St. Francis Institute of Technology, Mumbai-400 103

Department Of Information Technology

A.Y. 2020-2021

Class: SE-ITA/B, Semester: III

Subject: **Java Programming Lab**

Experiment-2: Java Program to implement classes, arrays and strings.

1. **Aim:** Write Java code to demonstrate the following:
   - **a)** Command Line Arguments
   - **b)** Scanner Class and BufferedReader Class
   - c) Write a program that would print the information (name, year of joining, salary, address) of three employees by creating a class named 'Employee'. The output should be as follows:

Name	Year of joining	Address
Robert	1994	64C- WallsStreat
Sam	2000	68D- WallsStreat
John	1999	26B- WallsStreat

   - **d)** Write a java programs to add n strings in a vector array. Input new string and check whether it is present in the vector. If it is present delete it otherwise add it to the vector.
   - e) Write a Java program to illustrate Constructor Chaining.

2. **Prerequisite:** Knowledge of class, arrays and vectors.

3. **Requirements:** Personal Computer (PC), Windows Operating System, Net beans 8.0.

4. **Pre-Experiment Exercise:**
   Theory:

   a. **String Class:**
   The java.lang.String class provides a lot of methods to work on string. By the help of these methods, we can perform operations on string such as trimming, concatenating, converting, comparing, replacing strings etc. Java String is a powerful concept because everything is treated as a string if you submit any form in window based, web based or mobile application.

   b. **String Buffer Class:**
   Java String Buffer class is used to create mutable (modifiable) string. The StringBuffer class in java is same as String class except it is mutable i.e. it can be changed.

   c. **Array:**

   An array is a group of contiguous or related data elements that share a common name. A list of items can be given only one variable name and

such a variable is called an array. Like any other variables, arrays must be declared and created in the computer memory before they are used. Creation of an array involves three steps:

1. Declaring the array
2. Creating memory locations
3. Putting values into the memory locations.

Java allows us to create arrays using 'new' operator only, as shown below:

1. One- Dimensional arrays:
   arrayname=new type[size];
2. Multi- Dimensional arrays:
3. arrayname=new type[row][col];

d. **Vector:**

The Vector class implements a grow able array of objects. Like an array, it contains components that can be accessed using an integer index. However, the size of a Vector can grow or shrink as needed to accommodate adding and removing items after the Vector has been created. Each vector tries to optimize storage management by maintaining a capacity and a capacity increment. The capacity is always at least as large as the vector size; it is usually larger because as components are added to the vector, the vector's storage increases in chunks the size of capacity increment. An application can increase the capacity of a vector before inserting a large number of components; this reduces the amount of incremental reallocation.

5. Laboratory Exercise

A. Procedure

i. Open Net beans for Java.
ii. Open File and Create New Java Project.
iii. Inside the Java Project rename give name to your Java Class.
iv. Click on Finish.
v. Type the Java Code in the opened class.
vi. Save the code by pressing Ctrl+S.
vii. Run the code by pressing Shift+F6.

B. Program code with comments:

Write and execute your program code to achieve the given aim and attach it **with your own comments with neat indentation**.

6. Post-Experiments Exercise

A. Extended Theory:

1. Differentiate between Scanner Class and Buffered Reader Class.
2. Differentiate between String Class and String Buffer Class.
3. Explain any 20 methods of Vector Class with syntax.
4. Differentiate between Array and Vector.

B. Results/Observations/Program output:

Present the program input/output results and comment on the same.

C. Questions/Programs:

1. Write a Java program to implement 15 methods of Vector class.
2. Write a Java program to compare a String to a specified String Buffer.

D. Conclusion:

1. Write what was performed in the experiment/program.
2. What is the significance of experiment/program?
3. Mention few applications of what was studied.

E. References

1. Balguruswamy, "Programming with java A primer", Fifth edition, Tata McGraw Hill Publication.
2. Learn to Master JAVA, from Star EDU solutions , by ScriptDemics.
3. www.programmingsimplified.com
4. www.javatpoint.com

Program 1:

```java
/**
 * Command Line Arguments
 */
public class Exp2_1 {
    public static void main (String args[]){
        for(int i=0;i<args.length;i++) //Loop over the arguments to print them
            System.out.println(args[i]);
    }
}
```

Output:

```
D:\College\JAVA\Experiments\Exp2>javac Exp2_1.java

D:\College\JAVA\Experiments\Exp2>java Exp2_1 1 2 3 4 5
1
2
3
4
5

D:\College\JAVA\Experiments\Exp2>java Exp2_1 Hello World From Java
Hello
World
From
Java

D:\College\JAVA\Experiments\Exp2>java Exp2_1 1.2 2.3 3.4 4.5
1.2
2.3
3.4
4.5

D:\College\JAVA\Experiments\Exp2>
```

