return (count);

}

public static void main(String args[])

{

System.out.println("Enter array elements");

Scanner sc = new Scanner(System.in);

int k = sc.nextInt();

int arr[] = new int[k];

for(int i=0;i<k;i++)

{

arr[i]=sc.nextInt();

}

int x = array\_count9(arr,k);

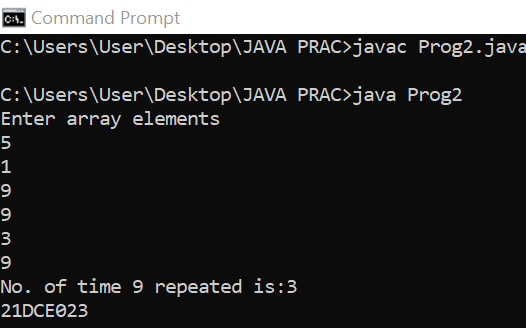
System.out.println("No. of time 9 repeated is:"+x);

System.out.println("21DCE066");

}

}

**OUTPUT:**

****

**CONCLUSION:**

* Here we understood the concept of passing an array through a function and also conditional statement if-else.

**PRACTICAL 2.3**

**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:**

import java.util.\*;

public class Prog3

{

public static boolean array\_front(int arr[])

{

for(int i=0;i<4;i++)

{

if(arr[i]==9)

{

return (true);

}

}

return (false);

}

public static void main(String args[])

{

Scanner sc = new Scanner(System.in);

System.out.println("Enter no.of elements :");

int n = sc.nextInt();

int a[] = new int[n];

System.out.println("Enter array elements :");

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

{

a[i]=sc.nextInt();

}

boolean k = array\_front(a);

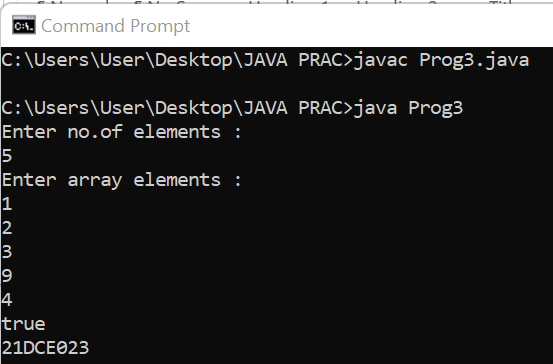
System.out.println(k);

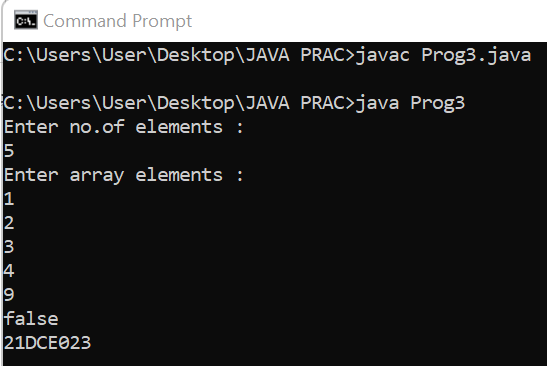
System.out.println("21DCE066");

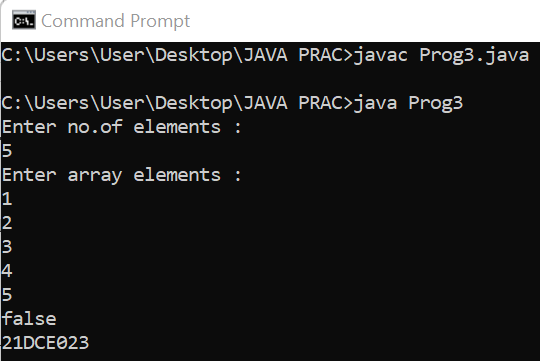
}

}

**OUTPUT:**

****

****

****

**CONCLUSION:**

* Here we understood about passing an array through a function and also about returning a Boolean datatype.

**PRACTICAL 2.4**

**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:**

import java.util.\*;

public class prog44

{

public static String double\_char(String str)

{

String res=" ";

for(int i=0;i<str.length();i++)

{

res = res + str.charAt(i) + str.charAt(i);

}

return (res);

}

public static void main(String args[])

{

Scanner sc = new Scanner(System.in);

String str = new String();

System.out.println("Enter a a string");

str = sc.nextLine();

String st = double\_char(str);

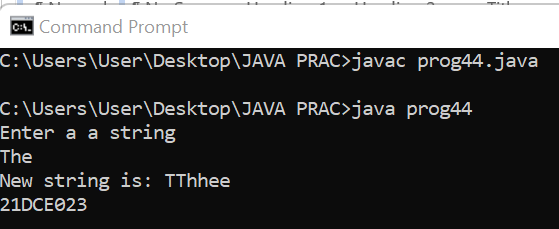
System.out.println("New string is:"+st);

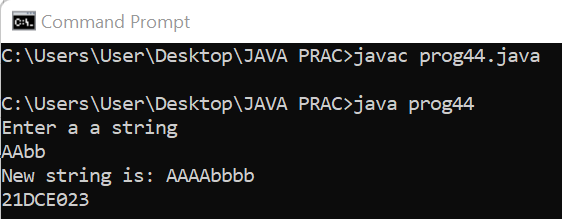
System.out.println("21DCE066");

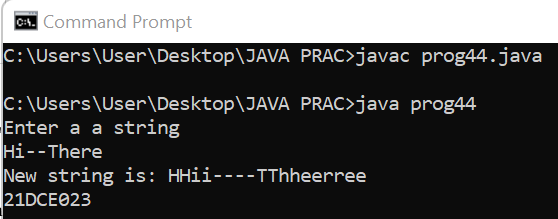
}

}

**OUTPUT:**



****

****

**CONCLUSION:**

* In this practical we understood the usage of charAt() function which converts our string into character and prints a particular character according to index.

**PRACTICAL 2.5**

**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”.**

**PROGRAM:**

import java.util.\*;

public class prog5

{

public static void main(String args[])

{

System.out.println("Enter a string:");

Scanner sc = new Scanner(System.in);

String st = sc.nextLine();

String []s=st.split(" ");

String s1 = new String();

for(int i=0;i<s.length;i++)

{

StringBuffer st1 = new StringBuffer(s[i]);

s1+=st1.reverse();

s1+=" ";

}

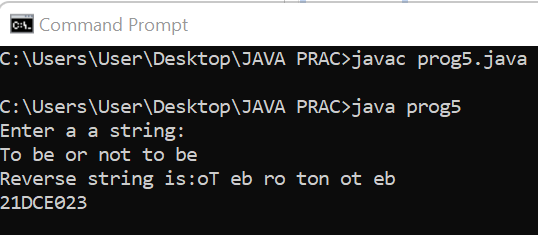
System.out.println("Reverse string is:"+s1);

System.out.println("21DCE066");

}

}

**OUTPUT:**

****

**CONCLUSION:**

* In this practical we understood using of the class called STRINGBUFFER and also understood about reverse function which reverses a string.

**PRACTICAL 2.6**

**AIM:**

**Perform following functionalities of the string:**

**● Find Length of the String**

**● Lowercase of the String**

**● Uppercase of the String**

**● Reverse String**

**● Sort the string**

**PROGRAM:**

import java.util.Arrays;

import java.util.\*;

public class Prog6

{

public static void main(String args[])

{

Scanner sc = new Scanner(System.in);

System.out.println("Enter a string");

String str1 = sc.nextLine();

System.out.println("Length of the string is:"+str1.length());

System.out.println("Lowercase is"+str1.toLowerCase());

System.out.println("Upperrcase is"+str1.toUpperCase());

char[]try1 = str1.toCharArray();

for(int i=try1.length-1;i>=0;i--){

System.out.print(try1[i]);

}

char[]ch = str1.toCharArray();

Arrays.sort(ch);

String sorted = new String(ch);

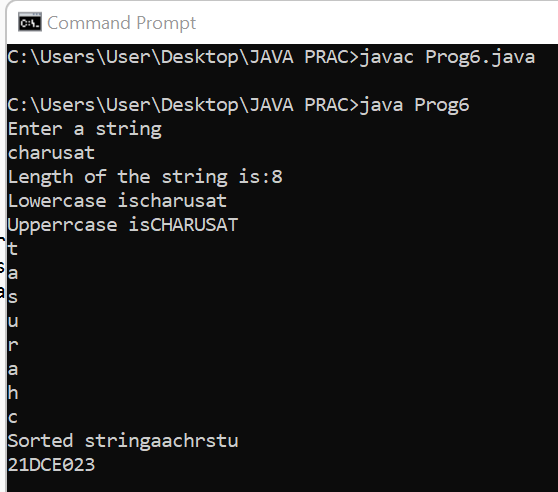
System.out.print("Sorted string"+sorted);

System.out.println("21DCE066");

}

}

**OUTPUT:**

****

**CONCLUSION:**

* In this practical we understood using of different string functions i.e length() function which finds length of string, toUpperCase() and toLowerCase() which converts string into uppercase and lowercase respectively and last one is toCharArray() which converts string into char array.

**PRACTICAL 2.7**

**AIM:**

**Perform following Functionalities of the string: “CHARUSAT University” ● Find length ● Replace ‘H’ by ‘N’ ● Convert all character in Uppercase ● Extract and print “CHARUSAT” from given string**

**PROGRAM:**

public class Main

{

            public static void main(String[] args) {

                        String name="CHARUSAT University";

                        System.out.println("The Length of the String is:"+name.length());

                        String rep=name.replace('H','N');

                        System.out.println("The replaced string is:"+rep);

                        System.out.println("The Upper case String is:"+ name.toUpperCase());

                        System.out.println("The Upper case String is:"+ name.toLowerCase());

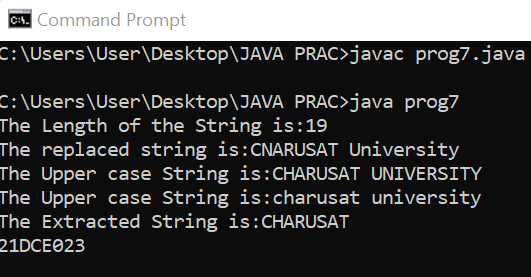
                        System.out.println("The Extracted String is:"+name.substring(0,8));

                        System.out.println("21DCE020");

            }

}

**OUTPUT:**

****

**CONCLUSION:**

* In this practical we understood using of different string functions i.e length() function which finds length of string, toUpperCase() and toLowerCase() which converts string into uppercase and lowercase respectively and last one is substring(Starting index, Ending index) which prints string according to the index given.

