.

Method Call	**Task Performed**
s2=s1.toLowerCase;	Converts the String s1 to all lowercase
s2=s1.toUpperCase;	Converts the String s1 to all Uppercase
s2=s1.replace('x','y');	replaces all appearances of x with y
s2=s1.trim();	removes white spaces at the begining and end of the String s1
s1.equals (s2)	Returns true if s1 equals to s2
s1.equalsIgnoreCase(s2)	Returns true if s1=s2, ignoring the case of Characters
s1.length()	gives the length of s1

Method Call	**Task Performed**
s1.CharAt(n)	gives the nth character of s1
s1.compareTo(s2)	returns negative if s1<s2, positive if s1>s2, and zero if s1 is equal to s2
s1.concat (s2)	concatenates s1 and s2
s1.substring(n)	gives the substring starting from nth character
s1.substring(n,m)	gives the substring starting from nth character upto mth (not including mth)

The following program demonstrates the use of some string methods:

//***Program. 5. 7 :*** A string is given below :

BACHELOR OF COMPUTER APPLICATION FOURTH SEMESTER

Find the following :

a) its length,
b) the character at index 6
c) convert it to lowercase letter
d) a substring from 2 to 15
e) length of the substring
f) the character at index 3

Solution:

```
class TestString
{
     public static void main(String[] args)
     {
         String text = "BACHELOR OF COMPUTER APPLICATION
                        FOURTH SEMESTER";
         System.out.println("\n\nThis string is :" +text);
                        // calculates length of the string
         System.out.println("Length of the String is :" +text.length());
                        // finds the character at index 6 position
         System.out.println("The character at index 6 is :"
                              +text.charAt(6));
```

```
            // Converts the string into lowecase letter
            System.out.println("The string in lowercase letter is :"
                                +text.toLowerCase());
            // Finding the substring of the main string
            String sub = text.substring(2, 15);
            System.out.println("sub=text.substring(2, 15)) :" +sub);
                        // Length of the sub string
            System.out.println("sub.length :" +sub.length());
                        // character at position 3
            System.out.println("sub.charAt(3) :" +sub.charAt(3));
        }
}
```

Output :

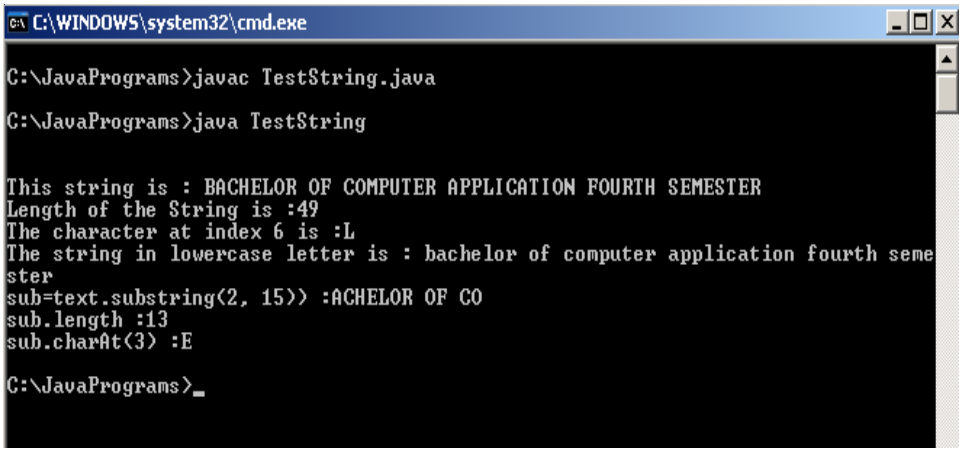


Figure 5.2 : Output of Program 5.7

You can also test two strings for equality by using the method **equals( )**. The general form of this method is **s1.equals (s2)** which means it will return true if s1 equals to s2.

The following program demonstrates the equality of two string.

