

**CHAROTAR UNIVERSITY OF SCIENCE and
TECHNOLOGY**

**DEVANG PATEL INSTITUTE OF ADVANCE
TECHNOLOGY and RESEARCH**

Computer Science and Engineering

Subject Name: Java Programming

Semester: III

Subject Code: CE251

Academic year: 2020-21

Name: Parth N Patel

ID NO :19DCS098

PART-I
DATA TYPES, VARIABLES, STRING, CONTROL
STATEMENTS, OPERATORS, ARRAYS

PRACTICAL-1

AIM:

Write a program that declares one integer variable called var1. Give value 10 to this variable and then, using one println() statement, display the value on the screen like this:

“10 is the value of var1.”

PROGRAM CODE:

```
class SP_1
{
    public static void main(String[] args)
    {
        int var1=10;
        System.out.println(var1+" is the value of var1");
        System.out.println("PARTH PATEL");
        System.out.println("19DCS098");
    }
}
```

OUTPUT:

```
C:\Users\Parth Patel>cd C:\Java\JAVA_practicals  
C:\Java\JAVA_practicals>javac SP_1.java  
C:\Java\JAVA_practicals>java SP_1  
10 is the value of var1  
PARTH PATEL  
19DCS098
```

CONCLUSION:

In this Practical, we learnt how to use println statement

PRACTICAL-2

AIM:

Write a console program to declare and initialize a double variable with some value such as 1234.5678. Then retrieve the integral part of the value and store it in a variable of type long, and the first four digits of the fractional part and store them in an integer of type short.

Display the value of the double variable by outputting the two values stored as integers

PROGRAM CODE:

```
class SP_2
{
    public static void main(String[] args)
    {
        double x=1234.5678;
        long x1;
        short x2;
        x1=(long)x;
        x=x% 1234;
        x*=10000;
        x2=(short)x;
        System.out.println(x2+"."+x1);
        System.out.println("PARTH PATEL");
        System.out.println("19DCS098");

    }
}
```

OUTPUT:

```
C:\Java\JAVA_practicals>javac SP_2.java  
  
C:\Java\JAVA_practicals>java SP_2  
5678.1234  
PARTH PATEL  
19DCS098
```

CONCLUSION:

In this practical, we learnt how to manipulate different datatypes

PRACTICAL-3

AIM:

Write an application that creates a two-dimension array with int values. The first, second and third elements should be arrays with one, two and three numbers respectively. Display the length of each dimension.

PROGRAM CODE:

```
class SP_3
{
    public static void main(String[] args)
    {

        int x[][]={{1},{1,2},{1,2,3}};
        System.out.println(x.length+" is the length of row");
        System.out.println(x[0].length+" is the length of column 1");
        System.out.println(x[1].length+" is the length of column 2");
        System.out.println(x[2].length+" is the length of column 3");
        System.out.println("PARTH PATEL");
        System.out.println("19DCS098");

    }
}
```

OUTPUT:

```
C:\Java\JAVA_practicals>javac SP_3.java
C:\Java\JAVA_practicals>java SP_3
3 is the length of row
1 is the length of column 1
2 is the length of column 2
3 is the length of column 3
PARTH PATEL
19DCS098
```

CONCLUSION:

In this practical, we learnt about 2-D array

PRACTICAL-4

AIM:

An electric appliance shop assigns code 1 to motor, 2 to fan, 3 to tube and 4 for wires. All other items have code 5 or more. While selling the goods, a sales tax of 8% to motor, 12% to fan, 5% to tube light, 7.5% to wires and 3% for all other items is charged. A list containing the product code and price in two different arrays. Write a java program using switch statement to prepare the bill.

PROGRAM CODE:

```
import java.util.*;

class SP_4

{

    public static void main(String args[])
    {

        int code[]={ 1,2,3,4,5};
        int price[]={ 200,100,50,50,500};
        double bill=0;
        Scanner input= new Scanner(System.in);
        System.out.print("Enter the code : ");
        int code2=input.nextInt();
        switch(code2)
        {

            case 1:
                bill+=price[0]+(price[0]*8)/100;
                System.out.println("Product Code : "+code[0]);
                System.out.println("Price : "+price[0]);
                System.out.println("TOTAL : "+bill);
                break;

            case 2:
```



```
        bill+=price[1]+(price[1]*12)/100;
        System.out.println("Product Code : "+code[1]);
        System.out.println("Price : "+price[1]);
        System.out.println("TOTAL : "+bill);
        break;
    case 3:
        bill+=price[2]+(price[2]*5)/100;
        System.out.println("Product Code : "+code[2]);
        System.out.println("Price : "+price[2]);
        System.out.println("TOTAL : "+bill);
        break;
    case 4:
        bill+=price[3]+(price[3]*7.5)/100;
        System.out.println("Product Code : "+code[3]);
        System.out.println("Price : "+price[3]);
        System.out.println("TOTAL : "+bill);
        break;
    case 5:
        bill+=price[4]+(price[4]*3)/100;
        System.out.println("Product Code : "+code[4]);
        System.out.println("Price : "+price[4]);
        System.out.println("TOTAL : "+bill);
        break;
    }
    input.close();
    System.out.println();
    System.out.println("PARTH PATEL");
    System.out.println("19DCS098");
}
}
```

OUTPUT:

```
C:\Java\JAVA_practicals>java SP_4
Enter the code : 2
Product Code : 2
Price : 100
TOTAL : 112.0

PARTH PATEL
19DCS098
```

CONCLUSION:

In this practical, we learnt about functionality of switch

PRACTICAL-5

AIM:

Write a program to show output like:

* * * * *

* * * *

* * *

* *

*

PROGRAM CODE:

```
class SP_5
{
    public static void main(String[] args)
    {
        for(int i=5;i>=1;i--)
        {
            for(int j=1;j<=i;j++)
                System.out.print("*");
            System.out.println();
        }
        System.out.println("PARTH PATEL");
        System.out.println("19DCS098");
    }
}
```

OUTPUT:

```
C:\Java\JAVA_practicals>javac SP_5.java

C:\Java\JAVA_practicals>java SP_5
*****
****
***
**
*
PARTH PATEL
19DCS098
```

CONCLUSION:

In this practical, we learnt how to form patterns in java

PART-II

String

PRACTICAL-1

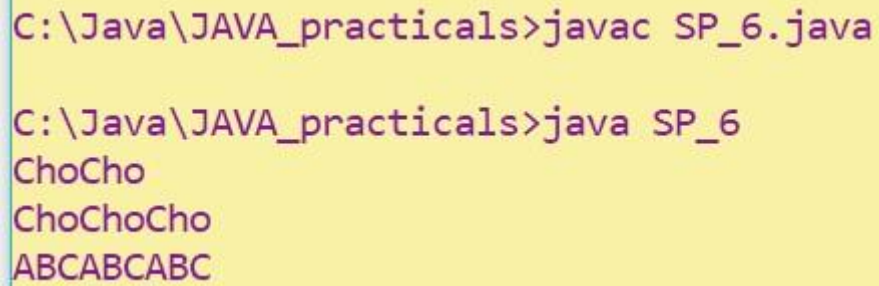
AIM:

Given a string and a non-negative int n, we'll say that the front of the string is the first 3 chars, or whatever is there if the string is less than length 3. Return n copies of the front;
front_times('Chocolate', 2) → 'ChoCho' front_times('Chocolate', 3) → 'ChoChoCho'
front_times('Abc', 3) → 'AbcAbcAbc'

PROGRAM CODE:

```
class SP_6
{
    public static void main(String args[])
    {
        SP_6 obj=new SP_6();
        String s1=obj.front_lines("Chocolate",2);
        String s2=obj.front_lines("Chocolate",3);
        String s3=obj.front_lines("ABC",3);
        System.out.println(s1);
        System.out.println(s2);          System.out.println(s3);
    }
    String front_lines(String s, int no)
    {
        String s1="";          for(int i=1;i<=no;i++)
        {
            s1=s1+s.substring(0,3);
        }
    }
    return s1;
}
```

```
}  
}
```

OUTPUT:A screenshot of a terminal window with a yellow background. It shows the command prompt at C:\Java\JAVA_practicals. The first command is 'javac SP_6.java' and the second is 'java SP_6'. The output of the second command is three lines: 'ChoCho', 'ChoChoCho', and 'ABCABCABC'.

```
C:\Java\JAVA_practicals>javac SP_6.java  
  
C:\Java\JAVA_practicals>java SP_6  
ChoCho  
ChoChoCho  
ABCABCABC
```

CONCLUSION:

In this Practical, we learnt how to play with strings

PRACTICAL-2

AIM:

Given an array of ints, return the number of 9's in the array.