**SET-III: Object Oriented Programming: Classes, Methods, Constructors.**

**PRACTICAL 3.1**

**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:**

import java.util.\*;

public class pract3\_1

{

public static void main(String[] args)

{

Scanner c= new Scanner(System.in);

System.out.println("Enter currency in pound: ");

int pound= c.nextInt();

int rs= pound\*100;

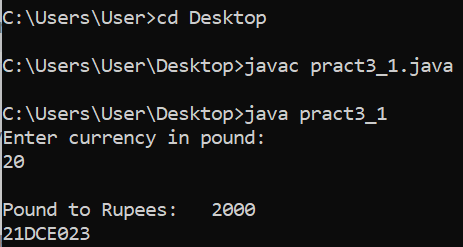
System.out.println("\nPound to Rupees: "+rs);

System.out.println("21DCE066");

}

}

**OUTPUT**:

****

**CONCLUSION:**

By performing this practical we learned how to convert a currency input by the user and convert it into a desired currency. In this program we converted pound which was input by the user into rupees that was desired.

**PRACTICAL 3.2**

**AIM:**

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

**PROGRAM:**

class TriangleArea

{

double b,h;

double area;

TriangleArea()

{

b=2;

h=3;

}

TriangleArea(double bas)

{

b=bas;

h=4;

}

TriangleArea(double ht,double bas)

{

h=ht;

b=bas;

}

void ar()

{

System.out.println("Area of triangle is:"+(0.5\*b\*h));

}

}

public class pra32 {

public static void main(String[] args) {

TriangleArea t1 = new TriangleArea();

TriangleArea t2 = new TriangleArea(2.3);

TriangleArea t3 = new TriangleArea(2,3.2);

t1.ar();

t2.ar();

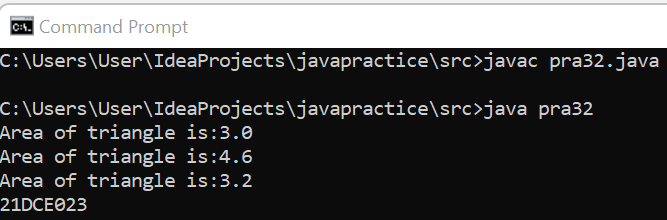
t3.ar();

System.out.println("21DCE066");

}

}

**OUTPUT**:



**Conclusion:**

By performing this practical we learned how to write a program using constructors of different class. We calculated the areas of various shapes in the constructors and called them in the main class.

**PRACTICAL – 3.3**

**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 EmployeeTest 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:**

import java.util.\*;

class Employee

{

String f\_name = new String();

String l\_name = new String();

double salary;

int raise = 10;

public void set()

{

System.out.println("Enter First name");

Scanner sc = new Scanner(System.in);

f\_name = sc.nextLine();

System.out.println("Enter Last name:");

l\_name = sc.nextLine();

System.out.println("Enter salary:");

salary = sc.nextInt();

if(salary<0)

{

salary=0.0;

}

}

public void get()

{

System.out.println("Name of employee is:"+f\_name+" "+l\_name);

System.out.println("Salary is:"+salary);

}

public void raise()

{

salary = salary + (raise\*salary)/100;

System.out.println("salary after raise is:"+salary);

}

}

public class pra33

{

public static void main(String[] args)

{

Employee e1 = new Employee();

Employee e2 = new Employee();

e1.set();

e2.set();

e1.get();

e2.get();

e1.raise();

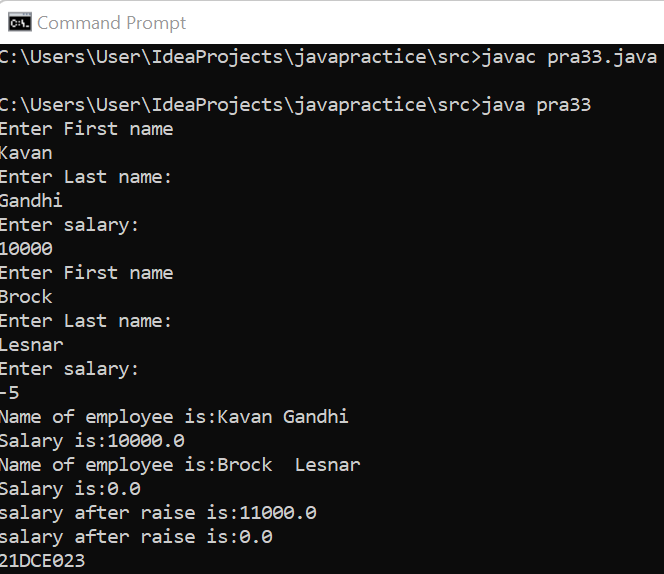
e2.raise();

System.out.println("21DCE066");

}

}

**OUTPUT:**



**Conclusion:**

By performing this practical we learned how we can input the information of two employees of a company by initializing the instances in the default constructor getting the values and displaying them using methods.

**PRACTICAL – 3.4**

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:**

import java.util.Scanner;

class DateTest{

public static void main(String[] args)

{

Scanner sc = new Scanner(System.in);

Date dt = new Date(requestInput("Enter month: ", sc),requestInput("Enter day: ", sc),requestInput("Enter year: ", sc));

dt.displayDate();

}

public static int requestInput(String s, Scanner sc)

{

System.out.print(s);

return sc.nextInt();

}

}

class Date{

private int month, day, year;

public Date(int month, int day, int year){

setMonth(month);

setDay(day);

setYear(year);

}

public void setMonth(int month)

{

this.month = month;

}

public void setDay(int day)

{

this.day = day;

}

public void setYear(int year)

{

this.year = year;

}

public int getMonth()

{

return month;

}

public int getDay()

{

return day;

}

public int getYear()

{

return year;

}

public void displayDate()

{

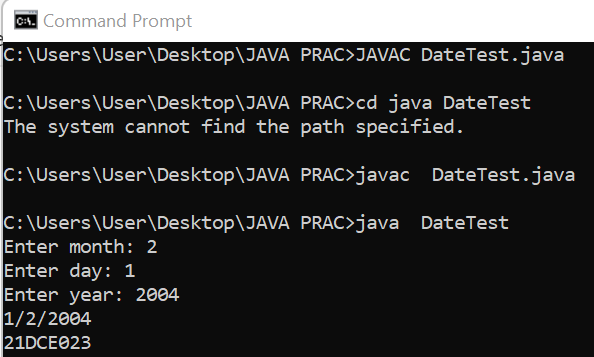
System.out.printf("%d/%d/%d\n", getDay(), getMonth(), getYear());

System.out.println("21DCE066");

}

}

**OUTPUT:**



**Conclusion:**

By performing this practical we learned how we can input the dates from the user then display it separately and combined using method.

**PRACTICAL – 3.5**

**Aim:**

**Write a program to print the area of a rectangle by creating a class named 'Area' taking the values of its length and breadth as parameters of its constructor and having a method named 'returnArea' which returns the area of the rectangle. Length and breadth of rectangle are entered through keyboard.**

**Program:**

import java.util.\*;

class area

{

int l,b;

area(int len, int bre)

{

l=len;

b=bre;

}

int returnArea()

{

return(l\*b);

}

}

public class prac3\_5

{

public static void main(String[] args)

{

Scanner sc = new Scanner(System.in);

int l,b;

System.out.println("Enter length:");

l=sc.nextInt();

System.out.println("Enter breadth:");

b=sc.nextInt();

area a1 = new area(l,b);

int x= a1.returnArea();

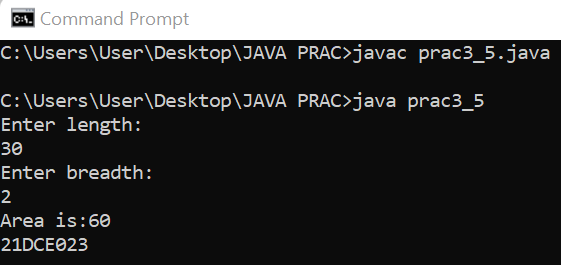
System.out.println("Area is:"+x);

System.out.println("21DCE066");

}

}

**OUTPUT:**

****

**Conclusion:**

By performing this practical we learned how we the find the area of any shape by entering the dimension of the shape. By passing the values to constructors and returning the area

**PRACTICAL – 3.6**

**Aim:**

**Print the sum, difference and product of two complex numbers by creating a class named ‘Complex’ with separate methods for each operation whose real and imaginary parts are entered by user.**

**Program:**

import java.util.\*;

class complex

{

int real,imag;

void setdata()

{

Scanner sc = new Scanner(System.in);

System.out.println("Enter real no.:");

real = sc.nextInt();

System.out.println("Enter imaginary no.:");

imag = sc.nextInt();

}

complex add(complex c)

{

complex temp = new complex();

temp.real=real+c.real;

temp.imag=imag+c.imag;

return (temp);

}

complex mul(complex k)

{

complex temps = new complex();

temps.real=real\*k.real;

temps.imag=imag\*k.imag;

return (temps);

}

complex sub(complex a)

{

complex tem = new complex();

tem.real=real-a.real;

tem.imag=imag-a.imag;

return (tem);

}

void getdata()

{

System.out.println("Complex no. is:"+real+ "+"+imag+ "i");

}

}

public class prac3\_6

{

public static void main(String[] args)

{

complex c1 = new complex();

complex c2 = new complex();

complex c3 = new complex();

c1.setdata();

c2.setdata();

c3 = c1.add(c2);

c3.getdata();

c3 = c1.mul(c2);

c3.getdata();

c3 = c1.sub(c2);

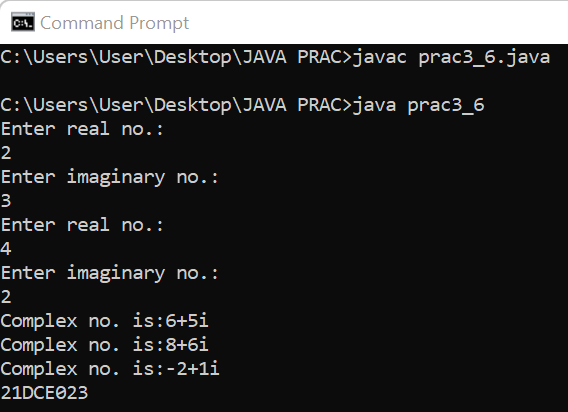
c3.getdata();

System.out.println("21DCE066");

}

}

**OUTPUT:**

****

**Conclusion:**

By performing this practical we learned to take real and imaginary parts of a complex number in a separate class and calculating the sum, difference and product of the two complex numbers in separate methods and printing the values.

