Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

Academic Year: 2022-2023

JAVA

EXPERIMENTS

Name:	Prerna Sunil Jadhav
Sap ID:	60004220127
Branch:	Computer Engineering
Semester:	III (DSE)
Course Code:	DJ19CEL306
Course:	Programming Laboratory – I (Java)
Experiments:	1 to 5

Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

Academic Year: 2022-2023

Name – Prerna Sunil Jadhav **SAP ID - 60004220127**

Experiment No - 01

AIM: TO IMPLEMENT JAVA PROGRAM STRUCTURES & SIMPLE PROGRAMS

THEORY:

- Java is a programming language and a platform. Java is a high level, robust, object-oriented, and secure programming language.
- main() method is essential for all Java programs because the execution of all Java programs starts from the main() method. In other words, it is an entry point of the class.
- A typical structure of a Java program contains the following elements:

PROGRAM 1: Write a Program to display "Hello" Message on screen.

CODE:

Code1_HelloWorld.java

```
Exp1 > Code1_HelloWorld.java > Code1_HelloWorld
1   class Code1_HelloWorld{
        Run | Debug
2       public static void main(String[] args) {                    //main function
3           System.out.println(x: "Prerna Jadhav - 60004220127\n\n");
4           System.out.println(x: "Hello");                         //prints Hello
5       }
6   }
```

THEORY:

In Java, System.out.println() is used to print a statement which has been passed in its argument. There are 2 printing statements in Java, the first being System.out.print() which prints the argument passed through it on the same line and the second being System.out.println() which is similar to System.out.print() method except that it moves the cursor to the next line after printing the result.

OUTPUT:

```
Prerna Jadhav - 60004220127


Hello
```

PROGRAM 1: Write a Java program that reads a positive integer from command line and count the number of digits the number (less than ten billion) has.

CODE:

Code2_CountDigits.java

```java
import java.util.Scanner;       //importing Scanner class

public class Code2_CountDigits {
    Run | Debug
    public static void main(String[] args) {
        System.out.println(x: "Prerna Jadhav - 60004220127\n\n");

        Scanner sc = new Scanner(System.in);    //Creating Scanner object to take input
        System.out.print(s: "Enter any Number: ");
        int num = sc.nextInt();                 //taking input from user
        int temp = num;
        int count = 0;
        while (num!=0){                         //looping until num becomes 0
            count++;
            num /= 10;
        }
        System.out.println("Count of "+temp+" is "+count);   //displaying sum
        sc.close();                             //closing the scanner
    }
}
```

THEORY:

The while loop in Java is a control flow statement that allows code to be executed repeatedly based on a given boolean condition. The loop goes on until the boolean condition turns false. When the number of iterations in not known to the user, they can use the while loop

OUTPUT:

```
Prerna Jadhav - 60004220127

Enter any Number: 768009567
Count of 768009567 is 9
```

CONCLUSION:

This Experiment covered a lot of territory relating to the fundamentals of the Java programming language. It started by learning about the structure of Java program code then learned about the different data types used in Java. From there it was learnt how Java evaluates expressions. Finally, we also coded a program about the different operators used to carry out operations such as assignment, addition, subtraction, modulus, division etc.

Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

Academic Year: 2022-2023

Name – Prerna Sunil Jadhav **SAP ID - 60004220127**

Experiment No - 02

AIM: TO IMPLEMENT JAVA CONTROL STATEMENTS AND LOOPS

THEORY:

- Java compiler executes the code from top to bottom. The statements in the code are executed according to the order in which they appear.
- Java provides statements that can be used to control the flow of Java code. Such statements are called control flow statements.
- It is one of the fundamental features of Java, which provides a smooth flow of program.
- Java provides three types of control flow statements.
  - Decision Making statements
    - ✓ if statements
    - ✓ switch statement
  - Loop statements
    - ✓ do while loop
    - ✓ while loop
    - ✓ for loop
    - ✓ for-each loop
  - Jump statements
    - ✓ break statement
    - ✓ continue statement

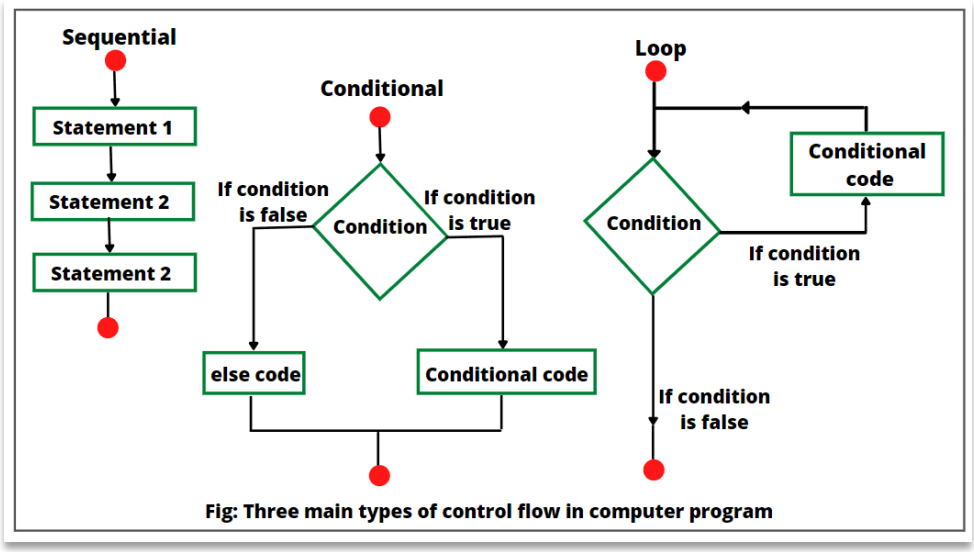


Fig: Three main types of control flow in computer program

PROGRAM 1: Write A Program to find roots of a Quadratic equation. Take care of imaginary values.