Program 2 a:

```java
/**
 * Scanner class Example
 */
import java.util.Scanner;
```

```java
public class Exp2_2_1 {
    public static void main(String args[]) {

        //Declare Scanner class object
        Scanner scanner=new Scanner(System.in);

        System.out.println("Enter Name");
        String name=scanner.nextLine();

        System.out.println("Enter Age");
        int age=scanner.nextInt();

        System.out.println("Enter Height");
        Double height=scanner.nextDouble();

        //Display result
        System.out.println("Name="+name+"\nAge="+age+"\nHeight="+height);
        scanner.close();
        }
}
```

Output:

```
D:\College\JAVA\Experiments\Exp2>javac Exp2_2_1.java

D:\College\JAVA\Experiments\Exp2>java Exp2_2_1
Enter Name
Yash
Enter Age
19
Enter Height
178.5
Name=Yash
Age=19
Height=178.5

D:\College\JAVA\Experiments\Exp2>
```

Program 2 b:

```java
/**
 * Bufferedreader class Example
 */
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.BufferedReader;

public class Exp2_2_2 {
    public static void main(String[] args) throws IOException {

        //Declare InputStreamReader object
        InputStreamReader ir=new InputStreamReader(System.in);
        //Declare BufferedReader object using InputStreamReader object
        BufferedReader br=new BufferedReader(ir);

        System.out.println("Enter Name");
        String name=br.readLine();

        System.out.println("Enter age");
        int age=Integer.parseInt(br.readLine());

        System.out.println("Enter height");
        Double height=Double.parseDouble(br.readLine());

        //Dislaying Result
        System.out.println("Name="+name+"\nAge="+age+"\nHeight="+height);
    }
}
```

Output:

```
D:\College\JAVA\Experiments\Exp2>javac Exp2_2_2.java

D:\College\JAVA\Experiments\Exp2>java Exp2_2_2
Enter Name
Yash
Enter age
19
Enter height
178.5
Name=Yash
Age=19
Height=178.5

D:\College\JAVA\Experiments\Exp2>
```

Program 3:

```
/**
 * Write a program that would print the information (name, year of joining,
 * salary, address) of three employees by creating a class named 'Employee'.
 * The output should be as follows:
 */


class Employee{
  private String name, address;
  private int year, salary;
  //Parameterized constructor definition
  public Employee(String name, int year, int salary, String address){
    this.name = name;
    this.year = year;
    this.salary = salary;
    this.address = address;
  }
  //method to return name
  public String getName(){
```

```
    return name;
  }
  //method to return year
  public int getYear(){
    return year;
  }
  //method to return salary
  public int getSalary(){
    return salary;
  }
  //method to retuen address
  public String getAddress(){
    return address;
  }
}

public class Exp2_3 {
  public static void main(String[] args){

    //Creating objects of Employee class
    Employee e1 = new Employee("Robert", 1994, 500000, "64C- WallsStreet");
    Employee e2 = new Employee("Sam", 2000, 740000, "68d- WallsStreet");
    Employee e3 = new Employee("John", 1999, 600000, "26B- WallsStreet");
    System.out.println("Name\tYear of joining\tSalary\tAddress");

System.out.println(e1.getName()+"\t\t"+e1.getYear()+"\t"+e1.getSalary()+"\t"+e1.g
etAddress());  // printing details of employee 1

System.out.println(e2.getName()+"\t\t"+e2.getYear()+"\t"+e2.getSalary()+"\t"+e2.g
etAddress());  // printing details of employee 2

System.out.println(e3.getName()+"\t\t"+e3.getYear()+"\t"+e3.getSalary()+"\t"+e3.g
etAddress());  // printing details of employee 3

  }
}
```

Output:

```
D:\College\JAVA\Experiments\Exp2>javac Exp2_3.java

D:\College\JAVA\Experiments\Exp2>java Exp2_3
Name    Year of joining Salary  Address
Robert          1994    500000  64C- WallsStreet
Sam             2000    740000  68d- WallsStreet
John            1999    600000  26B- WallsStreet

D:\College\JAVA\Experiments\Exp2>
```

Program 4:

```java
/**
 * Write a java programs to add n strings in a vector array. Input new string and
 * check whether it is present in the vector. If it is present delete it
otherwise add
 * it to the vector.
 */
import java.util.Vector;
import java.util.Scanner;

public class Exp2_4 {
    public static void main(String[] args) {
        int n;
        String str;
        Scanner scanner = new Scanner(System.in);
        //Initialize a string vector
        Vector<String> vect = new Vector<String>();

        System.out.println("Enter Number of strings ypu want to enter: ");
        n = scanner.nextInt();
        //Add n strings to the vector
        for (int i = 0; i < n; i++) {
            System.out.println("Enter a string:");
            str = scanner.next();
            vect.add(str);
        }
```

```
        System.out.println("Vector: " + vect);

        System.out.println("Enter a new string: ");
        String newStr = scanner.next();

        //If new string exists in vector, remove it, else add it
        if (vect.contains(newStr)) {
            vect.remove(newStr);
        }
        else{
            vect.add(newStr);
        }
        //Print the vector
        System.out.println("Vector: " + vect);

        scanner.close();
    }
}
```