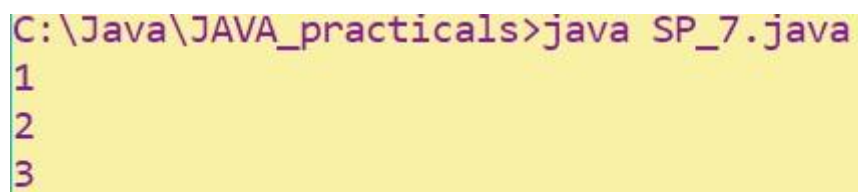
array_count9([1, 2, 9]) → 1 array_count9([1, 9, 9]) → 2 array_count9([1, 9, 9, 3, 9]) → 3

PROGRAM CODE:

```
class SP_7
{
    public static void main(String[] args)
    {
        SP_7 obj=new SP_7();
        int x1[]={1,2,9};
        int x2[]={1,9,9};
        int x3[]={1,9,9,3,9};
        int n1=obj.array_count(x1);
        int n2=obj.array_count(x2);
        int n3=obj.array_count(x3);
        System.out.println(n1);
        System.out.println(n2);
        System.out.println(n3);
    }
    int array_count(int[] arr)
    {
        int cnt=0;
        for(int i=0;i<arr.length;i++)
        {
            if(arr[i]==9)
```

```
        cnt++;  
    }  
    return cnt;  
}  
}
```

OUTPUT:



```
C:\Java\JAVA_practicals>java SP_7.java  
1  
2  
3
```

CONCLUSION:

In this practical, we learnt how to count number of occurrence of a number

PRACTICAL-3

AIM:

Given an array of ints, return True if one of the first 4 elements in the array is a 9.

The array length may be less than 4.

array_front9([1, 2, 9, 3, 4]) → True array_front9([1, 2, 3, 4, 9]) → False array_front9([1, 2, 3, 4, 5]) → False

PROGRAM CODE:

```
class SP_8
{
    public static void main(String[] args)
    {
        int[] a1={ 1,2,9,3,4};        int[] a2={ 1,2,3,4,9};        int[]
a3={ 1,2,3,4,5};        SP_8 s1=new SP_8();        boolean
b1=s1.array_front9(a1);        boolean b2=s1.array_front9(a2);        boolean
b3=s1.array_front9(a3);

        System.out.println(b1);

        System.out.println(b2);        System.out.println(b3);

    }
    boolean array_front9(int[] arr)
    {
        int n=arr.length;

        if(n>4)

            n=4;

        for(int i=0;i<n;i++)
        {

            if(arr[i]==9)                return true;

        }

        return false;
    }
}
```

```
    }  
}
```

OUTPUT:

```
C:\Java\JAVA_practicals>java SP_8.java  
true  
false  
false
```

CONCLUSION:

In this practical, we learnt how to check if number exist or not

PRACTICAL-4

AIM:

Given a string, return a string where for every char in the original, there are two chars.
double_char('The') → 'TThhee' double_char('AAbb') → 'AAAAbbbb' double_char('Hi-There')
→ 'HHii--TThheerree'

PROGRAM CODE:

```
class SP_9
{
    public static void main(String[] args)
    {
        SP_9 s=new SP_9();
        String s1=s.double_char("ABC");
        System.out.println(s1);
    }
    String double_char(String str)
    {
        String str2="";
        for(int i=0;i<str.length();i++)
        {
            str2=str2+str.charAt(i)+str.charAt(i);
        }
        return str2;
    }
}
```

OUTPUT:

```
C:\Java\JAVA_practicals>javac SP_9.java  
C:\Java\JAVA_practicals>java SP_9  
AABBCC
```

CONCLUSION:

In this practical, we learnt how to play with strings

PRACTICAL-5

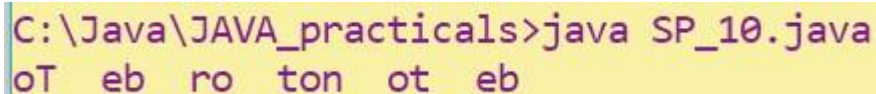
AIM:

Write a program that will reverse the sequence of letters in each word of your chosen paragraph. For instance, “To be or not to be” would become “oT e bro ton ot eb”.

PRACTICAL CODE:

```
import java.util.*; class SP_10
{
    public static void main(String[] args)
    {
        String tmp;
        StringTokenizer s=new StringTokenizer("To be or not to be");
        while(s.hasMoreTokens())
        {
            tmp=s.nextToken();
            for(int i=tmp.length();i>0;i--)
            {
                System.out.print(tmp.charAt(i-1));
            }
            System.out.print(" ");
        }
    }
}
```

OUTPUT:



```
C:\Java\JAVA_practicals>java SP_10.java
oT eb ro ton ot eb
```

CONCLUSION:

In this practical, we learnt how to reverse a string

PART-III

Object Oriented Programming : Classes, Methods, Constructor

PRACTICAL-1

AIM:

Write a java program for converting Pound into Rupees. (Accept Pounds from command line argument and using scanner class also and take 1 Pound = 100 Rupees.)

PROGRAM CODE:

```
import java.util.Scanner;
class SP_11
{
    public static void main(String[] args)
    {
        double pound=0,rupee=0;
        Scanner input=new Scanner(System.in);
        System.out.print("Enter the Value in Pound : ");
        pound=input.nextDouble();
        rupee=pound*100;
        System.out.println("The value of "+pound+" pound is "+rupee+" rupees");
    }
}
```

OUTPUT:

```
C:\Java\JAVA_practicals>javac SP_11.java

C:\Java\JAVA_practicals>java SP_11
Enter the Value in Pound : 15.5
The value of 15.5 pound is 1550.0 rupees
```

CONCLUSION:

In this practical, we learnt the role of constructor

PRACTICAL-2

AIM:

Write a program that defines TriangleArea class with three constructor. The first form accept no arguments. The second accept one double value for radius. The third form accept any two arguments.

PROGRAM CODE:

```
class TriangleArea
{
    double radius;
    String str=new String();
    public TriangleArea()
    {
        radius=0;
        str=null;
        System.out.println("Constructor has no argument");
    }
    public TriangleArea(double r)
    {
        radius=r;
        System.out.println("Constructor has one argument");
    }
    public TriangleArea(String s,double r)
    {
        radius=r;
        str=s;
        System.out.println("Constructor has two arguments");
        System.out.println("STR: "+str);
    }
}

class SP_12
{
    public static void main(String[] args)
```



```
{  
    TriangleArea obj=new TriangleArea();  
    TriangleArea obj1=new TriangleArea(10.22);  
    TriangleArea obj2=new TriangleArea("TRIANGLE",10.22);  
}  
}
```

OUTPUT:

```
C:\Java\JAVA_practicals>javac SP_12.java  
  
C:\Java\JAVA_practicals>java SP_12  
Constructor has no argument  
Constructor has one argument  
Constructor has two arguments  
STR: TRIANGLE
```

CONCLUSION:

In this practical, we learnt how to overload constructor

PRACTICAL-3

AIM:

Create a class called Employee that includes three pieces of information as instance variables—a first name (type String), a last name (type String) and a monthly salary(double). Your class should have a constructor that initializes the three instance variables. Provide a set and a get method for each instance variable. If the monthly salary is not positive, set it to 0.0. Write a test application named Employee Test that demonstrates class Employee's capabilities. Create two Employee objects and display each object's yearly salary. Then give each Employee a 10% raise and display each Employee's yearly salary again.