THEORY:

The standard form of a quadratic equation is: $ax^2 + bx + c = 0$
Here, a, b, and c are real numbers and a can't be equal to 0.
We can calculate the root of a quadratic by using the formula: $x = (-b \pm \sqrt{(b^2-4ac)}) / (2a)$
The ± sign indicates that there will be two roots:

- ✓ root1 = $(-b + \sqrt{(b^2-4ac)}) / (2a)$
- ✓ root1 = $(-b - \sqrt{(b^2-4ac)}) / (2a)$

The term b^2-4ac is known as the determinant of a quadratic equation. It specifies the nature of roots. That is,

- ✓ if determinant > 0, roots are real and different
- ✓ if determinant == 0, roots are real and equal
- ✓ if determinant < 0, roots are complex and different

Academic Year: 2022-2023

CODE:

Code1_Quadratic.java

```java
package Exp2;

public class Code1_Quadratic {
    Run | Debug
    public static void main(String[] args) {
        System.out.println(x: "Prerna Jadhav - 60004220127\n");

        double a = 2.3, b = 4, c = 5.6; // value a, b, and c
        double root1, root2;

        double determinant = b * b - 4 * a * c; // calculate the determinant (b2 - 4ac)

        if (determinant > 0) {  // check if determinant is greater than 0
            // two real and distinct roots
            root1 = (-b + Math.sqrt(determinant)) / (2 * a);
            root2 = (-b - Math.sqrt(determinant)) / (2 * a);
            System.out.format(format: "root1 = %.2f and root2 = %.2f", root1, root2);
        }

        else if (determinant == 0) {  // check if determinant is equal to 0
            // two real and equal roots and determinant is equal to 0
            root1 = root2 = -b / (2 * a);
            System.out.format(format: "root1 = root2 = %.2f;", root1);
        }

        else { // if determinant is less than zero
            // roots are complex number and distinct
            double real = -b / (2 * a);
            double imaginary = Math.sqrt(-determinant) / (2 * a);
            System.out.format(format: "root1 = %.2f+%.2fi", real, imaginary);
            System.out.format(format: "\nroot2 = %.2f-%.2fi", real, imaginary);
        }

    }
}
```

OUTPUT:

```
Prerna Jadhav - 60004220127

root1 = -0.87+1.30i
root2 = -0.87-1.30i
```

PROGRAM 2: Write a menu driven program using switch case to perform mathematical operations.

CODE:

Code2_SwitchCase.java

```java
import java.util.Scanner;
public class Code2_SwitchCase {
    public static void main(String[] args) {
        System.out.println("Prerna Jadhav - 60004220127");
        int operator;
        Double number1, number2;
        Scanner input = new Scanner(System.in); // create an object of Scanner class
        System.out.println("********MENU********\n1.Addition\n2.Subtraction\n3.Multiplication\n4.Division\n5.Exit\n");
        System.out.print("Enter your Choice: ");
        operator = input.nextInt();
        System.out.println("Enter first number");   // ask users to enter numbers
        number1 = input.nextDouble();
        System.out.println("Enter second number");
        number2 = input.nextDouble();
        switch (operator) {
            case 1:     // performs addition between numbers
                System.out.println(number1 + " + " + number2 + " = " + (number1 + number2));
                break;
            case 2:     // performs subtraction between numbers
                System.out.println(number1 + " - " + number2 + " = " + (number1 - number2));
                break;
            case 3:     // performs multiplication between numbers
                System.out.println(number1 + " * " + number2 + " = " + (number1 * number2));
                break;
            case 4:     // performs division between numbers
                System.out.println(number1 + " / " + number2 + " = " + (number1 / number2));
                break;
            case 5:     //To exit Code
                System.exit(0);
            default:    //if invalid choice is entered
                System.out.println("Invalid operator!");
                break;
            }
        input.close();
    }
}
```

OUTPUT:

```
Prerna Jadhav - 60004220127
********MENU********
1.Addition
2.Subtraction
3.Multiplication
4.Division
5.Exit

Enter your Choice: 3
Enter first number
324
Enter second number
901
324.0 * 901.0 = 291924.0
```

```
Prerna Jadhav - 60004220127
********MENU********
1.Addition
2.Subtraction
3.Multiplication
4.Division
5.Exit

Enter your Choice: 4
Enter first number
4329
Enter second number
90
4329.0 / 90.0 = 48.1
```

```
Prerna Jadhav - 60004220127
********MENU********
1.Addition
2.Subtraction
3.Multiplication
4.Division
5.Exit

Enter your Choice: 1
Enter first number
34902.32
Enter second number
2390.008
34902.32 + 2390.008 = 37292.328
```

THEORY:

Here, we have used the Scanner class to take 3 inputs from the user.
operator - specifies the operation to be performed
number1/number2 - operands to perform operation on
Since the operator matches the case 3, so the corresponding codes are executed.

```
System.out.println(number + " * " + number2 + " = " +( number1 * number2));
break;
```

These statements compute the product of two numbers and print the output. Finally, the break statement ends the switch statement.
Similarly, for different operators, different cases are executed.

PROGRAM 3: Write A Program to display odd numbers from given range/ prime numbers from given range.