**PRACTICAL – 3.7**

**Aim:**

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

**public class MethodOverloading**

**{ private void methodOverloaded()**

**{**

**//no argument, private method**

**} private int methodOverloaded(int i)**

**{ //code**

**}**

**protected int methodOverloaded(double d)**

**{ //code**

**}**

**public void methodOverloaded(int i, double d) {**

**//code**

**}**

**}**

**Program:**

class Main

{

private void methodOverloaded()

{

System.out.println("hi");

}

private int methodOverloaded(int i)

{

return i;

}

protected double methodOverloaded(double d)

{

return d;

}

public void methodOverloaded(int i, double d)

{

System.out.println("one two");

}

public static void main(String args[])

{

Main m=new Main();

m.methodOverloaded();

int g=m.methodOverloaded(6);

System.out.println(g);

double h=m.methodOverloaded(2.5d);

System.out.println(h);

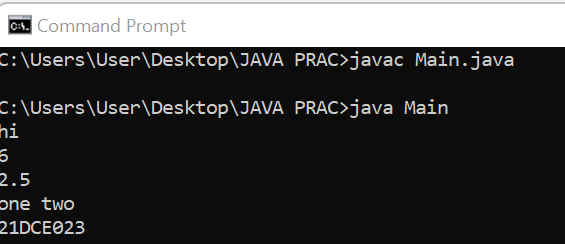
m.methodOverloaded(10,3.5);

System.out.println("21DCE066");

}

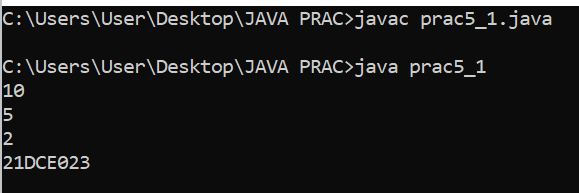
}

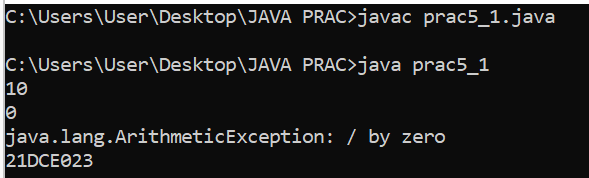
**OUTPUT:**

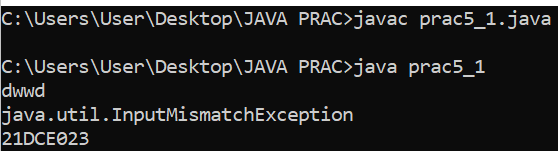


Conclusion:

By performing this practical we learn about how to call/access private, public and protected member and method and we know that the how will the method overloading work and how the compiler differentiate the all method with its arguments







**PART-IV**

**Inheritance, Interface, Package**

**PRACTICAL – 4.1**

**Aim:**

**Create a class with a method that prints "This is parent class" and its subclass with another method that prints "This is child class". Now, create an object for each of the class and call 1 - method of parent class by object of parent class 2 - method of child class by object of child class 3 - method of parent class by object of child class.**

**Program:**

class pc

{

public void pmethod(){

System.out.println("This is parent class");

}

}

class ch extends pc

{

public void cmethod(){

System.out.println("This is child class");

}

}

public class prac4\_1

{

public static void main(String ar[])

{

pc p1 = new pc();

ch c1 = new ch();

System.out.println("calling from parent object");

p1.pmethod();

System.out.println("calling from child object");

c1.cmethod();

System.out.println("calling from child object to parent function");

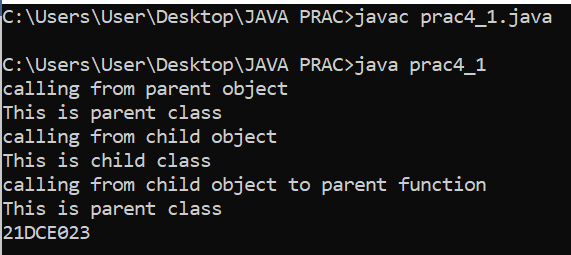
c1.pmethod();

System.out.println("21DCE066");

}

}

**Output:**

****

**Conclusion:**

By performing this practical we learnt about how we can inherit a class from the parent class by using the keyword extends and learnt the concepts of inheritance.

**PRACTICAL – 4.2**

**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:**

class member

{

String name = new String("Kavan");

int age = 19;

long ph\_no=43443535;

String addr = new String("S.nagar");

float sal = 10000.00f;

}

class employee extends member

{

String dep ="Police";

void display()

{

System.out.println("Name is:"+name);

System.out.println("Age is:"+age);

System.out.println("Phone no. is:"+ph\_no);

System.out.println("Address is:"+addr);

System.out.println("Department:"+dep);

System.out.println("Salary is:"+sal);

}

}

class manager extends member

{

String spe="Forensic";

void display()

{

System.out.println("Name is:"+name);

System.out.println("Age is:"+age);

System.out.println("Phone no. is:"+ph\_no);

System.out.println("Address is:"+addr);

System.out.println("Salary is:"+sal);

System.out.println("Specialization:"+spe);

}

}

class prac4\_2

{

public static void main(String a[])

{

employee e1 = new employee();

manager m1 = new manager();

e1.display();

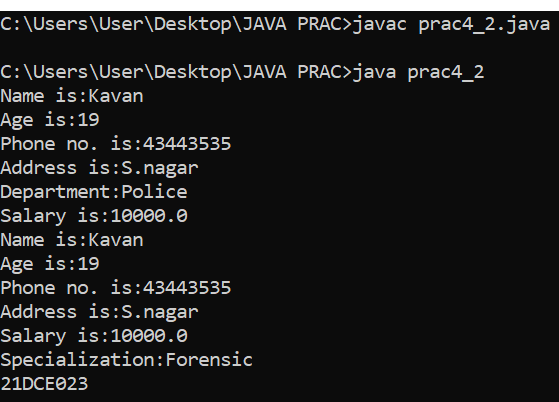
m1.display();

System.out.println("21DCE066");

}

}

**Output:**



**Conclusion:**

By performing this practical we learnt about the concept of “Hierarchical inheritance” how we can inherit a class by using the parent class by using the keyword extends.

**PRACTICAL – 4.3**

**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:**

class rectangle

{

float len,bre;

public rectangle()

{}

public rectangle(float a,float b)

{

len=a;

bre=b;

}

}

class square extends rectangle

{

float side;

public square()

{

super();

}

public square(float s)

{

super(s,s);

side=s;

}

void display()

{

System.out.println("Area of rectangle is:"+(len\*bre));

System.out.println("Perimeter of rectangle is:"+(2\*len+bre));

System.out.println("Area of square is:"+(4\*side));

System.out.println("Perimeter of square is:"+(2\*side+side));

}

}

class pra4\_3

{

public static void main(String a[])

{

square obj[]={new square(),new square(5.0f),new square(7.1f)};

for(int i=0;i<3;i++)

{

obj[i].display();

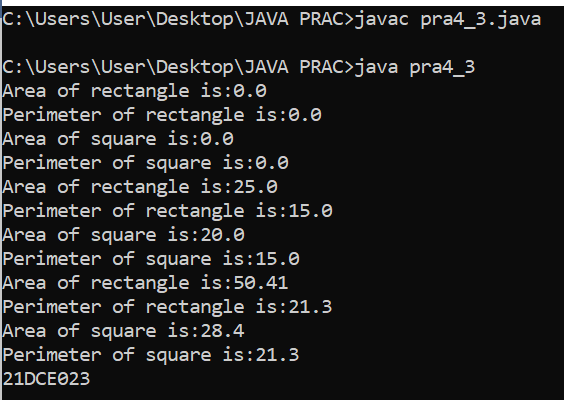
}

System.out.println("21DCE066");

}

}

**Output:**



**Conclusion:**By performing this practical we learnt about how we can inherit the classes by using the keyword “Super()”, and extends.

**PRACTICAL – 4.4**

**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:**

class shape

{

void outs()

{

System.out.println("This is shape");

}

}

class rectangle extends shape

{

void outr()

{

System.out.println("This is rectangular shape");

}

}

class circle extends shape

{

void outc()

{

System.out.println("This is circular shape");

}

}

class square extends rectangle

{

void outsq()

{

System.out.println("Square is a rectangle");

}

}

class prac4\_4

{

public static void main(String ar[])

{

square s1 = new square();

circle c1 = new circle();

s1.outs();

s1.outr();

c1.outc();

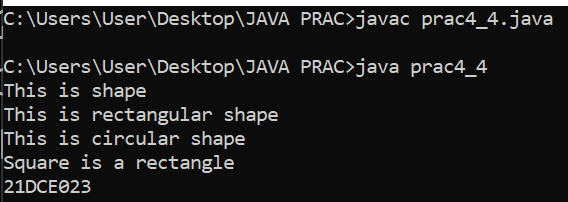
s1.outsq();

System.out.println("21DCE066");

}

}

**Output:**

****

**Conclusion:**

By performing this practical we have learned the concept of “Method Overloading” that how we can call methods with the same name but with different arguments.

**PRACTICAL – 4.5**

**Aim:**

**Create a class 'Degree' having a method 'getDegree' that prints "I got a degree". It has two subclasses namely 'Undergraduate' and 'Postgraduate' each having a method with the same name that prints "I am an Undergraduate" and "I am a Postgraduate" respectively. Call the method by creating an object of each of the three classes.**

**Program:**

class degree

{

void getdegree()

{

System.out.println("I got a degree");

}

}

class Undergraduate extends degree

{

void display()

{

System.out.println("I am an Undergraduate");

}

}

class Postgraduate extends degree

{

void display()

{

System.out.println("I am an Postgraduate");

}

}

class prac4\_5

{

public static void main(String arg[])

{

Undergraduate u = new Undergraduate();

Postgraduate p = new Postgraduate();

u.getdegree();

u.display();

p.getdegree();

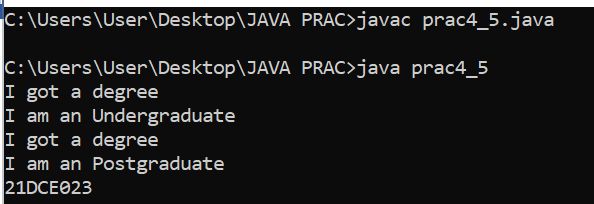
p.display();

System.out.println("21DCE066");

}

}

**Output:**

****

**Conclusion:**

By performing this practical we can conclude that by using the concept of inheritance and we can also call a class by creating different or separate objects for different classes and also we have learned the concept of “Method Overloading” that how we can call methods with the same name but with different arguments.