Output:

```
D:\College\JAVA\Experiments\Exp2>javac Exp2_4.java

D:\College\JAVA\Experiments\Exp2>java Exp2_4
Enter Number of strings ypu want to enter:
3
Enter a string:
Hello
Enter a string:
World
Enter a string:
Java
Vector: [Hello, World, Java]
Enter a new string:
Java
Vector: [Hello, World]

D:\College\JAVA\Experiments\Exp2>java Exp2_4
Enter Number of strings ypu want to enter:
3
Enter a string:
Hello
Enter a string:
World
Enter a string:
Java
Vector: [Hello, World, Java]
Enter a new string:
Lang
Vector: [Hello, World, Java, Lang]

D:\College\JAVA\Experiments\Exp2>
```

Program 5:

```
// Java program to illustrate Constructor Chaining
// within same class Using this() keyword
class Temp
{
    // default constructor 1
    // default constructor will call another constructor
    // using this keyword from same class
    Temp()
    {
        // calls constructor 2
        this(5);
        System.out.println("The Default constructor");
    }

    // parameterized constructor 2
    Temp(int x)
    {
        // calls constructor 3
        this(5, 15);
        System.out.println(x);
    }

    // parameterized constructor 3
    Temp(int x, int y)
    {
        System.out.println(x * y);
    }

}
public class Exp2_5 {
    public static void main(String args[])
    {
        // invokes default constructor first
        new Temp();
    }
}
```

Output:

```
D:\College\JAVA\Experiments\Exp2>javac Exp2_5.java

D:\College\JAVA\Experiments\Exp2>java Exp2_5
75
5
The Default constructor

D:\College\JAVA\Experiments\Exp2>
```

Questions

Question 1:

```java
//Write a Java program to implement 15 methods of Vector class.
import java.util.Enumeration;
import java.util.Vector;


public class Q1 {
public static void main(String[] args)
{
     Vector v = new Vector();
     System.out.println("Initial Capacity of Vector: "+v.capacity());
        v.add(1);
        v.add(2);
        v.add("mango");
        v.add("apple");
        v.add(3);
        v.add(5);
        v.add(3);

        System.out.println("Modified capacity of Vector: "+v.size());
        System.out.println("First element: " +v.firstElement());
        System.out.println("Last element: " +v.lastElement());
        if(v.contains(new String("mango")))
         System.out.println("Vector contains mango.");
        else
```

```
            System.out.println("Vector doesnt contain mango.");
        Enumeration e=v.elements();
        while(e.hasMoreElements())
        {
           Object o=e.nextElement();
           System.out.println(o+"");

        }


        v.insertElementAt("light",3);
        System.out.println("Index of mango is: "+v.indexOf("mango"));
        System.out.println("Index of first occurence of 3: "+v.indexOf(3,3));
        if(v.isEmpty())
            System.out.println("Vector doesnt contain elements");
        else
            System.out.println("Vector contains elements");
        v.removeElementAt(2);
        System.out.println("Modified vector is: "+v);
        v.setElementAt("litchi",3);
        System.out.println("Modified vector is: "+v);
        System.out.println("Returning sublist from the vector: "+v.subList(3,5));
        Vector<String> copy = (Vector<String>) v.clone();
        System.out.println("Cloned vector: "+copy);
        System.out.println("Hash Code value of Vector is: "+v.hashCode());

}
}
```

Output:

```
D:\College\JAVA\Experiments\Exp2>javac Q1.java
Note: Q1.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.

D:\College\JAVA\Experiments\Exp2>java Q1
Initial Capacity of Vector: 10
Modified capacity of Vector: 7
First element: 1
Last element: 3
Vector contains mango.
1
2
mango
apple
3
5
3
Index of mango is: 2
Index of first occurence of 3: 5
Vector contains elements
Modified vector is: [1, 2, light, apple, 3, 5, 3]
Modified vector is: [1, 2, light, litchi, 3, 5, 3]
Returning sublist from the vector: [litchi, 3]
Cloned vector: [1, 2, light, litchi, 3, 5, 3]
Hash Code value of Vector is: -1739718264

D:\College\JAVA\Experiments\Exp2>
```