CODE

Code3_OddNPrime.java

```java
package Exp2;
import java.util.Scanner;
public class Code3_OddNPrime {
    Run | Debug
    public static void main(String[] args) {
        System.out.println(x: "Prerna Sunil Jadhav - 60004220127");
        Scanner sc = new Scanner(System.in);
        System.out.print(s: "Enter a range: ");
        int range = sc.nextInt();
        System.out.print(s: "1.Odd numbers\n2.Prime numbers\nwhat do want to see?: ");
        int operation = sc.nextInt();
        switch(operation){
            case 1:
                OddNumbers(range);
                break;
            case 2:
                PrimeNumbers(range);
                break;
            default:
                System.out.println(x: "Invalid Choice");
        }
        sc.close();
    }
    private static void OddNumbers(int range) {
        System.out.print("Odd Numbers between 1 to "+range+" are ");
        for (int i = 0; i<=range; i++){
            if (i%2!=0)
                System.out.print(i + " , ");
        }
    }
```

Academic Year: 2022-2023

Code3_OddNPrime.java

```
Exp2 > Code3_OddNPrime.java > ...
30      private static void PrimeNumbers(int range) {
31          System.out.print("Prime Numbers between 1 to "+range+" are ");
32          for (int i = 2; i<=range; i++){
33              if (i == 1 || i == 0)
34                  continue;
35              int flag = 1;
36              for(int j = 2; j<=i/2; j++){
37                  if (i%j==0){
38                      flag=0;
39                      break;
40                  }
41              }
42              if(flag==1)
43                  System.out.print(i + " , ");
44          }
45      }
46  }
47
```

OUTPUT:

```
Prerna Sunil Jadhav - 60004220127
Enter a range: 120
1.Odd numbers
2.Prime numbers
what do want to see?: 1
Odd Numbers between 1 to 120 are 1 , 3 , 5 , 7 , 9 , 11 , 13 , 15 , 17 , 19 , 21 , 23 , 25 , 27 , 29 , 31 , 33 , 35 , 37 , 39 , 41 , 43
, 45 , 47 , 49 , 51 , 53 , 55 , 57 , 59 , 61 , 63 , 65 , 67 , 69 , 71 , 73 , 75 , 77 , 79 , 81 , 83 , 85 , 87 , 89 , 91 , 93 , 95 , 97 ,
 99 , 101 , 103 , 105 , 107 , 109 , 111 , 113 , 115 , 117 , 119 ,
```

```
Prerna Sunil Jadhav - 60004220127
Enter a range: 150
1.Odd numbers
2.Prime numbers
what do want to see?: 2
Prime Numbers between 1 to 150 are 2 , 3 , 5 , 7 , 11 , 13 , 17 , 19 , 23 , 29 , 31 , 37 , 41 , 43 , 47 , 53 , 59 , 61 , 67 , 71 , 73 ,
79 , 83 , 89 , 97 , 101 , 103 , 107 , 109 , 113 , 127 , 131 , 137 , 139 , 149 ,
```

THEORY:

For Odd:

- ✓ Firstly, consider the given number N as input.
- ✓ Then apply a for loop in order to iterate the numbers from 1 to N.
- ✓ At last, check if each number is a odd number using the modulus and if it's a odd number then print it

For Prime:

- ✓ Firstly, consider the given number N as input.
- ✓ Then apply a for loop in order to iterate the numbers from 1 to N.
- ✓ At last, check if each number is a prime number by checking if that number is divisible by any other number other than 1 and itself and if it's a prime number then print it

PROGRAM4: Write A Program to display default value of primitive data types

CODE

Code4_PrimitiveDataTypes.java

```java
package Exp2;

public class Code4_PrimitiveDataTypes {
    static boolean val1;
    static double val2;
    static float val3;
    static int val4;
    static long val5;
    static String val6;
    Run | Debug
    public static void main(String[] args) {
        System.out.println(x: "Prerna Sunil Jadhav - 60004220127\n");
        System.out.println("Default value of Boolean = " + val1);
        System.out.println("Default value of Double = " + val2);
        System.out.println("Default value of Float = " + val3);
        System.out.println("Default value of Integer = " + val4);
        System.out.println("Default value of Long = " + val5);
        System.out.println("Default value of String = " + val6);
    }
}
```

OUTPUT:

```
Prerna Sunil Jadhav - 60004220127

Default value of Boolean = false
Default value of Double = 0.0
Default value of Float = 0.0
Default value of Integer = 0
Default value of Long = 0
Default value of String = null
```

THEORY:

Primitive types are the Java data types used for data manipulation, for example, int, char, float, double, boolean, etc.

- ✓ byte: An 8-bit signed two's complement integer (128 – 127)
- ✓ short: A 16-bit signed two's complement integer (-32768 – 32767)
- ✓ int: A 32-bit signed two's complement integer (-2,147,483,648 - 2,147,483,647)
- ✓ long: A 64-bit two's complement integer (-9,223,372,036,854,775,808 - 9,223,372,036,854,775,807)
- ✓ char: A single 16-bit Unicode character. ('\u0000' (or 0) - '\uffff')
- ✓ float: A single-precision 32-bit IEEE 754 floating point (1.4E-45 - 3.4028235E38)
- ✓ double: A double-precision 64-bit IEEE 754 floating point (4.9E-324 - 1.7976931348623157E308)
- ✓ boolean: Possible values, TRUE and FALSE.

PROGRAM5A: Write A Program to display the following patterns:

```
1
2       1
1       2       3
4       3       2       1
1       2       3       4       5
6       5       4       3       2       1
1       2       3       4       5       6       7
```

CODE

Code5a_Pattern1.java

```java
package Exp2;
public class Code5a_Pattern1 {
    Run | Debug
    public static void main(String[] args) {
        System.out.println(x: "Prerna Sunil Jadhav - 60004220127");
        int n = 7;
        for (int i = 1; i<=n; i++){
            if (i%2==0){    //if is even row then reverse
                for (int j = i; j>=1; j--){
                    System.out.print(j + "\t");
                }
            }
            else{
                for (int j = 1; j<=i; j++){
                    System.out.print(j + "\t");
                }
            }
            System.out.println();
        }
    }
}
```

OUTPUT:

```
Prerna Sunil Jadhav - 60004220127
1
2       1
1       2       3
4       3       2       1
1       2       3       4       5
6       5       4       3       2       1
1       2       3       4       5       6       7
```

THEORY:

Here we started a loop till n, here its 7 And will check for every iteration of n if its even if its even reverse the order to print else print sequentially.