**PRACTICAL – 4.6**

**Aim:**

**Write a java that implements an interface AdvancedArithmetic which contains amethod signature int divisor\_sum(int n). You need to write a class calledMyCalculator 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:**

import java.util.\*;

class prac4\_6

{

public static void main(String ar[])

{

Scanner sc = new Scanner(System.in);

int n,sum;

System.out.println("Enter a no. ");

n=sc.nextInt();

MyCalculator m1 = new MyCalculator();

sum=m1.divisor\_sum(n);

System.out.println("Sum of divisors of "+n+" is "+sum);

}

}

interface AdvancedArithmetic

{

public int divisor\_sum(int n);

}

class MyCalculator implements AdvancedArithmetic

{

public int divisor\_sum(int n)

{

int sum=0,i;

if(n>1000)

{

return 0;

}

else

{

for(i=1;i<=n;i++)

{

if(n%i==0)

{

sum=sum+i;

}

}

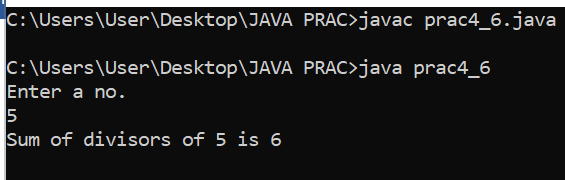
return (sum);

}

}

}

**Output:**



**Conclusion:**

By performing this practical we have learned the concept of “interface” keyword that allow us to do “Multilevel Inheritance” because java doesn’t support multilevel inheritance but with the help of “interface” keyword we can perform multilevel inheritance

**PRACTICAL – 4.7**

**Aim:**

**Assume you want to capture shapes, which can be either circles (with a radiusand 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:**

import java.util.\*;

class Signs

{

double area;

String color;

String text;

void get()

{

Scanner sc=new Scanner(System.in);

System.out.println("Enter color: ");

color=sc.nextLine();

System.out.println("Text: ");

text=sc.nextLine();

}

}

interface Shapes

{

void input();

void output();

}

class Circles extends Signs implements Shapes

{

double r;

public void input()

{

System.out.println("Enter radius: ");

Scanner sc=new Scanner(System.in);

r=sc.nextDouble();

area=3.14\*r\*r;

sc.close();

}

public void output()

{

System.out.println("CIRCLE ");

System.out.println("Text: "+text);

System.out.println("Color: "+color);

System.out.println("Radius: "+r);

System.out.println("Area: "+area);

}

}

class Rectangles extends Signs implements Shapes

{

double l,b;

public void input()

{

System.out.println("Enter Length and Breadth: ");

Scanner sc=new Scanner(System.in);

l=sc.nextDouble();

b=sc.nextDouble();

area=l\*b;

sc.close();

}

public void output()

{

System.out.println("RECTANGLE ");

System.out.println("Text: "+text);

System.out.println("Color: "+color);

System.out.println("Length: "+l);

System.out.println("Breadth: "+b);

System.out.println("Area: "+area);

}

}

public class Prac4\_7

{

public static void main(String[] args) {

int a;

Scanner sc=new Scanner(System.in);

System.out.println("Enter\nyour choice\n1.Circle\n2.Rectangle");

a=sc.nextInt();

switch(a)

{

case 1:

Circles c=new Circles();

c.get();

c.input();

c.output();

break;

case 2:

Rectangles d=new Rectangles();

d.get();

d.input();

d.output();

break;

default:

System.out.println("Wrong Input");

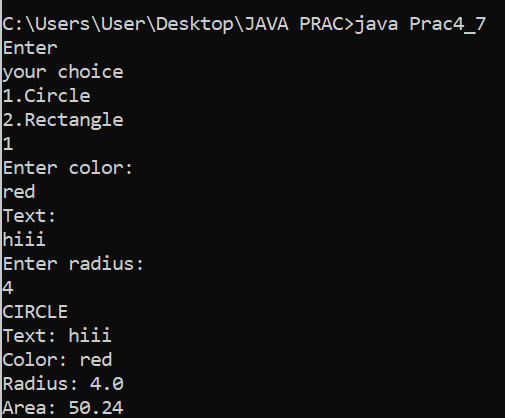
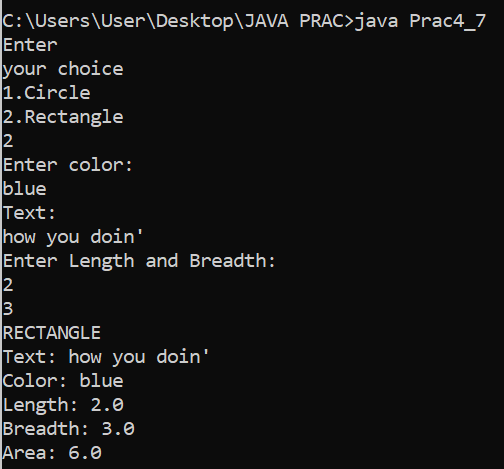
}

System.out.println("21DCE066");

}

}

**OUTPUT:**



**Conclusion:**

By performing this practical we learnt about the concept of interfaces. Interfaces shows full abstraction, how interfaces achieve multiple inheritance and variable declared inside interface are static by default.

**PRACTICAL – 4.8**

**Aim:**

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

**Program:**

package prac4\_8;

public class Prac4\_8 {

    public static void main(String[] args) {

              howyoudoin.hello();

    }

}

package prac4\_8;

public class howyoudoin {

    public static void hello(){

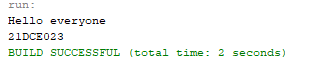
        System.out.println("Hello everyone");

         System.out.println("21DCE066");

    }

}

**Output:**



**Conclusion:**

By performing this practical we learnt how we can create a package simply by importing classes from other user defined packages.

**PART-V Exception Handling**

**PRACTICAL-5.1**

**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 :**

import java.util.\*;

public class prac5\_1

{

public static void main(String ar[])

{

int a,b,c;

Scanner sc = new Scanner(System.in);

try

{

a=sc.nextInt();

b=sc.nextInt();

c=a/b;

System.out.println(c);

}

catch(Exception e)

{

System.out.println(e);

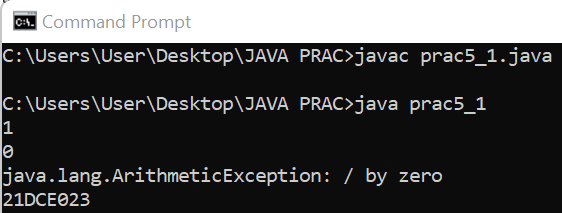
}

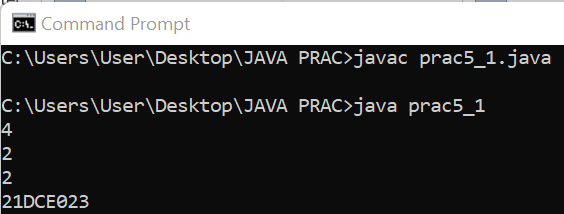
System.out.println("21DCE066");

}

}

**Output:**

****

****

**Conclusion:**

* From this practical, we have learned the concept of Exception by taking a simple example of ArithmaticException.

**PRACTICAL-5.2**

**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".**

**tring parsed to a numeric variable: Print "Format mismatch".**

**ccessing an invalid index in string: Print "Index is invalid".**

**ccessing an invalid index in array: Print "Array index is invalid".**

**MyException: This is a user defined Exception which you need to create. t takes a parameter param. When an exception of this class is encountered, he 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 bove ones fall this category.hint "Exception encountered". Finally, after the exception is handled,print "Exception Handling Completed".**

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

**Program:**

import java.util.\*;

public class Prac5\_2

{

public static void main(String[] args) {

Scanner s=new Scanner (System.in);

Scanner sc=new Scanner (System.in);

try{

try{

int x=10/0;

}

catch(ArithmeticException e)

{

System.out.println("Invalid division");

}

try{

System.out.println("Enter an integer:");

int y=s.nextInt();

s.close();

}

catch(InputMismatchException e)

{

System.out.println("Index is Invalid");

}

try{

System.out.println("Enter a String:");

String z=sc.nextLine();

sc.close();

}

catch(InputMismatchException e)

{

System.out.println("Index is Invalid");

}

try{

int w[]=new int[5];

System.out.println(w[10]);

}

catch(ArrayIndexOutOfBoundsException e)

{

System.out.println("Array Index is Invalid");

}

}

catch(Exception e)

{

System.out.println("Exception Encountered.");

}

finally

{

System.out.println("Exception Handling Completed");

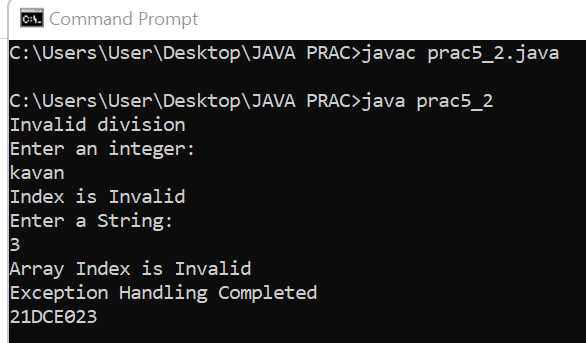
}

System.out.println("21DCE066");

}

}

**Output:**

****

**Conclusion :**

* From this practical , we have learned the concept of multiple try – catch block or say nested try-catch block.