PROGRAM CODE:

```
class Employee
{
    private String firstName;
    private String lastName;
    private double monthlySalary;
    Employee(String fname,String lname,double ms)
    {
        firstName=fname;
        lastName=lname;
        if(ms<0)
            monthlySalary=0.0;
        else
            monthlySalary=ms;
    }
    void set_firstName(String fname)
    {
        firstName=fname;
    }
    void set_lastName(String lname)
    {
        lastName=lname;
    }
}
```

```
void set_monthlySalary(double ms)
{
    if(ms<0)
        monthlySalary=0.0;
    else
        monthlySalary=ms;
}
String get_firstName()
{
    return firstName;
}
String get_lastName()
{
    return lastName;
}
double get_monthlySalary()
{
    return monthlySalary;
}
double yearlySalary()
{
    return 12*monthlySalary;
}
void raise(double raisePercent)
{
    monthlySalary=(monthlySalary*raisePercent)/100+monthlySalary;
}
}
class SP_13
{
    public static void main(String[] args)
    {
```

```
Employee emp1=new Employee("Parth","Patel",550000);
Employee emp2=new Employee("Jason","Taylor",120000);
System.out.println("Monthly Salary of "+ emp1.get_firstName() +" is "+
emp1.yearlySalary());
System.out.println("Monthly Salary of "+ emp2.get_firstName() +" is "+
emp2.yearlySalary());
emp1.raise(10.0);
emp2.raise(10.0);
System.out.println("Monthly Salary of "+emp1.get_firstName()+" is
"+emp1.yearlySalary());
System.out.println("Monthly Salary of "+emp2.get_firstName()+" is
"+emp2.yearlySalary());
    }
}
```

OUTPUT:

```
C:\Java\JAVA_practicals>javac SP_13.java

C:\Java\JAVA_practicals>java SP_13
Monthly Salary of Parth is 6600000.0
Monthly Salary of Jason is 1440000.0
Monthly Salary of Parth is 7260000.0
Monthly Salary of Jason is 1584000.0
```

CONCLUSION:

In this practical, we learnt importance of getters and setters

PRACTICAL-4

AIM:

Create a class called Date that includes three pieces of information as instance variables—a month (type int), a day (type int) and a year (type int). Your class should have a constructor that initializes the three instance variables and assumes that the values provided are correct. Provide a set and a get method for each instance variable. Provide a method displayDate that displays the month, day and year separated by forward slashes (/). Write a test application named DateTest that demonstrates class Date's capabilities.

PROGRAM CODE:

```
class Date
{
    private int day;
    private int month;
    private int year;
    Date(int d,int m,int y)
    {
        day=d;
        month=m;
        year=y;
    }
    void set_day(int d)
    {
        day=d;
    }
    void set_month(int m)
    {
        month=m;
    }
    void set_year(int y)
    {
        year=y;
    }
}
```

```
        int get_day()
        {
            return day;
        }
        int get_month()
        {
            return month;
        }
        int get_year()
        {
            return year;
        }
        void displayDate()
        {
            System.out.println(month+"/"+day+"/"+year);
        }
    }
    class SP_14
    {
        public static void main(String[] args)
        {
            Date d1=new Date(06,04,2001);
            d1.displayDate();
        }
    }
```

OUTPUT:

```
C:\Java\JAVA_practicals>javac SP_14.java  
  
C:\Java\JAVA_practicals>java SP_14  
4/6/2001
```

CONCLUSION:

In this practical, we learnt more details about constructor

PRACTICAL-5

AIM:

Complete the code and write main () method to execute program.

PROGRAM CODE:

```
public class MethodOverloading
{
    /*private void methodOverloaded()
    {
        System.out.println("Private method-1");
    }*/
    private int methodOverloaded(int i)
    {
        System.out.println("private int method\n"+"Entered value="+i);
        return i;
    }
    protected int methodOverloaded(double d)
    {
        System.out.println("protected int method\n"+"Entered value= "+d);
        return 1;
    }
    public void methodOverloaded(int i,double d)
    {
        System.out.println("Public int method\n"+"Entered value="+i+" "+d);
    }
}
class SP_15
{
    public static void main(String[] args)
    {
        MethodOverloading m1=new MethodOverloading();
```



```
//m1.methodOverloaded();  
System.out.println(m1.methodOverloaded(10));  
System.out.println(m1.methodOverloaded(20.99));  
m1.methodOverloaded(15,21.2);  
}  
}
```

OUTPUT:

Before Commenting :

```
C:\Java\JAVA_practicals>javac MethodOverloading.java  
MethodOverloading.java:27: error: no suitable method found for methodOverloaded(no arguments)  
    m1.methodOverloaded();  
      ^  
    method MethodOverloading.methodOverloaded(int) is not applicable  
      (actual and formal argument lists differ in length)  
    method MethodOverloading.methodOverloaded(double) is not applicable  
      (actual and formal argument lists differ in length)  
    method MethodOverloading.methodOverloaded(int,double) is not applicable  
      (actual and formal argument lists differ in length)  
1 error
```

After Commenting:

```
C:\Java\JAVA_practicals>javac MethodOverloading.java  
  
C:\Java\JAVA_practicals>java SP_15  
protected int method  
Entered value= 10.0  
1  
protected int method  
Entered value= 20.99  
1  
Public int method  
Entered value=15 21.2
```

CONCLUSION:

In this practical, we learnt about importance of syntax

PART-IV

Inheritance, Interface , Package

PRACTICAL-1

AIM:

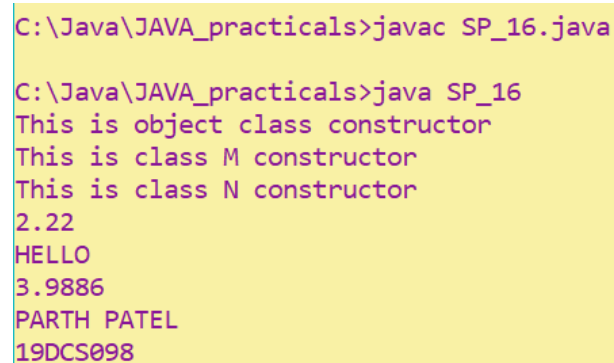
Write an application that demonstrates a class inheritance hierarchy. Class M extends object and has two instance variables of type float and String. Class N extends M and has one instance variable of type Double. Instantiate class N. Initialize and display its variables.

PROGRAM CODE:

```
class Object
{
    Object()
    {
        System.out.println("This is object class constructor");
    }
}
class M extends Object
{
    float x;
    String str;
    M()
    {
        System.out.println("This is class M constructor");
    }
}
class N extends M
{
    double y;
    N(float x,String str,double y)
```

```
        {
            System.out.println("This is class N constructor");
            this.x=x;
            this.str=str;
            this.y=y;
        }
        void display()
        {
            System.out.println(x+"\n"+str+"\n"+y);
            System.out.println("PARTH PATEL\n19DCS098");
        }
    }

class SP_16
{
    public static void main(String[] args)
    {
        N n=new N(2.22f,"HELLO",3.9886);
        n.display();
    }
}
```

OUTPUT:A screenshot of a terminal window with a yellow background. It shows the compilation and execution of a Java program. The commands and their outputs are as follows:
C:\Java\JAVA_practicals>javac SP_16.java
C:\Java\JAVA_practicals>java SP_16
This is object class constructor
This is class M constructor
This is class N constructor
2.22
HELLO
3.9886
PARTH PATEL
19DCS098

```
C:\Java\JAVA_practicals>javac SP_16.java
C:\Java\JAVA_practicals>java SP_16
This is object class constructor
This is class M constructor
This is class N constructor
2.22
HELLO
3.9886
PARTH PATEL
19DCS098
```

CONCLUSION:

In this practical, we learnt about basics of inheritance

PRACTICAL-2

AIM:

Create a class named 'Member' having the following members:

Data members

- 1 - Name
- 2 - Age
- 3 - Phone number
- 4 - Address
- 5 – Salary

It also has a method named 'printSalary' which prints the salary of the members. Two classes 'Employee' and 'Manager' inherits the 'Member' class. The 'Employee' and 'Manager' classes have data members 'specialization' and 'department' respectively. Now, assign name, age, phone number, address and salary to an employee and a manager by making an object of both of these classes and print the same.