//Program 5.8 : **To check equality of two strings.**

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

```
        {
            String str1 = "BCA Second Semester";
            String str2 = "BCA Third Semester";
            String str3 = str1;     //str1 string assigns to str3
                  // Finds the length of strings
            System.out.println("\nLength of str1: " + str1.length());
            System.out.println("\nLength of str2: " + str2.length());
            System.out.println("\nLength of str3: " + str3.length());
            if(str1.equals(str2))
                  System.out.println("str1 == str2");
            else
                  System.out.println("str1 != str2");
            if(str1.equals(str3))
                  System.out.println("str1 == str3");
            else
                  System.out.println("str1 != str3");
            }
    }
```

Output :

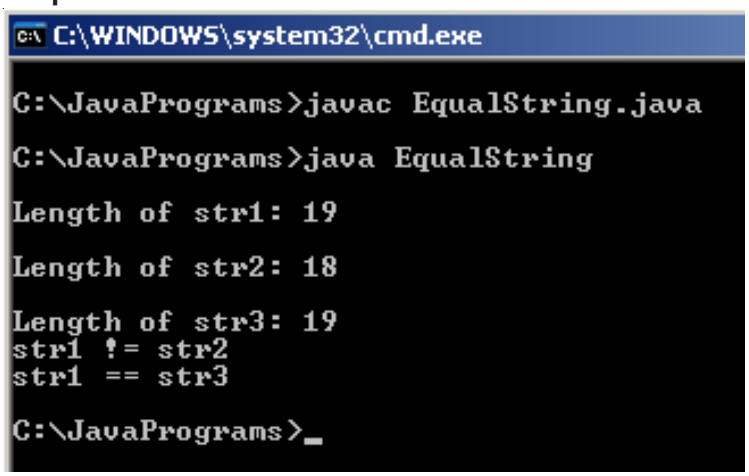


Figure 5.3 : Output of Program 5.8

In the program, you see the equality operator "==" which is used to test whether two references refer to the same object. Here in our expression, if str1 and str2 are both String references, then

the expression str1 == str2 will be true only if str1 and str2 are synonyms. Here in this example, str1 and str2 are not equivalent so it will return false and hence prints the else statement.

String Arrays : We can also create and use arrays for storing strings. The following statement will create an string array named **arraySt** of size 5 to hold 5 string constant.

String arraySt[ ] = new String[5];

We can assign the strings to the **arraySt** element by element using five different statements or more efficiently using a **for** loop.

The following program demonstrates the use of the string array :

//Program 5.9 : Program for string away

```
class arrayStr
{
        static String arraySt[ ] ={"BCA 1st Sem","BCA 2nd Sem","BCA
                    3rd Sem","BCA 4th Sem","BCA
                    5th Sem", "BCA 6th Sem"};
        public static void main(String args[])
        {
            for(int i=0;i<6;i++)
            {
                    System.out.println(arraySt[i]);
            }
        }
    }
```

Output :
BCA 1st Sem
BCA 2nd Sem
BCA 3rd Sem
BCA 4th Sem
BCA 5th Sem
BCA 6th Sem

5.5.2 StringBuffer Class

The string class is one of the most useful classes in Java. But its instances or objects are immutable i.e. they cannot be changed. In the above examples, whenever a string is modified, it has to be done by constructing a new String object, either explicitly or implicitly. Java provides another standard class for defining strings called **StringBuffer**, and its object can be altered directly. Strings that can be changed are often referred to as **mutable strings**.

We will use StringBuffer objects for transforming strings i.e., adding, deleting, or replacing substrings in a string. The following are the commonly used StringBuffer methods :

capacity(): Returns the current capacity of the String buffer.

length(): Returns the length (character count) of this string buffer.

toString(): Converts to a string representing the data in this string buffer

insert(int offset, char c): Inserts the string representation of the char argument into this string buffer.

delete(int start, int end): Removes the characters in a substring of this StringBuffer

replace(int start, int end, String str): Replaces the characters in a substring of this StringBuffer with characters in the specified String.

reverse(): The character sequence contained in this string buffer is replaced by the reverse of the sequence.

append(String str): Appends the string to this string buffer.

setLength(int newLength): Sets the length of this String buffer.