PROGRAM5B: Write A Program to display the following patterns:

```
    A
  C B
F E D
J I K G
```

CODE:

Code5b_Pattern2.java

```java
package Exp2;

public class Code5b_Pattern2 {
    Run | Debug
    public static void main(String[] args) {
        System.out.println(x: "Prerna Sunil Jadhav - 60004220127");
        int n=4;
        char A = 64;
        for (int i = 0; i<n; i++){
            //spaces
            for (int s = n-(i+1); s>0; s--){
                System.out.print(s: "\t");
            }

            A+=(i+1);
            char temp = A;
            for(int j = i+1; j>=1; j--){
                System.out.print(temp+"\t");
                temp-=1;
            }
            System.out.println();
        }
    }
}
```

OUTPUT:

```
Prerna Sunil Jadhav - 60004220127
                A
            C   B
        F   E   D
J       I   H   G
```

THEORY:

Here we Initially assume the character to be printed 64 i.e., character before A and will add 1, 2, 3 ... in our Ascii value as our row increments and print the sequence in reverse by decrementing the duplicate value by 1 in inner for loop.

CONCLUSION:

Primitive data types are the building blocks of data manipulation.
For statement consumes the initialization, condition, and increment/decrement in one line thereby providing a shorter, easy to debug
structure of looping.
If a loop exists inside the body of another loop, it's called a nested loop.
That is why nested loops are also called as “loop inside loop“.
The Java switch statement only works with:

- ✓ Primitive data types: byte, short, char, and int
- ✓ Enumerated types
- ✓ String Class
- ✓ Wrapper Classes: Character, Byte, Short, and Integer.

Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

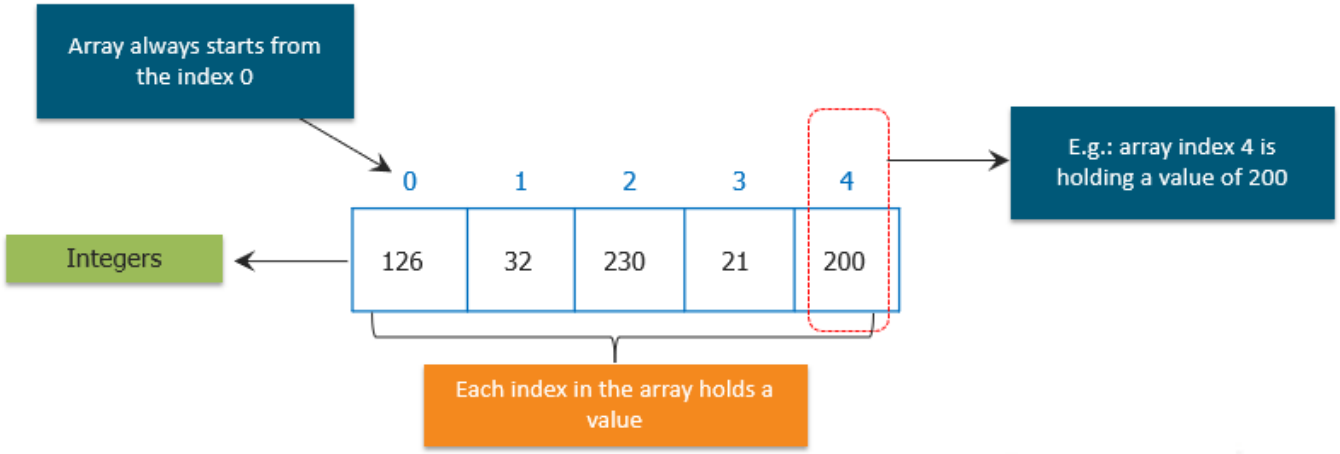
Academic Year: 2022-2023

Name – Prerna Sunil Jadhav **SAP ID - 60004220127**

Experiment No - 03

AIM: TO IMPLEMENT ARRAYS

THEORY:



An array is a collection of similar types of data.

For example, if we want to store the names of 100 people then we can create an array of the string type that can store 100 names.

```
String[] array = new String[100];
```

Here, the above array cannot store more than 100 names. The number of values in a Java array is always fixed.
In Java, here is how we can declare an array.

```
dataType[] arrayName;
```

datatype - it can be primitive data types like int, char, double, byte, etc. or Java objects
arrayName - it is an identifier

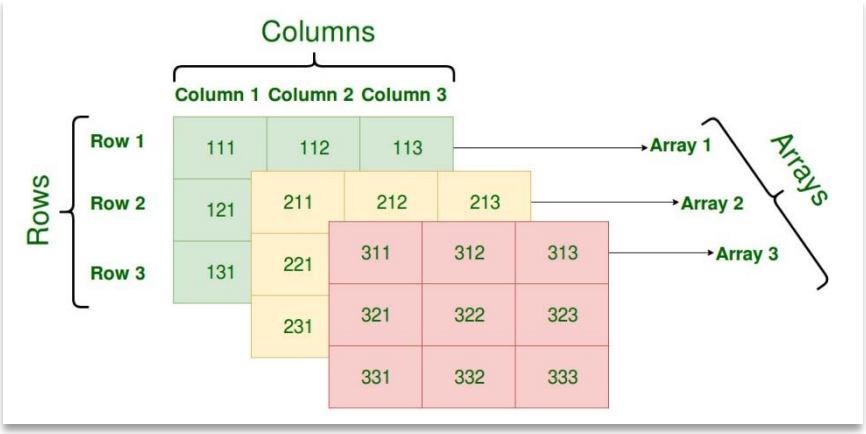
For example,

```
double[] data;
```

Here, data is an array that can hold values of type double.