PROGRAM CODE:

```
class Member
{
    String Name,ph_number,Add;
    int age,salary;
    void printSalary()
    {
        System.out.println("Salary : "+salary);
    }
}

class Employee extends Member
{
    String dep;
    String spec;
    Employee(String Name,int age,String ph_number,String Add,String dep,String spec,int salary)
    {
        this.Name=Name;
        this.age=age;
```

```
        this.ph_number=ph_number;
        this.Add=Add;
        this.dep=dep;
        this.spec=spec;
        this.salary=salary;
    }

    void display()
    {
        System.out.println("Name: "+Name);
        System.out.println("Age :"+age);
        System.out.println("Phone Number : "+ph_number);
        System.out.println("Address : "+Add);
        System.out.println("Specialization : "+spec);
        printSalary();
    }
}

class Manager extends Member
{
    String dep;
    String spec;
    Manager(String Name,int age,String ph_number,String Add,String dep,String spec,int salary)
    {
        this.Name=Name;
        this.age=age;
        this.ph_number=ph_number;
        this.Add=Add;
        this.dep=dep;
        this.spec=spec;
        this.salary=salary;
    }

    void display()
```

```
        {
            System.out.println("Name: "+Name);
            System.out.println("Age :"+age);
            System.out.println("Phone Number : "+ph_number);
            System.out.println("Address : "+Add);
            System.out.println("Specialization : "+spec);
            printSalary();
        }
    }
}

class SP_17
{
    public static void main(String[] args)
    {
        Employee e1=new
Employee("PNP",24,"88888","IND","DEvops","M.tech",10000);
        Manager m1=new
Manager("MNP",30,"66666","IND","ACC","MBA",30000);
        e1.display();
        System.out.println();
        m1.display();
        System.out.println("PARTH PATEL\n19DCS098");
    }
}
```

OUTPUT:

```
C:\Java\JAVA_practicals>javac SP_17.java

C:\Java\JAVA_practicals>java SP_17
Name: PNP
Age :24
Phone Number : 88888
Address : IND
Specialization : M.tech
Salary : 10000

Name: MNP
Age :30
Phone Number : 66666
Address : IND
Specialization : MBA
Salary : 30000
PARTH PATEL
19DCS098
```

CONCLSION:

In this practical, we learnt more about inheritance

PRACTICAL-3

AIM:

Create a class named 'Rectangle' with two data members 'length' and 'breadth' and two methods to print the area and perimeter of the rectangle respectively. Its constructor having parameters for length and breadth is used to initialize length and breadth of the rectangle. Let class 'Square' inherit the 'Rectangle' class with its constructor having a parameter for its side (suppose s) calling the constructor of its parent class as 'super(s,s)'. Print the area and perimeter of a rectangle and a square.

Also use array of objects.

PROGRAM CODE:

```
class Rectangle
{
    double l,b;

    Rectangle(double l, double b)
    {
        this.l = l;
        this.b = b;
    }

    double Perimeter()
    {
        return 2*(l+b);
    }

    double Area()
    { return l*b;
    }
}

class Square extends Rectangle
{
    Square(double length, double breadth)
    {
        super(length,breadth);
    }
}
```



```
}  
}  
class SP_18  
{  
    public static void main(String arg[])  
    {  
        Square s1[] = new Square[2];  
        s1[0] = new Square(10,12.5);  
        s1[1] = new Square(5,15);  
        System.out.println("Perimeter of Rectangle : "+s1[0].Perimeter());  
        System.out.println("Area of Rectangle :"+s1[0].Area());  
        System.out.println("Perimeter of Rectangle : "+s1[1].Perimeter());  
        System.out.println("Area of Rectangle is:"+s1[1].Area());  
        System.out.println("PARTH PATEL\n19DCS098");  
    }  
}
```

OUTPUT:

```
C:\Java\JAVA_practicals>javac SP_18.java  
  
C:\Java\JAVA_practicals>java SP_18  
Perimeter of Rectangle : 45.0  
Area of Rectangle :125.0  
Perimeter of Rectangle : 40.0  
Area of Rectangle is:75.0  
PARTH PATEL  
19DCS098
```

CONCLUSION:

In this practical, we learnt about super keyword

PRACTICAL-4

AIM:

Create a class named 'Shape' with a method to print "This is This is shape". Then create two other classes named 'Rectangle', 'Circle' inheriting the Shape class, both having a method to print "This is rectangular shape" and "This is circular shape" respectively. Create a subclass 'Square' of 'Rectangle' having a method to print "Square is a rectangle". Now call the method of 'Shape' and 'Rectangle' class by the object of 'Square' class.

PROGRAM CODE:

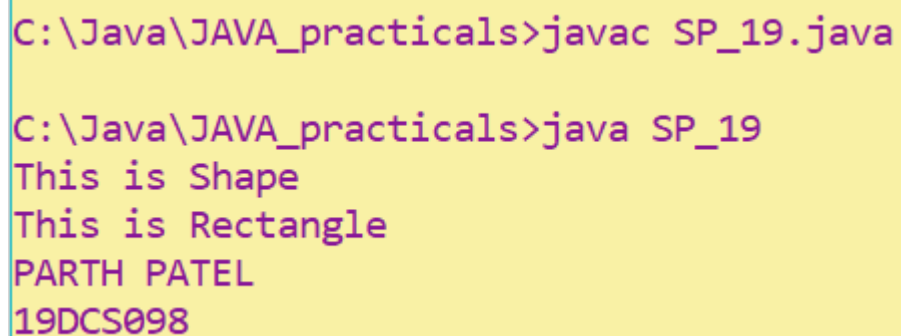
```
class Shape
{
    void printShape()
    {
        System.out.println("This is Shape");
    }
}

class Rectangle extends Shape
{
    void printRectangle()
    {
        System.out.println("This is Rectangle");
    }
}

class Circle extends Shape
{
    void printCircle()
    {
        System.out.println("This is Circle");
    }
}
```

```
class Square extends Rectangle
{
    void printSquare()
    {
        System.out.println("Square is a Rectangle");
    }
}

class SP_19
{
    public static void main(String args[])
    {
        Square s=new Square();
        s.printShape();
        s.printRectangle();
        System.out.println("PARTH PATEL\n19DCS098");
    }
}
```

OUTPUT:A screenshot of a Windows command prompt window with a yellow background. It shows the compilation and execution of a Java program. The first command is 'javac SP_19.java' and the second is 'java SP_19'. The output of the second command is displayed on four lines: 'This is Shape', 'This is Rectangle', 'PARTH PATEL', and '19DCS098'.

```
C:\Java\JAVA_practicals>javac SP_19.java

C:\Java\JAVA_practicals>java SP_19
This is Shape
This is Rectangle
PARTH PATEL
19DCS098
```

CONCLUSION:

In this practical, we learnt about multilevel inheritance

PRACTICAL-5

AIM:

Write a java that implements an interface AdvancedArithmetic which contains a method signature `int divisor_sum(int n)`. You need to write a class called `MyCalculator` which implements the interface. `divisorSum` function just takes an integer as input and return the sum of all its divisors. For example divisors of 6 are 1, 2, 3 and 6, so `divisor_sum` should return 12. The value of `n` will be at most 1000.