**PRACTICAL-5.3**

**AIM: Write a java program to generate user defined exception using "throw" and "throws" keyword.**

**Also Write a java that differentiates checked and unchecked exceptions. (Mention at least two checked and two unchecked exception in program).**

**PROGRAM:**

import java.util.\*;

class Prac5\_3

{

Scanner sc=new Scanner(System.in);

int check\_exception() throws ArithmeticException

{

int x, y, z;

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

x = sc.nextInt();

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

y = sc.nextInt();

z = x / y;

return z;

}

public void checkAge(int age)

{

if(age<18)

{

System.out.println("Not eligible for voting.");

}

else

{

System.out.println("Eligible for voting.");

}

}

public static void main(String[] args)

{

Prac5\_3 obj=new Prac5\_3();

//throws

try

{

obj.check\_exception();

}

catch(ArithmeticException e)

{

System.out.println("Exception occurred.");

}

//throw

int age;

Scanner sc1=new Scanner(System.in);

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

age=sc1.nextInt();

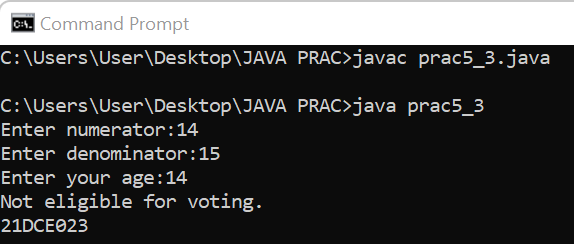
obj.checkAge(age);

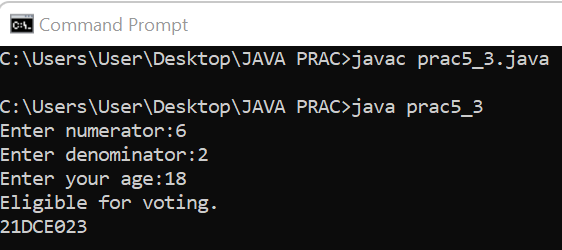
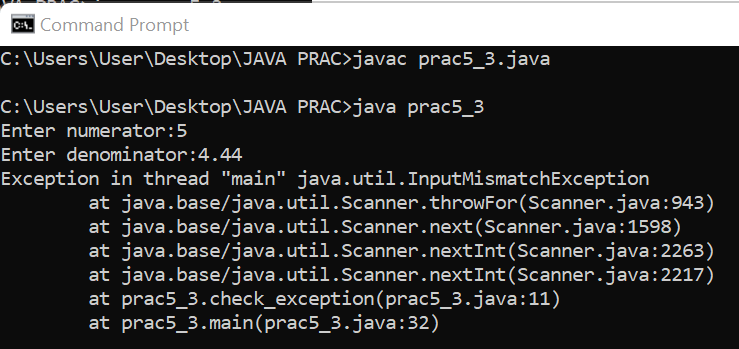
System.out.println("21DCE066");

}

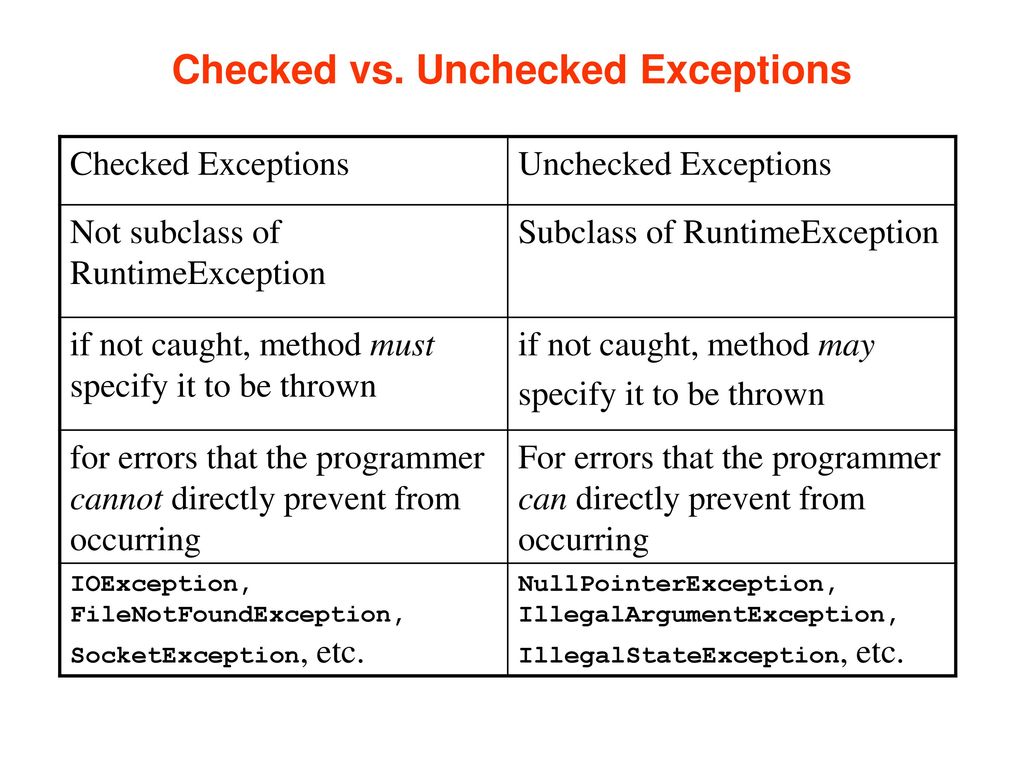
}

**Output:**



* **Difference Between Checked And Unchecked Exception :**



**Conclusion:**

* From this practical , we have learned about checked and unchecked exceptions that occurs during compile time and runtime respectively.

## PART-VI File Handling & Streams

## PRACTICAL-6.1

**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 Line Counts 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:**

import java.util.\*;

import java.io.\*;

class pra6\_1{

public static void main(String a[])

{

int x;

for(x=0;x<a.length;x++)

{

int y=0;

try

{

File fileobj = new File(a[x]);

if(fileobj.exists())

{

System.out.println("Searching for "+a[x]+"...");

System.out.println(a[x]+" Found");

}

Scanner sc = new Scanner(fileobj);

while(sc.hasNextLine())

{

y++;

sc.nextLine();

}

System.out.println("They are "+y+" lines in" +fileobj.getName()+ "\n");

sc.close();

}

catch(IOException e)

{

System.out.println("An erreor occured while finding "+a[x]+"file");

e.printStackTrace();

}

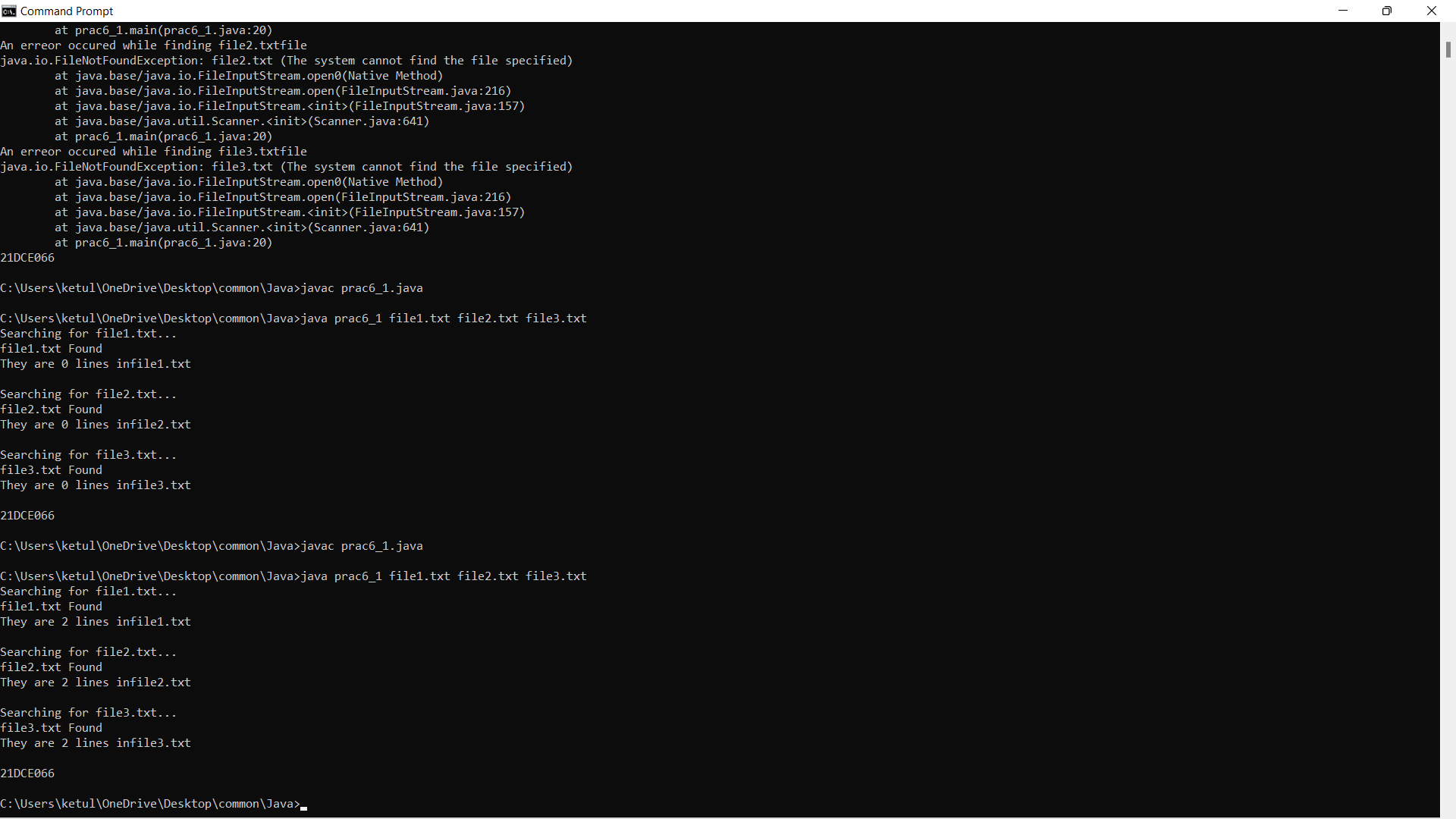
}

System.out.println("21DCE066");

}

}

**Output:**



**Conclusion:** By this practical we learned the concept of file handling and for counting the line in this program we used hasNextline.