A multidimensional array is an array of arrays. Each element of a multidimensional array is an array itself.

```
int[][] a = new int[3][4];
```



PROGRAM 1: Write A Program to find whether the entered 4 digit number is vampire or not. Combination of digits from this number forms 2 digit number. When they are multiplied by each other we get the original number. (1260=21*60, 1395=15*93, 1530=30*51)

CODE:

Code1_Vampire.java

Exp3 > Code1_Vampire.java > Code1_Vampire > main(String[])

```
1   package Exp3;
2
3   import java.util.*;
4
5   class Code1_Vampire
6   {
        Run | Debug
7       public static void main(String args[])
8       {
9           System.out.println(x: "Prerna Sunil Jadhav - 60004220127");
10          Scanner sc=new Scanner(System.in);
11          int v=0;
12          long n,i,c=0,f=0,j=0,x,k,p,c1,c2,b[]=new long[100];
13          System.out.println(x: "Enter a number:");
14          n=sc.nextLong();
15          sc.close();
16          //Counting the number of digits
17          for(i=n;i>0;i/=10)
18              c++;
19          if(c%2!=0)
20              f=1;
21          else
22          {
23              for(i=(int)Math.pow(a: 10,c/2-1); i<(int)Math.pow(a: 10,c/2);  i++)
24              {
25                  if(n%i==0)  //i is one factor
26                  {
27                      j=n/i;   //j is another factor
28                      if(!(j>=Math.pow(a: 10,c/2-1) && j<Math.pow(a: 10,c/2)))
29                          f=1;
30
```

Code1_Vampire.java

Exp3 > Code1_Vampire.java > Code1_Vampire > main(String[])

```
                    x=i*(int)Math.pow(a: 10,c/2)+j;

                    if(i%10==0 && j%10==0)
                    {
                        f=1;
                        break;
                    }
                    else //Check all digits are distinct or not
                    {
                        f=0;
                        for(k=n;k>0;k/=10)
                        {
                            c1=c2=0;
                            for(p=n;p>0;p/=10)
                            {
                                if(k%10==p%10)
                                    c1++;
                            }
                            for(p=x;p>0;p/=10)
                            {
                                if(k%10==p%10)
                                    c2++;
                            }
                            if(c1!=c2)
                            {
                                f=1;
                                break;
                             }
                        }
                        if(f==0)
                        {
                            int fl=0;
                            for(int ii=0;ii<v;ii++)
                                if(b[ii]==i || b[ii]==j)
                                    fl=1;
                            if(fl==0)
                                b[v++]=i;

                         }
                    }
                }
            }

        }
```

```
        if(v==0)
            System.out.println(x: "Not a vampire number");
        else
        {
            System.out.println(x: "Vampire number fangs are: ");
            for(int ii=0;ii<v;ii++)
                System.out.println(b[ii]+"\t"+(n/b[ii]));
        }
    }
}
```

OUTPUT:

```
Prerna Sunil Jadhav - 60004220127
Enter a number:
1260
Vampire number fangs are:
21      60
```

```
Prerna Sunil Jadhav - 60004220127
Enter a number:
1593
Not a vampire number
```

PROGRAM 2: WAP to display the following using irregular arrays

2 3
4 5 6

CODE:

Code2_IrregularArray.java

```
Exp3 > Code2_IrregularArray.java > ...
package Exp3;

public class Code2_IrregularArray {
    Run | Debug
    public static void main(String[] args) {
        System.out.println(x: "Prerna Sunil Jadhav - 60004220127");
        // Declaring 2-D array with 2 rows
        int arr[][] = new int[2][];
        arr[0] = new int[2];    // First row has 3 columns
        arr[1] = new int[3];    // Second row has 2 columns
        int count = 2;
        for (int i = 0; i < arr.length; i++)
            for (int j = 0; j < arr[i].length; j++)
                arr[i][j] = count++;                    // Initializing array
        // Displaying the values of 2D Jagged array
        for (int i = 0; i < arr.length; i++) {
            for (int j = 0; j < arr[i].length; j++)
                System.out.print(arr[i][j] + " ");
            System.out.println();
        }
    }
}
```

OUTPUT:

```
Prerna Sunil Jadhav - 60004220127
2 3
4 5 6
```

PROGRAM3: WAP that queries a user for no. of rows & columns representing students & their marks. Reads data row by row & displays the data in the tabular form along with the row & columns & grand total.

CODE:

Code3_2DArray.java

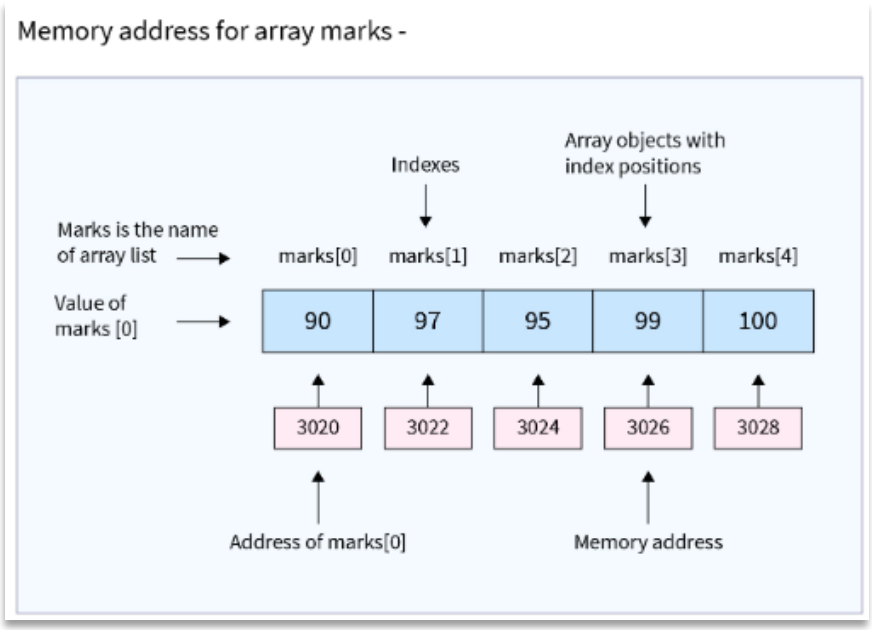
Exp3 > Code3_2DArray.java > ...

```java
package Exp3;
import java.util.Scanner;
public class Code3_2DArray {
    Run | Debug
    public static void main(String[] args) {
        System.out.println(x: "Prerna Sunil Jadhav - 60004220127");
        Scanner sc = new Scanner(System.in);
        System.out.print(s: "Enter Number of Students: ");
        int row = sc.nextInt();
        System.out.print(s: "Enter Number of Subjects: ");
        int col = sc.nextInt();
        int student[][] = new int[row][col];
        for (int i = 0; i<row; i++){
            System.out.println("Enter Marks for Student "+(i+1)+" :");
            for (int j = 0; j<col; j++){
                System.out.print("Enter Marks for Subject "+(j+1)+" : ");
                student[i][j] = sc.nextInt();
            }
        }
        for (int i = 0; i<row; i++){
            int sum = 0;
            for (int j = 0; j<col; j++)
                System.out.print(student[i][j]+"\t");
            for (int k = 0; k<col; k++)
                sum += student[i][k];
            System.out.println(" | "+sum);
        }
        System.out.println(x: "----------------------------");
        int total =0;
        for (int i = 0; i<col; i++){
            int sum=0;
            for(int j = 0; j<row; j++)
                sum += student[j][i];
            System.out.print(sum + "\t");
            total += sum;
        }
        System.out.println(" | "+total);
        sc.close();
    }
}
```