The following program demonstrates the using of String Buffer methods :

//***Program 5.10 :*** Program showing use of string buffer methods

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

```
                    //  Creation of Strings
        StringBuffer strBuf1 = new StringBuffer("KKHSOU");
        StringBuffer strBuf2 = new StringBuffer(40); //With capacity 40
        StringBuffer strBuf3 = new StringBuffer(); //Default Capacity 16
        System.out.println("\n\n strBuf1 : " + strBuf1);
                    // Finds the capacity of StringBuffer
        System.out.println("strBuf1 capacity : " + strBuf1.capacity());
        System.out.println("strBuf2 capacity : " + strBuf2.capacity());
        System.out.println("strBuf3 capacity : " + strBuf3.capacity());
                    // Finds the length of Buffer1
        System.out.println("strBuf1 length : " + strBuf1.length());
                    // Finds the character at position 2
        System.out.println("strBuf1 charAt 2 : " + strBuf1.charAt(2));
                    // Appends a string to strBuf 3
        strBuf3.append("Fourth Semester BCA");
        System.out.println("strBuf3 when appended with a String : "
                            + strBuf3.toString());
        strBuf3.insert(16, 'M'); // insert 'M' at position 16
        System.out.println("strBuf3 when M is inserted at 16 : "
                            + strBuf3.toString());
        strBuf3.delete(11, 's'); // delete the substring from 11 position
        System.out.println("strBuf3 when s is deleted at 11 : "
                            + strBuf3.toString());
        strBuf3.reverse();      // reverse the string constant of strBuf3
        System.out.println("Reversed strBuf3 : " + strBuf 3);
        strBuf2.setLength(12);  //set the length of strBuf 2 to 12
        strBuf2.append("java programs"); // appends the string to strBuf2
        System.out.println("strBuf2 : " + strBuf2);
        }
}
```

Output :

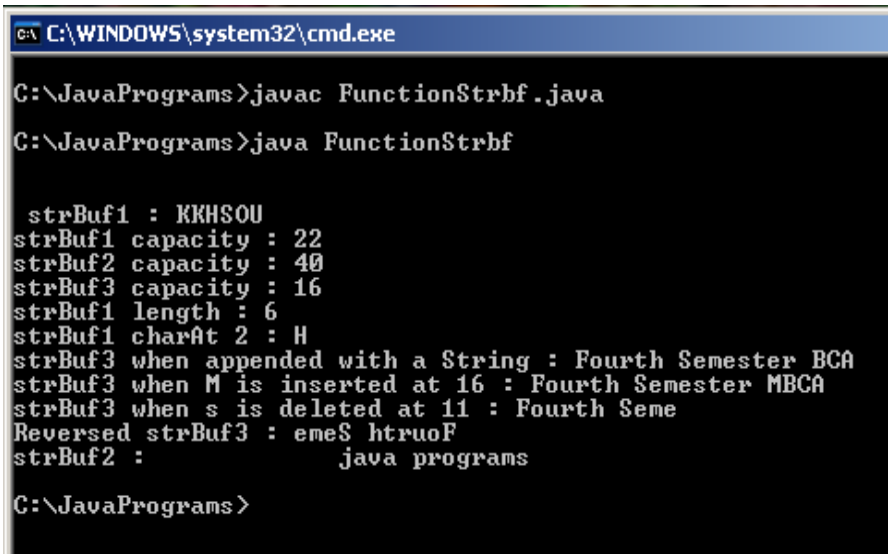


Figure 5.4 : Output of Program 5.10



Note that a vector can be declared without specifying any size explicitly. A vector can accomodate an un-known number of items. Even, when a size is specified, this can be overlooked and a different number of items may be put into the vector. Remember, in contrast, an array must always have its size specified.

5.6 VECTORS

You can consider a **vector** as an expansible array. Before using an array, you need to know the maximum number of elements in order to declare it. If the array is full and it requires to add another element, you need to declare a bigger one, copy all the elements and add that element.

A vector hides all these things. Internally, a vector stores its elements also in an array but it will increase its capacity automatically to ensure all elements be stored. So, use a vector when you don't know the number of elements ahead of time. A vector is a standard class that resides in the **java.util package**.

The key difference between arrays and vectors in Java is that vectors are dynamically-allocated. They are not declared to contain a type of variable; instead, each vector contains a dynamic list of references or pointers to other objects. Since vector stores pointers to the objects, and not the objects themselves, these vector items could be any kind of object. We can add and delete objects from the list as and when required. Always remember we can not directly store simple data types (int, float...etc) in vectors.