**PRACTICAL 6.2**

**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:**

import java.util.\*;

import java.io.FileInputStream;

import java.io.BufferedInputStream;

public class prac6\_2

{

public static void main(String arg[])

{

try

{

FileInputStream fis = new FileInputStream("file1.txt");

BufferedInputStream bis = new BufferedInputStream(fis);

int i;

int c=0;

while((i=fis.read())!=-1)

{

char a = (char)i;

if(a==arg[0].charAt(0))

{

c++;

}

}

System.out.println("Number of "+arg[0]+" is present "+c+" times in the file");

}

catch(Exception e)

{

System.out.println("Character not found");

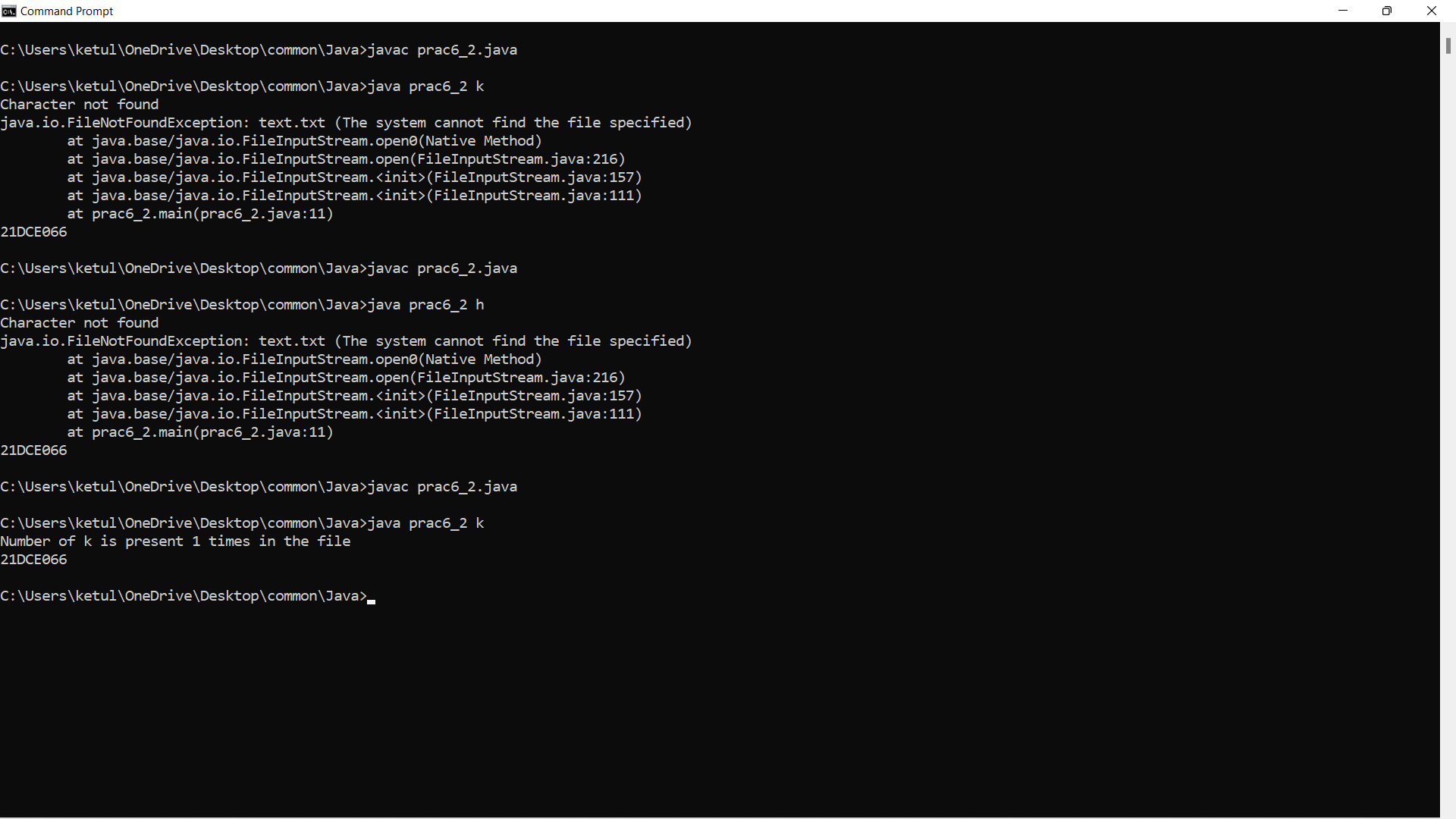
e.printStackTrace();

}

System.out.println("21DCE066");

}}

**Output:**



**Conclusion:** By this practical we learned how to count the character from the input file we’ve created in text mode.

**PRACTICAL 6.3**

**AIM:**

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

**Program:**

import java.io.\*; class prac6\_3

{

public static void main(String a[]) throws IOException

{

File f1= new File("63.txt");

String[] words=null;

FileReader fr =new FileReader(f1);

BufferedReader br=new BufferedReader(fr);

String s;

String input="ketul"; int count=0;

while((s=br.readLine())!=null)

{

words=s.split(" ");

for(String word: words)

{

if(word.equals(input))

{

count++;

}

}

}

if(count!=0) //check count not equal to zero

{

System.out.println("The given word is present for " +count + " times in the file");

}

else

{

System.out.println("The given word is not present in the file" );

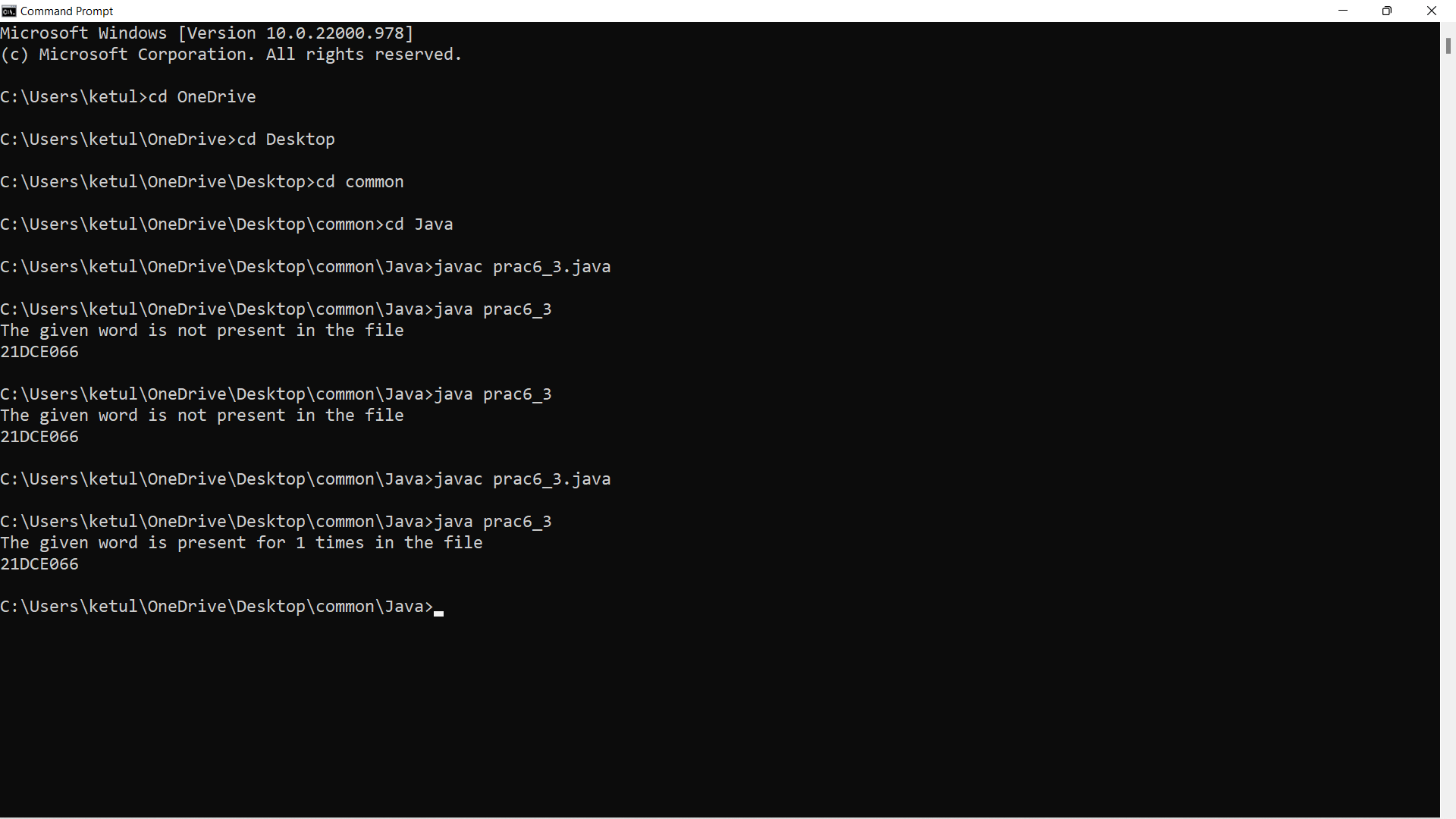
}

System.out.println(“21DCE066”);

}

}

**Output:**



**Conclusion:** by this practical learned the concept file reader and buffer reader of file handling concepts and how to search the word we have input, from the file.

**Wrapper class:**

public class wrapper

{

public static void main(String a[])

{

byte b=10;

short s=20;

int i=22;

long l=29;

float f=23.5f;

double d=60.0d;

char c='f';

Byte bobj=b;

Short shobj=s;

Integer inobj=i;

Long lobj=l;

Float flobj=f;

Double dobj=d;

Character cobj=c;

System.out.println("---Printing object valuse---");

System.out.println("Byte object: " + bobj);

System.out.println("Short object: " + shobj);

System.out.println("Int object: " + inobj);

System.out.println("Long object: " + lobj);

System.out.println("Float object: " + flobj);

System.out.println("Double object: " + dobj);

System.out.println("Character object: " + cobj);

byte bytevalue=bobj;

short shortvalue=shobj;

int intvalue=inobj;

long longvalue=lobj;

float floatvalue=flobj;

double doublevalue= dobj;

char charvalue=cobj;

System.out.println("------------------------------------");

System.out.println("---Printing primitive valuse---");

System.out.println("Byte value: " + bytevalue);

System.out.println("Short value: " + shortvalue);

System.out.println("Int value: " + intvalue);

System.out.println("Long value: " + longvalue);

System.out.println("Float value: " + floatvalue);

System.out.println("Double value: " + doublevalue);