OUTPUT:

```
Prerna Sunil Jadhav - 60004220127
Enter Number of Students: 2
Enter Number of Subjects: 3
Enter Marks for Student 1 :
Enter Marks for Subject 1 : 1
Enter Marks for Subject 2 : 3
Enter Marks for Subject 3 : 6
Enter Marks for Student 2 :
Enter Marks for Subject 1 : 7
Enter Marks for Subject 2 : 9
Enter Marks for Subject 3 : 8
1       3       6       | 10
7       9       8       | 24
---------------------------
8       12      14      | 34
```

CONCLUSION:



- Array indices always start from 0. That is, the first element of an array is at index 0.
- If the size of an array is n, then the last element of the array will be at index n-1.
- Array in Java is a non-primitive data type used to store multiple values of the same data type.
- Elements of the array can be accessed by the index ranging from 0 to array length - 1.
- We can use for loop and for-each loop for looping through all the array elements.
- There are two arrays in Java: Single and Multi-dimensional, where single has one dimension, and multi includes 2D, 3D, and nD dimensions.
- Likewise primitive data types, we can also implement an array of objects in Java.

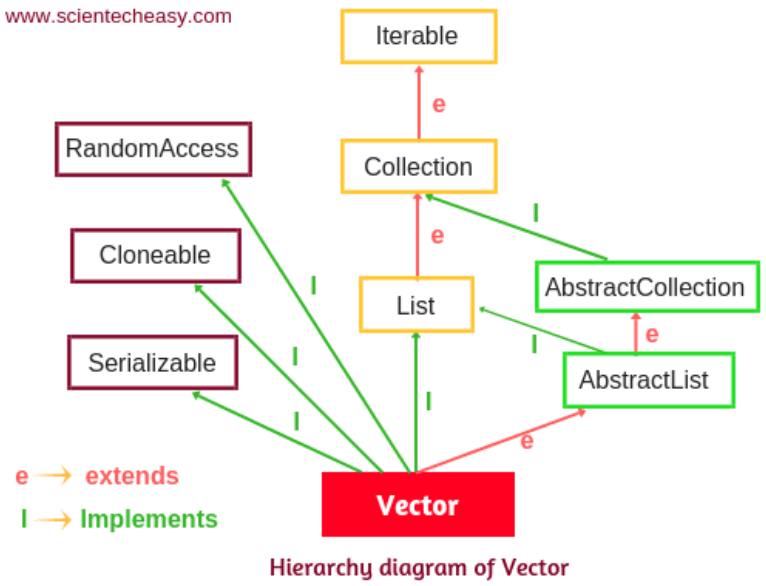
Name – Prerna Sunil Jadhav **SAP ID - 60004220127**

Experiment No - 04

AIM: TO IMPLEMENT VECTORS

THEORY:

- Vector is a data structure that is used to store a collection of elements. Elements can be of all primitive types like int, float, Object etc. Vectors are dynamic in nature and accordingly grow or shrink as per the requirement.
- Vector class is found in the java.util package.
- Vector class is a child class of the AbstractList class and implements the List interface. Therefore we can use all the methods of the List interface.
- Vectors are known to give ConcurrentModificationException when accessed concurrently at the time of modification.
- When a Vector is created, it has a certain capacity to store elements that can be defined initially. This capacity is dynamic in nature and can be increased or decreased.
- By definition, Vectors are synchronized, which implies that at a time, only one thread is able to access the code while other threads have to wait. Due to this, Vectors are slower in performance as they acquire a lock on a thread.



Hierarchy diagram of Vector

- Declaration of Vector in Java

Syntax:

```
public class Vector<E> extends AbstractList<E> implements List<E>,
RandomAccess, Cloneable, Serializable
```

Here, E denotes the Element Type

Vector Class extends AbstractList and implements multiple interfaces like Serializable, Cloneable, Iterable<E>, Collection<E>, List<E>, RandomAccess interfaces.