PROGRAM CODE:

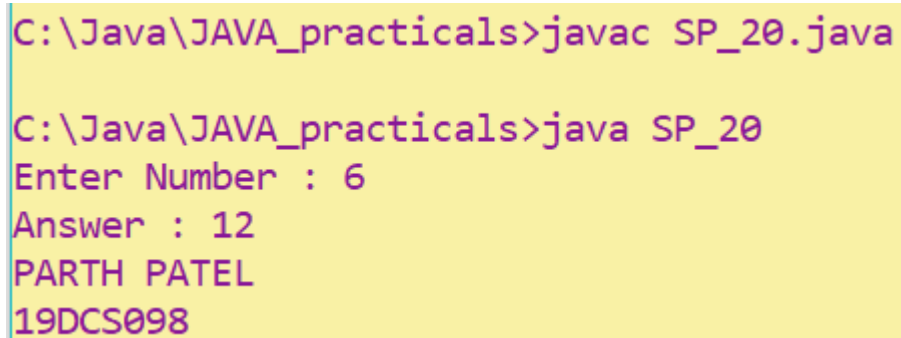
```
import java.util.*;

interface AdvancedArithmetic
{
    int divisor_sum(int n);
}

class MyCalculator implements AdvancedArithmetic
{
    public int divisor_sum(int n)
    {
        int sum=0;
        for(int i=1;i<=n;i++)
        {
            if(n%i==0)
                sum+=i;
        }
        return sum;
    }
}

class SP_20
{
    public static void main(String arg[])
    {
        Scanner input = new Scanner(System.in);
        System.out.print("Enter Number : ");
        int num = input.nextInt();
    }
}
```

```
while(num>1000)
{
    System.out.print("Enter the value of num : ");
    num = input.nextInt();
}
AdvancedArithmetic c1 = new MyCalculator();
System.out.println("Answer : "+c1.divisor_sum(num));
System.out.println("PARTH PATEL\n19DCS098");
}
```

OUTPUT:A screenshot of a Windows command prompt window with a yellow background. It shows the compilation and execution of a Java program. The first command is 'javac SP_20.java' and the second is 'java SP_20'. The output of the program is displayed below the second command: 'Enter Number : 6', 'Answer : 12', 'PARTH PATEL', and '19DCS098' on separate lines.

```
C:\Java\JAVA_practicals>javac SP_20.java

C:\Java\JAVA_practicals>java SP_20
Enter Number : 6
Answer : 12
PARTH PATEL
19DCS098
```

CONCLUSION:

In this practical, we learnt about interface

PRACTICAL-6

AIM:

Assume you want to capture shapes, which can be either circles (with a radius and a color) or rectangles (with a length, width, and color). You also want to be able to create signs (to post in the campus center, for example), each of which has a shape (for the background of the sign) and the text (a String) to put on the sign.

Create classes and interfaces for circles, rectangles, shapes, and signs.

Write a program that illustrates the significance of interface default method.

PROGRAM CODE:

```
interface Sign
{
    void text(String s);
}

class Shape
{
    void display()
    {
        System.out.println("This is shape");
    }
}

interface C extends Sign
{
    default void print()
    {
        System.out.println("This is Circle");
    }
}

class Circle extends Shape implements C , Sign
{

```

```
int radius = 20;
String color = "Red";
void display()
{
    super.display();
    System.out.println("Radius of circle: "+radius+" and Color: "+color);
}
public void text(String s)
{
    if(s.equals("Left") || s.equals("left") || s.equals("LEFT") )
    {
        System.out.println("Point towards: Left");
        System.out.println("Message: "+s);
    }
    else if(s.equals("Right") || s.equals("right") || s.equals("RIGHT") )
    {
        System.out.println("Point towards: Right");
        System.out.println("Message: "+s);
    }
    else {
        System.out.println("Point towards: Center");
        System.out.println("Message: "+s);
    }
}
} interface R
{
    default void print()
    {
        System.out.println("This is Rectangle");
    }
}
class Rectangle extends Shape implements R , Sign
{
    int length = 10 , width = 15;    String color = "White";    void display()
```

```
{
    super.display();
    System.out.println("Length of Rectangle: "+length+" and Width: "+width+" with Color: "+color);
}
public void text(String s)
{
    if(s.equals("Left") || s.equals("left") || s.equals("LEFT") )
    {
        System.out.println("Point towards: Left");
        System.out.println("Message: "+s);
    }
    else if(s.equals("Right") || s.equals("right") || s.equals("RIGHT") )
    {
        System.out.println("Point towards: Right");
        System.out.println("Message: "+s);
    }
    else
    {
        System.out.println("Point towards: Center");
        System.out.println("Message: "+s);
    }
}
}
class SP_21
{
    public static void main(String args[])
    {
        Circle c = new Circle();
        c.display();
        c.text("Center");
        Rectangle r1 = new Rectangle();
        r1.display();
    }
}
```



```
        r1.text("Right");  
        System.out.println("PARTH PATEL\n19DCS098");  
    }  
}
```

OUTPUT:

```
C:\Java\JAVA_practicals>javac SP_21.java  
  
C:\Java\JAVA_practicals>java SP_21  
This is shape  
Radius of circle: 20 and Color: Red  
Point towards: Center  
Message: Center  
This is shape  
Length of Rectangle: 10 and Width: 15 with Color: White  
Point towards: Right  
Message: Right  
PARTH PATEL  
19DCS098
```

CONCLUSION:

In this practical, we learnt about default methods in interface

PRACTICAL-7

AIM:

Write a java program which shows importing of classes from other user define packages.

PROGRAM CODE:

```
package pack_1;

public class pack_prac_1
{
    public void show()
    {
        System.out.println(" This is Package");
    }
}

import pack_1.*;

class SP_21
{
    public static void main(String[] args)
    {
        pack_prac_1 p1 = new pack_prac_1();
        p1.show();
    }
}
```

OUTPUT:

```
C:\Java\JAVA_practicals>java SP_22
```

```
C:\Java\JAVA_practicals>This is package
```

```
C:\Java\JAVA_practicals>javac SP_22.java
```

CONCLUSION:

In this practical, we learnt about packages

PART-V

Exception Handling

PRACTICAL-1

AIM:

Write a java program which takes two integers x & y as input, you have to compute x/y. If x and y are not integers or if y is zero, exception will occur and you have to report it.

PROGRAM CODE:

```
import java.util.*;

class SP_23
{
    public static void main (String args[])
    {
        float result;
        Scanner input = new Scanner(System.in);
        System.out.print("Enter Num-1: ");
        int n1 = input.nextInt();
        System.out.print("Enter Num-2: ");
        int n2 = input.nextInt();
        try
        {
            result = n1/n2;
            System.out.println("Answer :"+result);
        }
        catch (NumberFormatException e)
        {
            System.out.println("Format of the Number is wrong!");
        }
        catch (ArithmeticException e)
        {
            System.out.println("Number cannot be divided by zero");
        }
    }
}
```

```
}  
}  
}  
}
```

OUTPUT:

```
C:\Users\Parth Patel>cd C:\Java\JAVA_practicals  
  
C:\Java\JAVA_practicals>javac SP_23.java  
  
C:\Java\JAVA_practicals>java SP_23  
Enter Num-1: 23  
Enter Num-2: 0  
Number cannot be divided by zero
```

CONCLUSION:

In this practical, we learnt about basics of exceptional handling

PRACTICAL-2

AIM:

A piece of Java code is given below. You have to complete the code by writing down the handlers for exceptions thrown by the code. The exceptions the code may throw along with the handler message are listed below:

Division by zero: Print "Invalid division". String parsed to a numeric variable: Print "Format mismatch". Accessing an invalid index in string: Print "Index is invalid". Accessing an invalid index in array:

Print "Array index is invalid".

MyException: This is a user defined Exception which you need to create. It takes a parameter param. When an exception of this class is encountered, the handler should print "MyException[param]", here param is the parameter passed to the exception class.

Exceptions other than mentioned above: Any other exception except the above ones fall in this category. Print "Exception encountered". Finally, after the exception is handled, print "Exception Handling Completed".

Example: For an exception of MyException class if the parameter value is 5, the message will look like MyException[5].