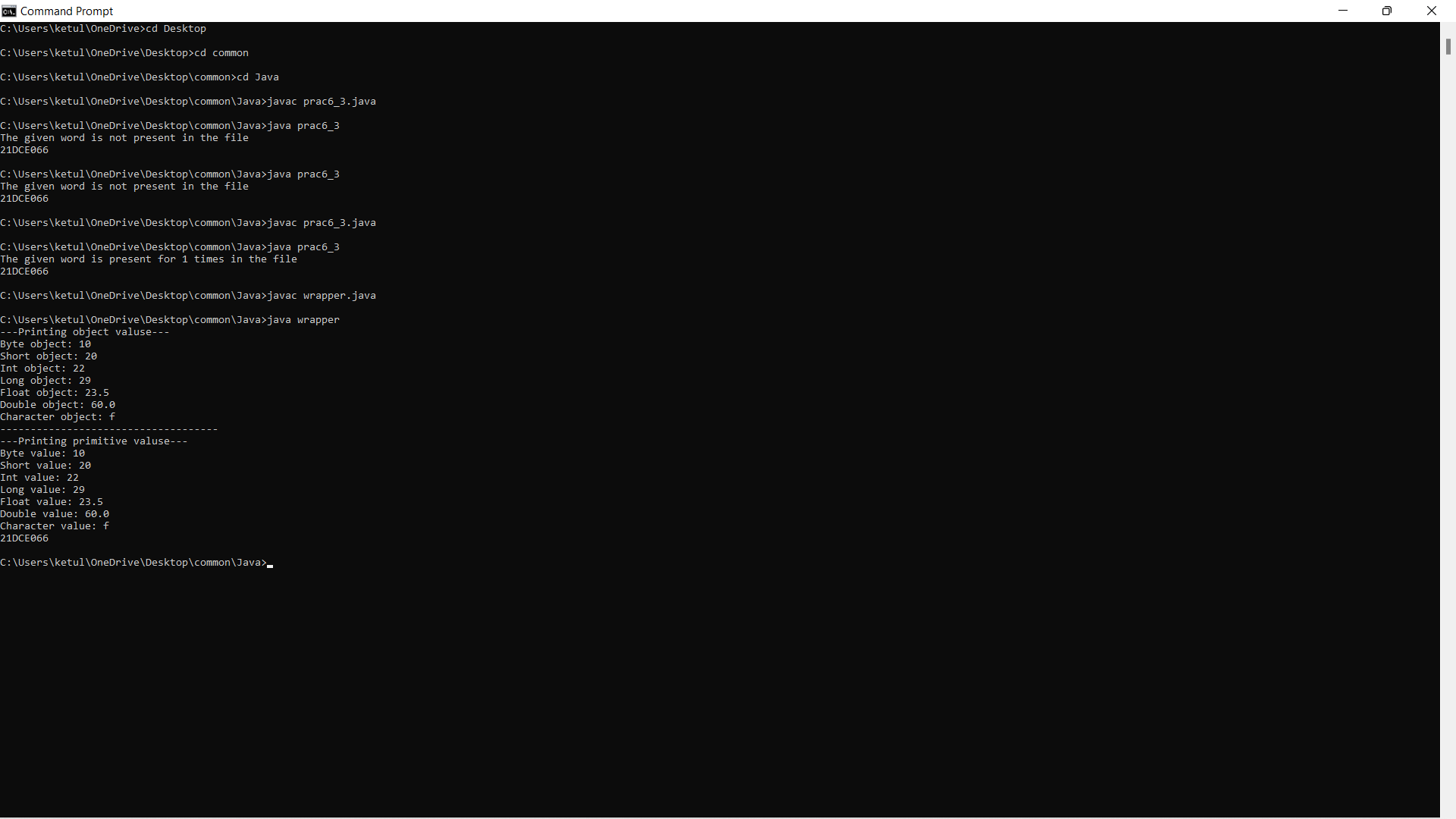
System.out.println("Character value: " + charvalue);

System.out.println("21DCE066");

}

}

**OUTPUT:**



**Conclusion:** In this practical I learned the concept of wrapper class and also its implementation for different types of data types.

**PRACTICAL 6.4**

**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:**

import java.util.\*;

import java.io.\*;

class prac6\_4 {

public static void main(String a[])

throws FileNotFoundException, IOException

{

Scanner sc =new Scanner(System.in);

System.out.print("Source file:");

String sfile=sc.next();

System.out.print("Destination file:");

String dfile=sc.next();

FileReader fin= new FileReader(sfile);

FileWriter fout = new FileWriter(dfile,true);

int c;

while((c=fin.read())!=-1)

{

fout.write(c);

}

System.out.println("Copying Finished...!");

System.out.println("21DCE066");

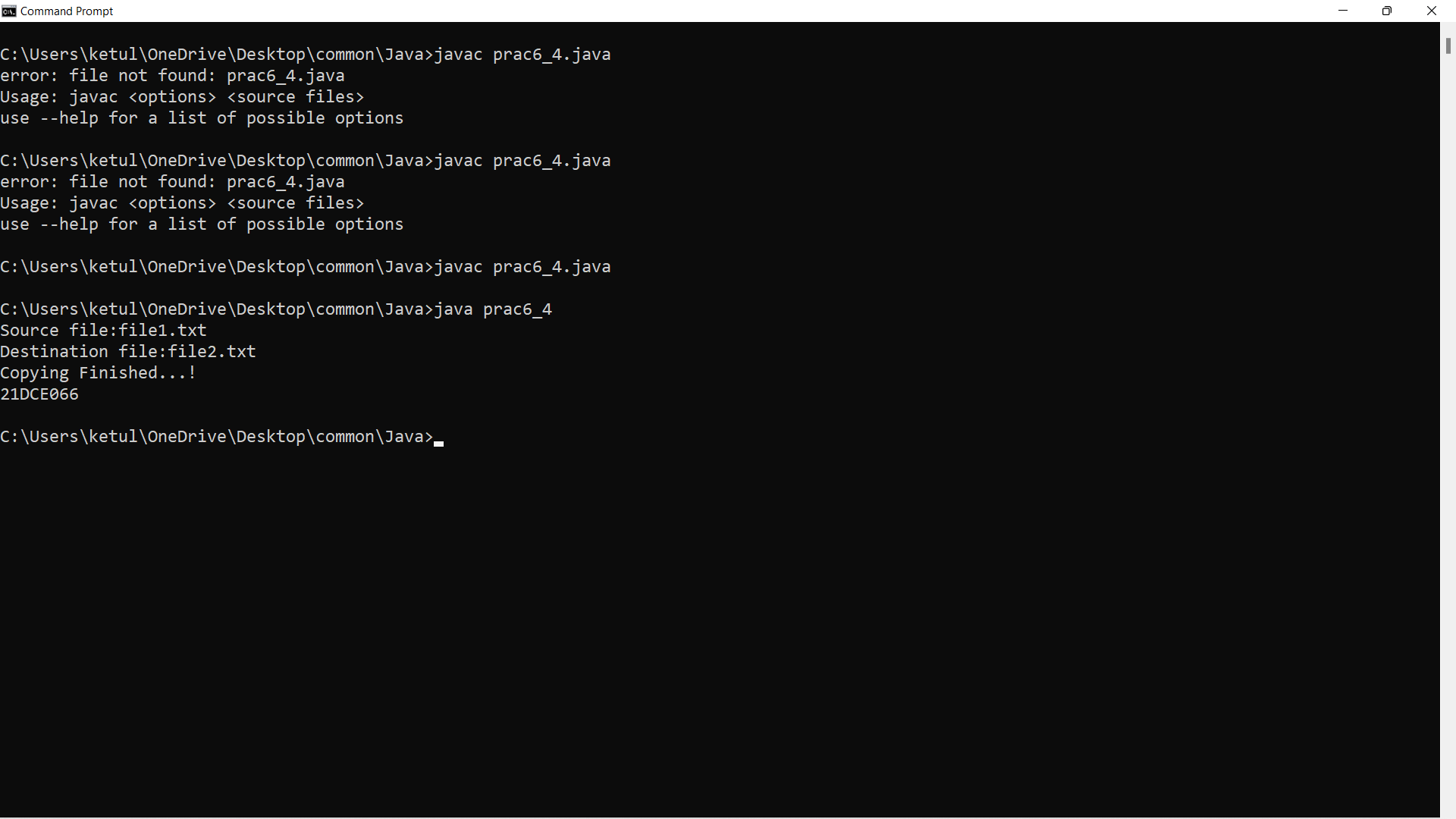
fin.close();

fout.close();

}

}

**OUTPUT:**



**Conclusion**: By this practical I learned how to copy text from source file to destination file using concept of file handling.

**AIM:**

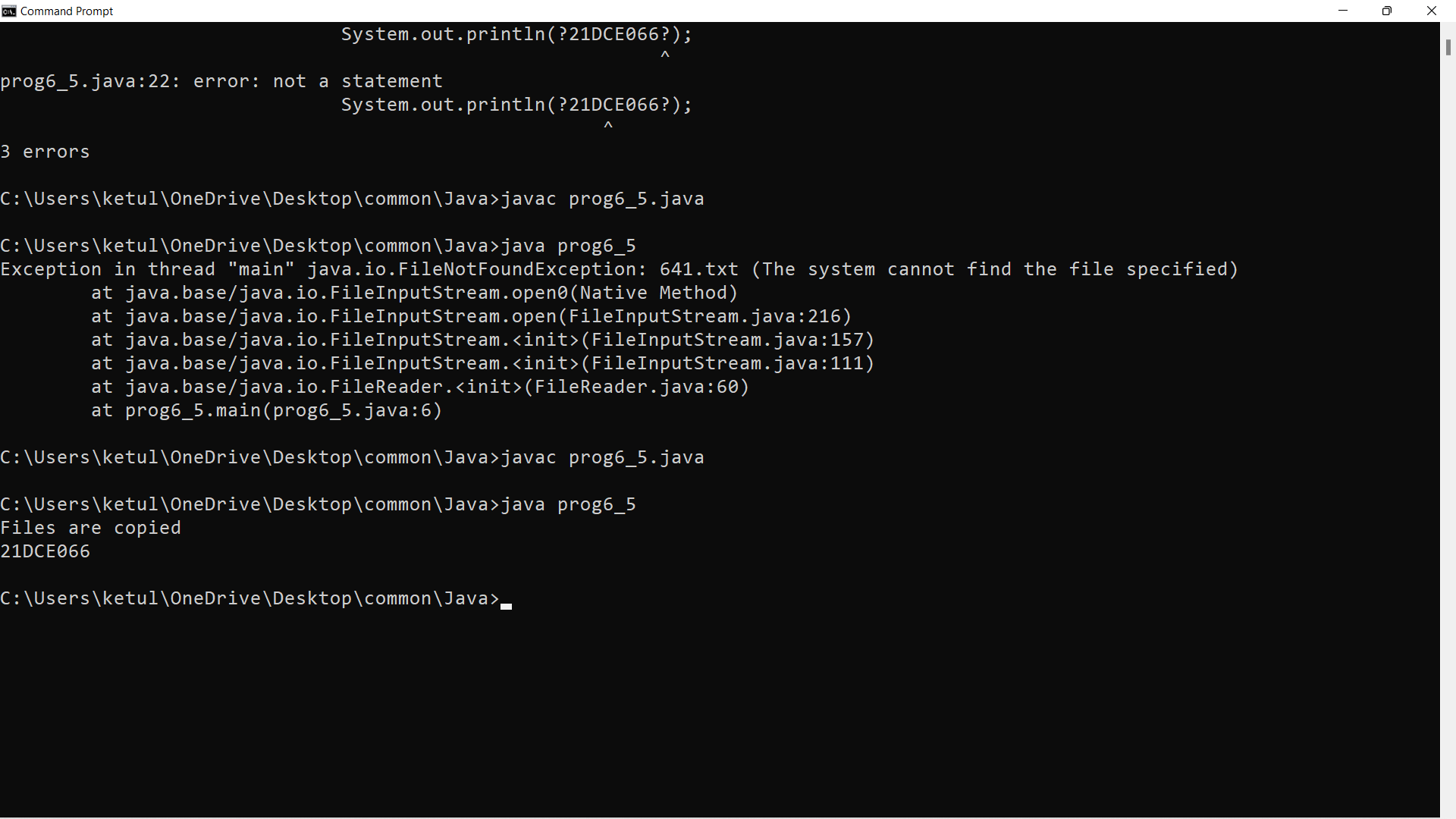
**Write a program to show use of character and byte stream. Also show use of BufferedReader/BufferedWriter to read console input and write them into a file.**