Arrays can be easily implemented as vectors. Vectors are created like arrays as follows :

Vector V = new Vector(); //declaring without size

or

Vector V = new Vector(3); // declaring with size

The vector class supports a number of methods that can be used to manipulate the vectors created. Some are listed in table 5.1:

Table 5.1: Vector Class and Methods

Method Call	Task Performed
v.addElement (item)	Add the item specified to the list at the end
v.elementAt(n)	Give the name of the nth object
v.size()	Give the number of objects present
v.removeElement (item)	Removes the specified item from the list
v.removeAllElements()	Removes all the elements in the list
v.copyInto (array)	Copies all items from list to array
v.insertElementAt (item, n)	Inserts the item at n th position.

Wrapper Classes : Vectors can not handle primitive data types like int, float, long, char and double. Primitive data types may be converted into object types by using the wrapper classes contained in the **java.lang** package. Table 5.2 shows the simple data types and their corresponding wrapper class types.

Table 5.2: Wrapper Classes for Converting Simple Types

Simple Type	Wrapper Class
boolean	Boolean
char	Char
double	Double
float	Float
int	Int
long	Long



CHECK YOUR PROGRESS

Q2. Fill in the blank :

i) Character strings are represented by the class _____.

ii) An _____ is a container object that holds a fixed number of values of a single type.

Q3. Consider the following string :

String strTest = “Programming in Java is easy”;

a) What is the value displayed by the expression strTest.length()?

b) What is the value returned by the method call strTest.charAt(12)?

c) Write an expression that refers to the letter J in the string referred to by strTest.

Q4. Show two ways to concatenate the following two strings together to get the string “KKHSOU”:

String str1= “KKH “;

String str2 = “SOU”;

Q5. What is the main difference between String class and StringBuffer class.

Q6. Which package of Java contains Vector Class?



5.7 LET US SUM UP

- An array holds multiple values of the same type, identified through a single variable name.
- Individual elements of an array referred using an integer index value; The index value for an array element is the offset of that element from the first element in the array.

- An array element can be used in the same way as a single variable of the same type.
- A string object stores a fixed character string that cannot be changed. However, you can assign a given string variable to a different String object.
- You can obtain the number of characters stored in a string object by using the *length()* method for the object.
- The *string* class provides methods for joining, searching, and modifying strings – the modifications being achieved by creating a new string object.
- A *StringBuffer* object can store a string of characters that can be modified.
- You can change both the length and the capacity for a StringBuffer object.
- Java does not support the concept of variable arguments to a function. This feature can be achieved in Java through the use of the Vector class contained in *java.util* package.



5.8 FURTHER READINGS

1) Balagurusamy, E. (2007). Programming With Java: A Primer 3E.

2) Schildt, H. (2000). C/C++ Programmer's Reference. McGraw-Hill, Inc..

3) Deitel, P., & Deitel, H. (2011). Java How to program. Prentice Hall Press.



5.9 ANSWERS TO CHECK YOUR PROGRESS

Ans. to Q. No. 1 : i) b; ii) d; iii) a; iv) c; v) a; vi) d; vii) b

Ans. to Q. No. 2 : i) java.lang.String; ii) array

Ans. to Q. No. 3 : a) 26; b) i; c) strTest.charAt(15)

Ans. to Q. No. 4 : str1.concat(str2) and str1 + str2

Ans. to Q. No. 5 : Instance (objects) of the **String** class are immutable : they cannot be changed. Instances of the **StringBuffer** class donot have constraints i.e. they are mutable.

Ans. to Q. No. 6 : **java.util** package.



5.10 MODEL QUESTIONS

Q1. What is an array?

Q2. Write a statement to declare and instantiate an array to hold marks obtained by students in different subjects in a class. Make your own assumptions.

Q3. Write a program to add two matrices.

Q4. Write a program to extract a portion of a character string and print the extracted string.

Q5. What is a vector? How is it different from an array?

Q6. How does a String class differ from a StringBuffer class?

Q7. Write a program which will read a text and count all occurrences of a particular word.

Q8. Write a program to demonstrate the use of arraycopy() function.

*****