PROGRAM 1: Write A Program that accepts a shopping list of items and performs the following operations: Add an item at a specified location, delete an item in the list, and print the contents of the vector

CODE:

Code1_ShoppingListVector.java

Exp4 > Code1_ShoppingListVector.java > Code1_ShoppingListVector > main(String[])

```java
package Exp4;

import java.util.Scanner;
import java.util.Vector;

public class Code1_ShoppingListVector {
    Run | Debug
    public static void main(String[] args) {
        System.out.println(x: "Prerna Sunil Jadhav - 60004220127");

        Scanner sc = new Scanner(System.in);
        System.out.println(x: "Enter the No. of Item \n(you can extend it later as it is in Vector: ");
        int initialSize = sc.nextInt();
        // creating a vector of given Capacity = 20 and Increment=10
        // Here vector capacity will increase by 10 when needed
        Vector<String> shoppingList = new Vector<String>(initialSize, capacityIncrement: 10);
        for(int i = 0; i<initialSize; i++){
            System.out.print("Enter Item "+(i+1)+" : ");
            String item = sc.next();
            shoppingList.add(item);
        }
        int ch = 0;
        while(ch!=4){
            System.out.print(s: "Menu\n1.Add a new item\n2.Delete an item\n3.Show List\n4.Exit\nEnter
            your choice: ");
            ch = sc.nextInt();

            switch(ch){
                case 1:
                    System.out.print(s: "Enter the element to be added: ");
                    String item = sc.next();
                    System.out.print(s: "Enter the position at which it needs to added: ");
                    int index = sc.nextInt();
                    shoppingList.add(index-1, item);
                    System.out.println(item+" added Successfully!! --> "+shoppingList.toString());
                    break;
                case 2:
                    System.out.print(s: "Enter the element to be removed: ");
                    String remove_item = sc.next();
                    shoppingList.remove(remove_item);
                    System.out.println(remove_item+" removed Successfully!! --> "+shoppingList.toString
                    ());
                    break;
                case 3:
                    System.out.println(shoppingList.toString());
                    break;
```

```
44                  case 4:
45                      System.exit(status: 0);
46                  default:
47                      System.out.println(x: "Invalid choice");
48              }
49          }
50          sc.close();
51      }
52  }
```

OUTPUT:

```
Prerna Sunil Jadhav - 60004220127
Enter the No. of Item
(you can extend it later as it is in Vector:
3
Enter Item 1 : Top
Enter Item 2 : Jeans
Enter Item 3 : Kurta
Menu
1.Add a new item
2.Delete an item
3.Show List
4.Exit
Enter your choice: 1
Enter the element to be added: Jacket
Enter the position at which it needs to added: 2
Jacket added Successfully!! --> [Top, Jacket, Jeans, Kurta]
Menu
1.Add a new item
2.Delete an item
3.Show List
4.Exit
Enter your choice: 2
Enter the element to be removed: Top
Top removed Successfully!! --> [Jacket, Jeans, Kurta]
Menu
1.Add a new item
2.Delete an item
3.Show List
4.Exit
Enter your choice: 3
[Jacket, Jeans, Kurta]
Menu
1.Add a new item
2.Delete an item
3.Show List
4.Exit
```

PROGRAM2: Write a java program to find frequency of an element in the given Vector array.

CODE:

Code2_FrequencyOfElement.java

Exp4 > Code2_FrequencyOfElement.java > Code2_FrequencyOfElement > main(String[])

```java
package Exp4;
import java.util.*;
public class Code2_FrequencyOfElement {
    Run | Debug
    public static void main(String[] args) {
        System.out.println(x: "Prerna Sunil Jadhav - 60004220127");
        Vector<Integer> v = new Vector<>();
        v.add(e: 23);
        v.add(e: 89);
        v.add(e: 23);
        v.add(e: 45);
        v.add(e: 23);
        v.add(e: 89);
        System.out.println("Vector: "+v.toString());
        Map<Integer, Integer> mp = new HashMap<>();
        for (int i = 0; i < v.size(); i++){  // Traverse through array elements and count frequencies{
            if (mp.containsKey(v.get(i))){
                mp.put(v.get(i), mp.get(v.get(i)) + 1);
            }
            else{
                mp.put(v.get(i), value: 1);
            }
        }
        for (Map.Entry<Integer, Integer> entry : mp.entrySet()){ // Traverse through map and print frequencies
            System.out.println(entry.getKey() + " occurs " + entry.getValue()+" times ");
        }
    }
}
```

OUTPUT:

```
Prerna Sunil Jadhav - 60004220127
Vector: [23, 89, 23, 45, 23, 89]
23 occurs 3 times
89 occurs 2 times
45 occurs 1 times
```

CONCLUSION:

- The usage of vectors in java is mainly in cases when we want the processes in a synchronized manner since ArrayList and Vector both possess the property of dynamic sizes, but ArrayList is avoided when working with multiple threads.
- Vector class in Java throws ConcurrentModificationException, IllegalArgumentException and NullPointerException exceptions.
- Vectors in Java can be initialized using four types of constructors.
- Various methods are provided in the Vector class for handling the vector operations.
- We can use vectors to implement Tree Data structure or anywhere we are unsure about the size.

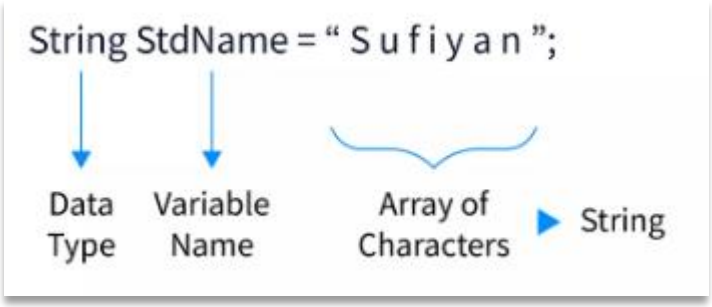
Name – Prerna Sunil Jadhav **SAP ID - 60004220127**

Experiment No - 05

AIM: TO IMPLEMENT STRINGS

THEORY:

- Strings are an integral part of programming.
- In Java, We have a String class for creating and manipulating strings.
- Also, there is an interface called CharSequence used for representing a character sequence.
- The String class is one of the classes which implement this interface.
- Hence, string is basically an object that represents a sequence of char values..
- For example, the word "sufiyan" is an array of characters 's', 'u', 'f', 'i', 'y', 'a', 'n. Here "sufiyan" is nothing but a sequence of characters i.e. a string.
- We have different String methods such as concat(), length(), compareTo(), equals(), etc. to handle various string operations.