PROGRAM CODE:

```
import java.util.*;

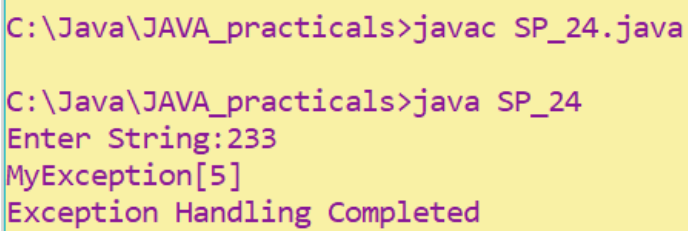
class MyException extends Exception
{
    int n1;
    public MyException(int n)
    {
        this.n1 = n;
    } void printout()
    {
        System.out.println("MyException["+n1+"]");
    }
}

class SP_24
{
```

```
public static void main (String args[])
{
    try
    {
        Scanner sc = new Scanner(System.in);
        int n1 = 2/2;
        System.out.print("Enter String:");
        String str = sc.next();
        int n2 = Integer.parseInt(str);
        int n3[] = {1,2};
        n3[1] = 5;
        String s = "PNP";
        s.charAt(5);
        throw new MyException(5);
    }
    catch(NumberFormatException e)
    {
        System.out.println("Number format is wrong!");
    }
    catch(ArithmeticException e)
    {
        System.out.println("Division by zero is not defined!");
    }
    catch(ArrayIndexOutOfBoundsException e)
    {
        System.out.println("The Array index you have entered is invalid!");
    }
    catch(StringIndexOutOfBoundsException e)
    {
        System.out.println("The String index you have entered is invalid");
    }
    catch(MyException ex)
    {

```

```
        ex.out();  
    } finally  
    {  
        System.out.println("Exception Handling Completed");  
    }  
}  
}
```

OUTPUT:

```
C:\Java\JAVA_practicals>javac SP_24.java  
  
C:\Java\JAVA_practicals>java SP_24  
Enter String:233  
MyException[5]  
Exception Handling Completed
```

CONCLUSION:

In this practical, we learnt about throw keyword

PRACTICAL-3

AIM:

Write a java program to generate user defined exception using “throw” and “throws” keyword. Also write a java with a differentiate checked and unchecked exceptions. (Mention at least two checked and two unchecked exception in program).

PROGRAM CODE:

```
import java.util.*;

class Check {

    void checkLicenseAge(int age)

    {
        if(age < 18)
            System.out.println("Ineligible to get driving license");
        else
            System.out.println("Eligible to get driving license");
    }

    int division(int a,int b) throws ArithmeticException

    {
        return a/b;
    }
}

class SP_25

{

    public static void main(String []args)

    {

        Scanner input = new Scanner(System.in);

        try{

            Check c1 = new Check();

            System.out.print("Enter age : ");

            c1.checkLicenseAge(input.nextInt());

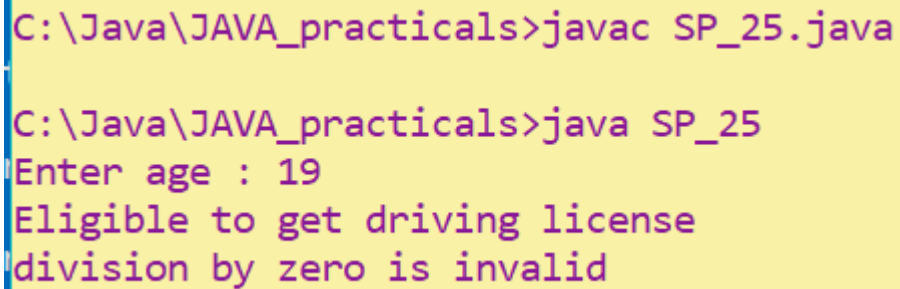
            System.out.println(c1.division(222,0));

        }

    }

}
```

```
        catch(InputMismatchException e)
        {
            System.out.println("number is not valid");
        }
        catch(ArithmeticException e)
        {
            System.out.println("division by zero is invalid");
        }
    }
}
```

OUTPUT:A screenshot of a Windows command prompt window with a yellow background. The text is in a purple font. It shows the compilation and execution of a Java program. The first line is the compilation command, the second is the execution command, followed by the program's output which includes a prompt for age and two lines of feedback based on the input.

```
C:\Java\JAVA_practicals>javac SP_25.java
C:\Java\JAVA_practicals>java SP_25
Enter age : 19
Eligible to get driving license
division by zero is invalid
```

CONCLUSION:

In this practical, we learnt about difference between throw and throws

PART-VI

File Handling & Streams

PRACTICAL-1

AIM:

Write a program that will count the number of lines in each file that is specified on the command line. Assume that the files are text files. Note that multiple files can be specified, as in "java LineCounts file1.txt file2.txt file3.txt". Write each file name, along with the number of lines in that file, to standard output. If an error occurs while trying to read from one of the files, you should print an error message for that file, but you should still process all the remaining files

PROGRAM CODE:

```
import java.io.*;
class SP_26
{
    public static void main(String[] args) throws Exception
    {
        try {
            for(int i=0;i<args.length;i++)
            {
                File f1=new File(args[i]);
                FileReader fr = new FileReader(f1);
                BufferedReader breader=new BufferedReader(fr);
                String str=new String();
                int cnt=0;
                while((str=breader.readLine())!=null)
                    cnt++;
                System.out.println("File Name :"+f1.getName()+" Line Count : "+cnt);
            }
        }
        catch(Exception e)
        {
            e.printStackTrace();
        }
        finally {System.out.println("PARTH PATEL\n19DCS098");}
    }
}
```

OUTPUT:

```
C:\Java\JAVA_practicals\SGP>javac SP_26.java  
  
C:\Java\JAVA_practicals\SGP>java SP_26 p.txt q.txt  
File Name :p.txt Line Count : 8  
File Name :q.txt Line Count : 6  
PARTH PATEL  
19DCS098
```

CONCLUSION:

In this practical, we learnt about basics of file handling

PRACTICAL-2

AIM:

Write an example that counts the number of times a particular character, such as e, appears in a file. The character can be specified at the command line. You can use xanadu.txt as the input file.

Program Code:

```
import java.io.*;
class SP_27
{
    public static void main(String[] args)
    {
        try
        {
            File f=new File("p.txt");
            FileReader fr=new FileReader(f);
            BufferedReader breader=new BufferedReader(fr);
            int count=0;
            int c;
            while((c=breader.read())!=-1)
            {
                if(c==args[0].charAt(0))
                    count++;
            }
            System.out.println("Letter : "+args[0].charAt(0)+" count : "+count);
        }
        catch(Exception e)
        {e.printStackTrace();}
        finally {System.out.println("PARTH PATEL\n19DCS098");}
    }
}
```

OUTPUT:

```
C:\Java\JAVA_practicals\SGP>javac SP_27.java

C:\Java\JAVA_practicals\SGP>java SP_27 q
Letter : q count : 35
PARTH PATEL
19DCS098
```

CONCLUSION:

In this practical, we learnt about how to fetch data from file

PRACTICAL-3

AIM:

Write a Java Program to Search for a given word in a File. Also show use of Wrapper Class with an example

PROGRAM CODE:

```
import java.io.*;
import java.util.*;

class SP_28
{
    public static void main(String []args) throws Exception
    {
        File f = new File("p.txt");
        FileReader fr = new FileReader(f);
        BufferedReader br = new BufferedReader(fr);
        String words[] = null;
        String s = "parth";
        String i;
        int count = 0;
        while((i = br.readLine()) != null)
        {
            words = i.split(" ");
            for(String word : words)
            {
                if(word.equals(s))
                    count = 1;
            }
        }
        if(count == 1)
        {
            System.out.println(s+" word is present in the file!!");
        }
        else
        {
            System.out.println(s+ " word is not present in the file");
        }
        System.out.println("PARTH PATEL\n19DCS098");
        fr.close();
    }
}
```

OUTPUT:

```
C:\Java\JAVA_practicals\SGP>javac SP_28.java  
  
C:\Java\JAVA_practicals\SGP>java SP_28  
parth word is present in the file!!  
PARTH PATEL  
19DCS098
```

CONCLUSION:

In this practical, we learnt about how to search for a word in file

PRACTICAL-4

AIM:

Write a program to copy data from one file to another file. If the destination file does not exist, it is created automatically.