Question 2:

```java
//Write a Java program to compare a String to a specified String Buffer.
import java.util.Scanner;
public class Q2 {
    public static void main(String[] args)
    {
```

```
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter the First string: ");
        String str1 = scanner.nextLine();
        System.out.println("Enter the Second string: ");
        String str2 = scanner.nextLine();

        StringBuffer strbuf = new StringBuffer(str1);

        System.out.println("Comparing "+str1+" and "+strbuf+": " +
str1.contentEquals(strbuf));

        System.out.println("Comparing "+str2+" and "+strbuf+": " +
str2.contentEquals(strbuf));

        scanner.close();
    }
}
```

Output:

```
D:\College\JAVA\Experiments\Exp2>javac Q2.java

D:\College\JAVA\Experiments\Exp2>java Q2
Enter the First string:
hello word from java
Enter the Second string:
Hello World From Java
Comparing hello word from java and hello word from java: true
Comparing Hello World From Java and hello word from java: false

D:\College\JAVA\Experiments\Exp2>
```

Page No :
Date :

Yash Mahajan OA SE IT B

6) Post - Experiment Exercise

A) Extended Theory

1) Differentiate between scanner class & buffered reader class.

Scanner	Buffered Reader
1) Scanner is not synchronous in nature & should be used only in single-threaded case.	1) Buffered Reader is synchronous in nature during multi threading enviroment, buffered reader should be used.
2) Scanner is has little buffer of 1KB charbuffer.	2) Buffered Reader has large buffer of 8KB byte buffer.
3) Scanner is bit slower as it needs to parse data as well.	3) Buffered Reader is faster than scanner as only reads a character ~~structure~~ stream
4) Scanner has methods like nextInt(), nextShort() etc	4) Buffered reader has methods like parseInt(), parseShort() etc.

Page No :
Date :

Yash Mahajan 04 SE I T B

2) Differentiate between string buffer class and string class

String	String Buffer
① The length of the string object is fixed	① The length of string buffer can be increased.
② String object is im-mutable	② String buffer is mutable
③ It is slower during concatenation	③ It is faster during concatenation.
④ Memory is consumed more	④ less memory consumed.

3) Explain 20 methods of vector class with syntax

→ Vector implements a dynamic array but with two differences.

(i) vector is synchronised

(ii) vector contains many legacy methods that are not part of collections framework

1) Syntax :- void add(int index, Object element)
~~Explain~~ Inserts the specified element at specified position in vector.

2) boolean add(Object a) →
Append the specified element at the end of the ~~list~~ vector.

Yash Mahajan SE IT B 04

3) int capacity () :-
This method returns the current capacity of the vector.

4) void clear () :-
Removes all the elements from the vector.

5) Object clone () :-
Returns a clone of the vector.

6) boolean contains (Object elem)
Tests if the specified object is a component in the vector.

7) void copyInto (Object [], array)
copies the components of this vector into the specified array.

8) Object elementAt (int index)
- Returns the component at specified index

9) Enumeration elements ()
Returns an enumeration of the components of this vector.

10) void ensureCapacity (int minCapacity)
Increases the capacity of vector.

Page No :
Date :

Yash Mahajan SE IT B 04

11) boolean equals(Object o)
- compares the specified object with this vector for equality.

12) Object firstElement()
Returns the first element of vector.

13) int hashCode()
Returns the hash code value for this vector.

14) Object[] toArray()
Returns an array containing all the elements into the vector in correct order.

15) String toString()
Returns a String representation of the vector, containing the string representation of each element.

16) void trimToSize()
Trims the ~~size~~ capacity of the vector to be the vector's current size.

17) int size()
returns the number of components in this ~~array~~ vector.

18) void removeAllElements()
removes all elements in vector, sets it's size to 0.

Yash Mahajan SE IT B 04

19) protected void removeRange(int fromIndex, int toIndex)
removes the elements in the vector from the from index to the toindex.

20) Object Set (int index, Object element).
Replaces the element of specified index with specified element in vector.

Q4) Difference between vector and Array

Vector	Array
1) Vector is a sequential container to store elements and not index based.	1) Array stores fixed size sequential collection of elements of the same type & index based.
2) Vector is dynamic in nature so increases size with insertion of elements	2) An array is fixed size once initialized can't be resized.
3) Vector occupies more memory	3) Array is memory efficient data structure.

Page No :
Date :

Yash Mahajan SE IT B 04

D. Conclusion :-

In this experiment we have written Java programs to implement command line arguments, Scanner class, Buffered reader class, vectors and constructor chaining.

Scanner class and Buffered Reader are two important classes that help us to enter our input in the program. Vectors are one of the most important datatype as they don't have a fixed capacity.

To take input from the user the Scanner class and Buffered Reader class are essential. Vectors can store various type of data. String class is used to create and manipulate strings.