- There can be two ways(or methods) of doing this:
  - ✓ Using char[] array in Java

```
public class Main {

  public static void main(String args[]) {
    char[] ch = { 's', 'u', 'f', 'i', 'y', 'a', 'n' };
    System.out.println(ch); // sufiyan
  }
}
```

  - ✓ Output:

```
sufiyan
```

  - ✓ Using String class in Java

```
// student name
public class Main {

  public static void main(String args[]) {
    String stdName = "sufiyan";
    System.out.println(stdName); // sufiyan
  }
}
```

  - ✓ Output:

```
sufiyan
```

PROGRAM 1: Write A Program to check if 2 strings are Meta strings or not. Meta strings are the strings which can be made equal by exactly one swap in any of the strings. Equal string are not considered here as Meta strings.
Example: str1 = "geeks", str2 = "keegs"
By just swapping 'k' and 'g' in any of string, both will become same.
Example: str1 = "Converse", str2 = "Conserve"
By just swapping 'v' and's' in any of string, both will become same.

CODE:

Code1_MetaStrings.java

Exp5 > Code1_MetaStrings.java > ...

```java
package Exp5;
import java.util.Arrays;
public class Code1_MetaStrings {
    Run | Debug
    public static void main(String[] args) {
        System.out.println(x: "Prerna Sunil Jadhav");
        String str1 = "converse";
        String str2 = "converse";
        if(areAlmostEqual(str1,str2)){
            System.out.println(x: "They are Meta Strings");
        }else{
            System.out.println(x: "They are not Meta Strings");
        }
    }
    static boolean areAlmostEqual(String s1, String s2) {
        int[] s1Array = new int[26];
        int[] s2Array = new int[26];
        int counter = 0;
        for(int i = 0;i<s1.length();i++){
           char s = s1.charAt(i);
           char ss = s2.charAt(i);
           if(s != ss)
              counter++;
           if(counter > 2)
              return false;
            s1Array[s -'a']++;
           s2Array[ss -'a']++;
        }
        return Arrays.equals(s1Array, s2Array);
    }
}
```

OUTPUT:

```
Prerna Sunil Jadhav
They are Meta Strings
```

PROGRAM2: Write a java program to count number of alphabets, digits, special symbols, blank spaces, and words from the given sentence. Also count number of vowels and consonants**.**

CODE:

Code2_CountInString.java

Exp5 > Code2_CountInString.java > Code2_CountInString > countCharacterType(String)

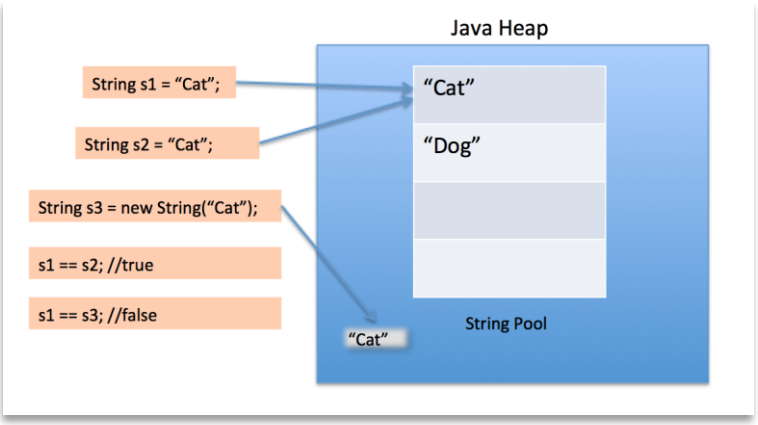
```java
package Exp5;

import java.util.Scanner;

public class Code2_CountInString {
    Run | Debug
    public static void main(String[] args) {
        System.out.println(x: "Prerna Sunil Jadhav - 60004220127");
        Scanner sc = new Scanner(System.in);
        System.out.print(s: "Enter a String: ");
        String str = sc.nextLine();
        countCharacterType(str);
        sc.close();
    }
    static void countCharacterType(String str)
    {
        int vowels = 0, consonant = 0, specialChar = 0, digit = 0, alphabets = 0, blank = 0;

        for (int i = 0; i < str.length(); i++) {
            char ch = str.charAt(i);
            if ( (ch >= 'a' && ch <= 'z') || (ch >= 'A' && ch <= 'Z') ) {
                alphabets++;
                // To handle upper case letters
                ch = Character.toLowerCase(ch);
                if (ch == 'a' || ch == 'e' || ch == 'i' ||
                    ch == 'o' || ch == 'u')
                    vowels++;
                else
                    consonant++;
            }
            else if (ch >= '0' && ch <= '9')
                digit++;
            else if (ch == ' ' || ch == '\t')
                blank++;
            else
                specialChar++;
        }

        System.out.println("Alphabets: " + alphabets);
        System.out.println("Vowels: " + vowels);
        System.out.println("Consonant: " + consonant);
        System.out.println("Digit: " + digit);
        System.out.println("Blank Spaces: " + blank);
        System.out.println("Special Character: " + specialChar);
    }
}
```

OUTPUT:

```
Prerna Sunil Jadhav - 60004220127
Enter a String: #ComputerEngineering In DJSCE 2023
Alphabets: 26
Vowels: 10
Consonant: 16
Digit: 4
Blank Spaces: 3
Special Character: 1
```

CONCLUSION:

- In Java, Strings are the Objects which are internally a sequence of characters. In simple words, Strings are the collection/combination of characters.
- Newly created strings are stored in a special area in the heap called String pool or String Constant Pool.
- Strings can be created using the string literals as well as using the new keyword as strings are objects in Java.
- Strings created using the new keyword are allocated memory in the heap memory outside the string constant pool.
- There are different methods of Strings in Java that can be used to make it easy to work with string in Java.
- Strings can be concatenated using the two ways: the + operator and the concat() method of string in Java.
- Strings can be formatted using the format() method.



- We can use the escape character(\) to escape some characters in String and go through all of the string text.
- Strings in Java are immutable which simply means that the string value can not be changed once it gets initialized(or created) .