PROGRAM CODE:

```
import java.io.*;
class SP_29
{
    public static void main(String[] args)
    {
        try
        {
            FileInputStream fi=new FileInputStream("p.txt");
            FileOutputStream fo=new FileOutputStream("xyz.txt");
            int n;
            while((n=fi.read())!=-1)
            {
                fo.write(n);
            }
            System.out.println("Process is Successful\nFile Created");
        }
        catch(Exception e)
        { e.printStackTrace();}
        finally
        { fi.close();
          fo.close();
          System.out.println("PARTH PATEL\n19DCS098");
        }
    }
}
```

OUTPUT:

```
C:\Java\JAVA_practicals\SGP>javac SP_29.java
C:\Java\JAVA_practicals\SGP>java SP_29
Process is Successful
File Created
PARTH PATEL
19DCS098
```

CONCLUSION:

In this practical, we learnt about how to copy one file to another

PRACTICAL-5

AIM:

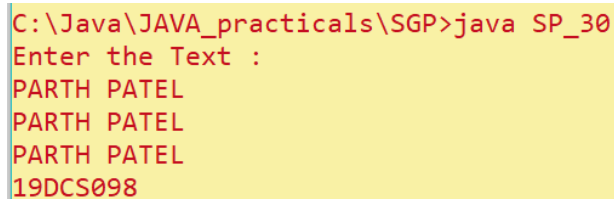
Write a program to show use of character and byte stream. Also show use of BufferedReader /BufferedWriter to read console input and write them into a file

PROGRAM CODE:

```
import java.io.*;
import java.util.Scanner;
class SP_30
{
    public static void main(String[] args) throws IOException
    {
        Scanner input=new Scanner(System.in);
        BufferedWriter bwriter=new BufferedWriter(new FileWriter(new
File("f.txt")));
        String str="";
        System.out.println("Enter the Text : ");
        str=input.nextLine();
        bwriter.write(str);

        bwriter.close();
        BufferedReader breader=new BufferedReader(new FileReader(new
File("f.txt")));
        while((str=breader.readLine())!=null)
            System.out.println(str);
        breader.close();
        System.out.println("PARTH PATEL\n19DCS098");
    }
}
```

OUTPUT:



```
C:\Java\JAVA_practicals\SGP>java SP_30
Enter the Text :
PARTH PATEL
PARTH PATEL
PARTH PATEL
19DCS098
```

CONCLUSION:

In this practical, we learnt about basics of BufferedWriter and BufferedReader

Part-7

Multithreading

PRACTICAL 1

AIM:

Write a program to create thread which display “Hello World” message.

- A. by extending Thread class
- B. by using Runnable interface

PROGRAM CODE:

```
class A extends Thread
{
    public void run()
    {
        System.out.println("Hello World from class Thread");
    }
}
class B implements Runnable
{
    public void run()
    {
        System.out.println("Hello World from Runnable");
    }
}
class SP_31
{
    public static void main(String[] args)
    {
        A obj1=new A();
        B obj2=new B();
        Thread th=new Thread(obj2);
        obj1.start();
        th.start();
        System.out.println("PARTH PATEL\n19DCS098");
    }
}
```

OUTPUT:

```
C:\Java\JAVA_practicals\SGP>javac SP_31.java  
  
C:\Java\JAVA_practicals\SGP>java SP_31  
PARTH PATEL  
19DCS098  
Hello World from Runnable  
Hello World from class Thread
```

CONCLUSION:

In this practical, we learnt basics of multithreading

PRACTICAL 2

AIM:

Write a program which takes N and number of threads as an argument. Program should distribute the task of summation of N numbers amongst number of threads and final result to be displayed on the console.

PROGRAM CODE:

```
import java.util.*;
class ThreadClass implements Runnable
{
    int n,t,s=0;
    int[] t1 = new int[100];
    ThreadClass(int n, int t , int[] t1)
    {
        this.n=n;
        this.t=t;
        this.t1=t1;
    }
    public void run()
    {
        for(int i=0;i<t;i++)
        {
            s=s+t1[i];
            System.out.println(t1[i]);
        }
    }
}
class SP_32
{
    public static void main(String[] args)
    {
        Scanner input = new Scanner(System.in);
        System.out.print("Enter the number :");
        int n = input.nextInt();
        System.out.print("Enter the number of threads :");
        int t = input.nextInt();
        int[] t1 = new int[100];
        t1[0] = n / t;
        int sum=0;
        for(int i=0;i<t-1;i++)
        {
            t1[i]=t1[1];
            sum=sum+t1[i];
        }
        t1[t]=n-sum;
        System.out.print("The number are :");
```

```
ThreadClass tc = new ThreadClass(n,t,t1);
Thread a = new Thread(tc);
a.start();
    System.out.println("PARTH PATEL\n19DCS098");

}
}
```

OUTPUT:

```
C:\Java\JAVA_practicals\SGP>javac SP_32.java
C:\Java\JAVA_practicals\SGP>java SP_32
Enter the number : 5
Enter the number of threads : 5
Result: 15
PARTH PATEL
19DCS098
```

CONCLUSION:

In this practical, we learnt about how to use more than one threads

PRACTICAL 3

AIM:

Write a java program that implements a multi-thread application that has three threads. First thread generates random integer every 1 second and if the value is even, second thread computes the square of the number and prints. If the value is odd, the third thread will print the value of cube of the number

PROGRAM CODE:

```
import java.util.Random;
class RandomThread extends Thread {
    public void run() {
        Random rand = new Random();
        for (int i = 0; i < 5; i++) {
            int randomInt = rand.nextInt(100);
            System.out.println("Random number is " + randomInt);
            if ((randomInt % 2) == 0) {
                SquareOfThread sThread = new SquareOfThread(randomInt);
                sThread.start();
            } else {
                CubeOfThread cThread = new CubeOfThread(randomInt);
                cThread.start();
            }
        }
        try {
            Thread.sleep(2000);
        } catch (InterruptedException e) {
            System.out.println(e);
        }
    }
}

class SquareOfThread extends Thread {
    int n;
    SquareOfThread(int n) {
        this.n = n;
    }
    public void run() {
        System.out.println("Square : " + (n*n));
    }
}

class CubeOfThread extends Thread {
    int n;
    CubeOfThread(int n) {
        this.n = n;
    }
    public void run() {
        System.out.println("Cube : " + (n*n*n));
    }
}
```

```
}  
}  
class SP_33 {  
public static void main(String[] args) {  
RandomThread rt = new RandomThread();  
rt.start();  
System.out.println("PARTH PATEL\n19DCS098");  
}  
}
```

OUTPUT:

```
C:\Java\JAVA_practicals\SGP>javac SP_33.java  
  
C:\Java\JAVA_practicals\SGP>java SP_33  
PARTH PATEL  
19DCS098  
Random number is 48  
Square : 2304  
Random number is 69  
Cube : 328509  
Random number is 47  
Cube : 103823  
Random number is 1  
Cube : 1  
Random number is 94  
Square : 8836
```

CONCLUSION:

In this practical, we learnt about how to use random numbers

PRACTICAL 4

AIM:

Write a program to increment the value of one variable by one and display it after one second using thread using sleep() method.

PROGRAM CODE:

```
import java.util.*;
class Increase extends Thread{
int num = 1;
public void run(){
for(int i=0;i<10;i++){
System.out.println(num++);
try {
Thread.sleep(1000);
} catch (InterruptedException e) {
System.out.println(e);
}
}
}
}
class SP_34 {
public static void main(String[] args) {
Increase t1 = new Increase();
t1.start();
System.out.println("PARTH PATEL\n19DCS098");
}
}
```


OUTPUT:

```
C:\Java\JAVA_practicals\SGP>javac SP_34.java  
  
C:\Java\JAVA_practicals\SGP>java SP_34  
PARTH PATEL  
19DCS098  
1  
2  
3  
4  
5  
6  
7  
8  
9  
10
```

CONCLUSION:

In this practical, we learnt about sleep() method

PRACTICAL 5

AIM:

Write a program to create three threads 'FIRST', 'SECOND', 'THIRD'. Set the priority of the 'FIRST' thread to 3, the 'SECOND' thread to 5(default) and the 'THIRD' thread to 7.

PROGRAM CODE:

```
class FIRST extends Thread{
public void run(){
System.out.println("Priority of FIRST Thread " + getPriority());
}
}
class SECOND extends Thread{
public void run(){
System.out.println("Priority of SECOND Thread " + getPriority());
}
}
class THIRD extends Thread{
public void run(){
System.out.println("Priority of THIRD Thread " + getPriority());
}
}
class SP_35 {
public static void main(String[] args) {
FIRST first = new FIRST();
SECOND second = new SECOND();
THIRD third = new THIRD();
first.setPriority(3);
second.setPriority(5);
third.setPriority(7);
first.start();
second.start();
third.start();
System.out.println("PARTH PATEL\n19DCS098");
}
}
```

OUTPUT:

```
C:\Java\JAVA_practicals\SGP>javac SP_35.java  
  
C:\Java\JAVA_practicals\SGP>java SP_35  
PARTH PATEL  
19DCS098  
Priority of FIRST Thread 3  
Priority of SECOND Thread 5  
Priority of THIRD Thread 7
```

CONCLUSION:

In this practical, we learnt about priority of threads

PRACTICAL 6

AIM:

Write a program to solve producer-consumer problem using thread synchronization.

PROGRAM CODE:

```
import java.util.ArrayList;
class SP_36 {
    public static void main(String[] args) throws InterruptedException {
        final Producer p1 = new Producer();
        Thread myThread1 = new Thread(new Runnable() {
            @Override
            public void run() {
                try {
                    p1.produce();
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
        });
        Thread myThread2 = new Thread(new Runnable() {
            @Override
            public void run() {
                try {
                    p1.consume();
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
        });
        myThread1.start();
        myThread2.start();
        myThread1.join();
        myThread2.join();
    }

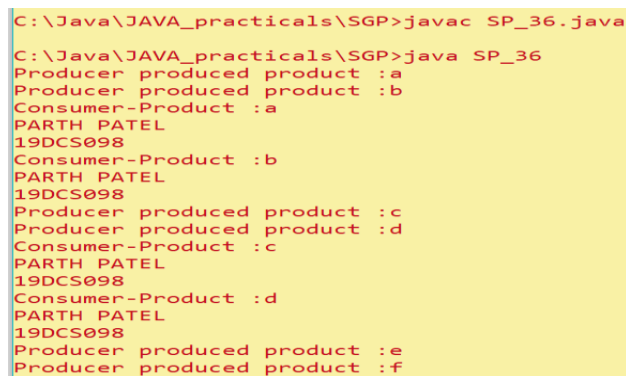
    public static class Producer {
        ArrayList<Character> list = new ArrayList<Character>();
        int capacity = 2;
        public void produce() throws InterruptedException {
            char value = 'a';
            while (true) {
                synchronized (this) {
                    while (list.size() == capacity)
                        wait();
                    System.out.println("Producer produced product :" + value);
                    list.add(value);
                }
            }
        }
    }
}
```

```
value++;
if (value == 'g') {
    System.exit(0);
    break;
}
notify();
Thread.sleep(1000);
}
}
}

public void consume() throws InterruptedException {
    while (true) {
        synchronized (this) {
            while (list.size() == 0)
                wait();
            char val = list.remove(0);
            if (val == 'g') {
                break;
            }

            System.out.println("Consumer-Product :" + val);
            notify();
            Thread.sleep(1000);
            System.out.println("PARTH PATEL\n19DCS098");
        }
    }
}
}
```

OUTPUT:



```
C:\Java\JAVA_practicals\SGP>javac SP_36.java
C:\Java\JAVA_practicals\SGP>java SP_36
Producer produced product :a
Producer produced product :b
Consumer-Product :a
PARTH PATEL
19DCS098
Consumer-Product :b
PARTH PATEL
19DCS098
Producer produced product :c
Producer produced product :d
Consumer-Product :c
PARTH PATEL
19DCS098
Consumer-Product :d
PARTH PATEL
19DCS098
Producer produced product :e
Producer produced product :f
```

CONCLUSION:

In this practical, we learnt about synchronization of threads

PART-8

COLLECTION FRAMEWORK AND GENERIC

PRACTICAL 1

AIM:

Design a Custom Stack using ArrayList class, which implements following functionalities of stack.

PROGRAM CODE:

```
import java.util.*;
class SP_37 {
    List lst = new ArrayList();
    public void push(Object obj) {
        lst.add(obj);
    }
    public void myPop() {
        lst.remove(peek());
    }
    public Object peek() {
        int temp = lst.size();
        return lst.get(temp - 1);
    }
    public boolean isEmpty() {
        return lst.isEmpty();
    }
    public int getSize() {
        return lst.size();
    }
    public String toString() {
        return "Array-List : [ " + lst.toString() + " ]";
    }
    public static void main(String[] args) {
        ArrayList<Character> Lst = new ArrayList<Character>();
        Lst.add('P');
        Lst.add('A');
        Lst.add('R');
        Lst.add('T');
        Lst.add('H');
        System.out.println(Lst.toString());
        Lst.remove(1);
        Lst.set(2, 'p');
        System.out.println(Lst.toString());
        System.out.println("PARTH PATEL\n19DCS098");
    }
}
```

```
}
```

OUTPUT:

```
C:\Java\JAVA_practicals\SGP>javac SP_37.java
Note: SP_37.java uses unchecked or unsafe operations
Note: Recompile with -Xlint:unchecked for details.

C:\Java\JAVA_practicals\SGP>java SP_37
[P, A, R, T, H]
[P, R, p, H]
PARTH PATEL
19DCS098
```

CONCLUSION:

In this practical, we learnt about arraylist

PRACTICAL 2

AIM:

Create a generic method for sorting an array of Comparable objects.

PROJECT CODE:

```
import java.util.*;
public class Employee implements Comparable<Employee> {
    String EName;
    int Eage;
    int Eid;
    Employee() {
        EName = null;
        Eage = 0;
        Eid = 0;
    }
    Employee(String EName, int Eage, int Eid) {
        this.EName = EName;
        this.Eage = Eage;
        this.Eid = Eid;
    }
    public int compare(Employee e) {
        return this.Eage - e.Eage;
    }
    public String toString() {
        return String.format("[%s, %d]", EName, Eid);
    }
}
class SP_38 {
    public static void main(String[] args) {
        Employee[] e = new Employee[3];
        e[0] = new Employee("Ram", 28, 9880);
        e[1] = new Employee("Sam", 22, 1100);
        e[2] = new Employee("Ravi", 45, 0001);
        e[3] = new Employee("Rao", 55, 1000);
        System.out.println("Before sorting: " + Arrays.toString(e));
        System.out.println();
        Arrays.sort(e);
        System.out.println("After sorting: " + Arrays.toString(e));
        System.out.println("PARTH PATEL\n19DCS098");
    }
}
```


OUTPUT:

```
C:\Java\JAVA_practicals\SGP>javac Employee.java

C:\Java\JAVA_practicals\SGP>java SP_38
Before sorting: [[Ram,9880], [Sam,1100], [Ravi,0001], [Rao,1000]]

After sorting: [[Ravi,0001], [Rao,1000], [Sam,1100], [Ram,9880]]
PARTH PATEL
19DCS098
```

CONCLUSION:

In this practical, we learnt about basics of generic method

PRACTICAL 3

AIM:

Write a program that counts the occurrences of words in a text and displays the words and their occurrences in alphabetical order of the words. Using Map and Set Classes.

PROGRAM CODE:

```
import java.util.*;
class SP_39 {
    public static void main(String args[]) {
        Map<String, Integer> map = new HashMap<>();
        Scanner input = new Scanner(System.in);
        System.out.print("Enter the string:");
        String sentence = input.next();
        String[] tokens = sentence.split(" ");
        for (String token : tokens) {
            String word = token.toLowerCase();
            if (map.containsKey(word)) {
                int count = map.get(word);
                map.put(word, count + 1);
            } else {
                map.put(word, 1);
            }
        }
        Set<String> keys = map.keySet();
        TreeSet<String> sortedKeys = new TreeSet<>(keys);
        for (String str : sortedKeys) {
            System.out.println(str + " count : " + map.get(str));
            System.out.println("PARTH PATEL\n19DCS098");
        }
    }
}
```

OUTPUT:

```
C:\Java\JAVA_practicals\SGP>javac SP_39.java  
  
C:\Java\JAVA_practicals\SGP>java SP_39  
Enter the string:Hello  
hello count : 1  
PARTH PATEL  
19DCS098
```

CONCLUSION:

In this practical, we learnt about basics of Map and set classes