**Program:**

**Character Stream:**

|  |
| --- |
| import java.io.\*;  public class prog6\_5  {  public static void main(String args[]) throws IOException  {  FileReader fin = new FileReader("file1.txt");  FileWriter fout = new FileWriter("file2.txt");    try  {  int c;  while ((c = fin.read()) != -1)  {  fout.write(c);  } }  catch(Exception e)  {  System.out.println(e);  }  System.out.println("Files are copied");  System.out.println(“21DCE066”);  fin.close();  fout.close();  }  } |

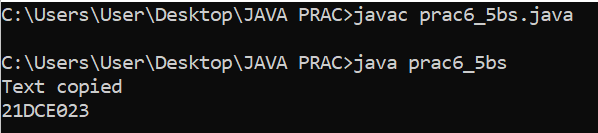
**OUTPUT:**



**Byte Stream:**

|  |
| --- |
| import java.io.\*;  public class prac6\_5bs  {  public static void main(String args[]) throws IOException  {  FileReader fin = new FileReader("file1.txt");  Bufferedreader fout = new BufferedReader(“fin”);    try  {  int c;  while ((c = fin.read()) != -1)  {  fout.write(c);  } }  catch(Exception e)  {  System.out.println(e);  }  System.out.println(“21DCE066”);  fin.close();  fout.close();  }  } |

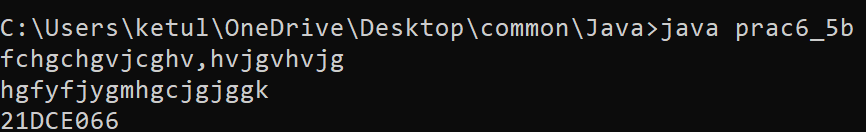
**OUTPUT:**

****

**Buffer Reader:**

|  |
| --- |
| import java.io.BufferedReader; import java.io.FileReader;  import java.io.IOException;    public class prac6\_5br {  public static void main(String[] args) { try{  FileReader fr= new FileReader("file1.txt");  BufferedReader br= new BufferedReader(fr);  String line="";  while((line=br.readLine())!=null){  System.out.println(line);  }  }  System.out.println(“21DCE066”);  catch(IOException e){  e.printStackTrace();  }  }  } |

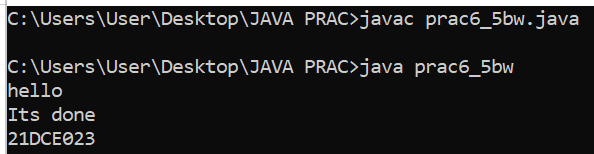
**Output:**

****

**BufferWriter:**

|  |
| --- |
| import java.io.BufferedWriter; import java.io.FileWriter; import java.io.IOException; import java.util.Scanner;  public class prac6\_5bw {  public static void main(String[] args) { try{  FileWriter fw=new FileWriter("file1.txt");  BufferedWriter bw=new BufferedWriter(fw);  Scanner sc=new Scanner(System.in);  String s= sc.nextLine();  System.oyt.println("Its done");  bw.write(s);  bw.close();  }  catch(IOException e){ e.printStackTrace();  }  System.out.println(“21DCE066”);  }  } |

**Output**:



**PART-VII Multithreading**

**PRACTICAL 7.1**

**AIM:**

**Write a program to create thread which display “Hello World” message. A. by extending Thread class B. by using Runnable interface.**

**PROGRAM using Thread:**

public class prac7\_1a extends Thread {

public void run()

{

System.out.println("Hello world");

System.out.println("21DCE066");

}

public static void main(String[] args) {

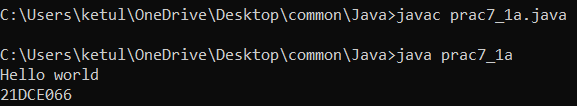
prac7\_1a k = new prac7\_1a();

k.start();

}

}

**OUTPUT SCREENSHOT using Thread:**



**PROGRAM using runnable interface:**

public class prac7\_1b implements Runnable{

public void run(){

System.out.println("Hello world");

System.out.println("21DCE066");

}

public static void main(String[] args) {

prac7\_1b p = new prac7\_1b();

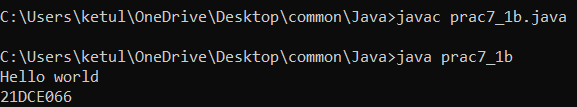
Thread t =new Thread(p);

t.start();

}

}

**OUTPUT SCREENSHOT using rummable interface:**



**CONCLUSION:**

In this practical we learn how to implement the Thread using runnable interface and Thread.

**PRACTICAL 7.2**

**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:**

import java.util.\*;

class mythread extends Thread{

int n;

int nt;

int sum = 0;

int th[] = new int[200];

mythread(int a,int b,int[] th1){

n = a;

nt = b;

th = th1;

}

public void run(){

try{

for(int i=1;i<nt;i++){

sum = sum + th[i];

System.out.println("threads are as follows");

System.out.println(th[i]);

}

System.out.println("21DCE066");

}

catch(Exception e){

e.getMessage();

}

}

}

public class prac7\_2 {

public static void main(String[] args) {

try{

Scanner sc = new Scanner(System.in);

int n;

System.out.println("enter the number");

n = sc.nextInt();

int nt;

System.out.println("enter the number of threads");

nt = sc.nextInt();

int sum=0;

int th[] = new int[1000];

th[1] = n/nt;

for(int i=1;i<nt;i++){

th[i] = th[1];

sum = sum + th[i];

}

th[nt] = n-sum;

mythread m1 =new mythread(n,nt,th);

m1.start();

}

catch(Exception e){

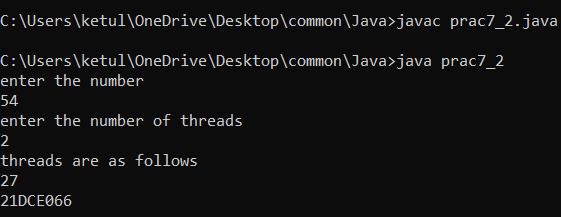
e.getMessage();

}

}

}

**OUTPUT:**

****

**CONCLUSION:**

By performing this practical we learnt about the distribution of the threads and the tasks to reduce the load from the code.

**PRACTICAL 7.3**

**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:**

import java.util.Random;

import java.lang.Thread;

class prac7\_3 {

public static void main(String[] args) {

randomno r = new randomno();

r.start();

System.out.println("21DCE066");

}

}

class randomno extends Thread

{

public void run()

{

for(int i=0;i<5;i++)

{

Random k = new Random();

int y = k.nextInt(100);

System.out.println("The generated no. is "+y);

try

{

Thread.sleep(1000);

}

catch(Exception e)

{

System.out.println(e);

e.printStackTrace();

}

if(y%2==0)

{

Square s = new Square(y);

s.start();

}

else

{

Cube c = new Cube(y);

c.start();

}

}

}

}

class Square extends Thread

{

int d;

Square(int b)

{

d=b;

}

public void run()

{

try

{

System.out.println("Square of no. is "+d\*d);

}

catch(Exception e)

{

System.out.println(e);

}

}

}

class Cube extends Thread

{

int v;

Cube(int a)

{

v=a;

}

public void run()

{

try

{

System.out.println("Cube of no. is "+v\*v\*v);

}

catch(Exception e)

{

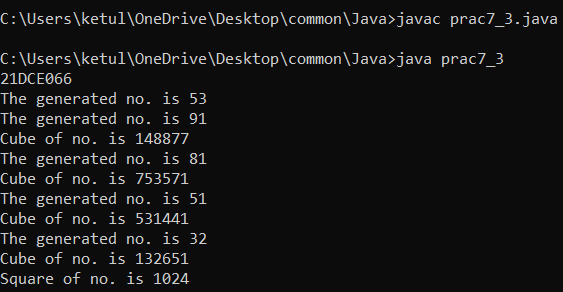
System.out.println(e);

}

}

}

**OUTPUT:**

****

**CONCLUSION:**

In this practical we learnt the usage of multithreading and performing the tasks depending on the arguments.

**PRACTICAL 7.4**

**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:**

import java.util.\*;

import java.lang.\*;

class prac7\_4 extends Thread {

public void run()

{

int x;

Scanner sc = new Scanner(System.in);

System.out.println("Enter a value");

x=sc.nextInt();

x++;

try{

Thread.sleep(10000);

System.out.println(x);

}

catch(InterruptedException e){

System.out.println(e);

}

System.out.println("21DCE066");

}

public static void main(String[] args) {

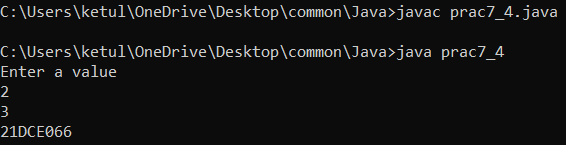
prac7\_4 t = new prac7\_4();

t.start();

}

}

**OUTPUT:**



**CONCLUSION:**

By performing this practical we learnt the concept of thread sleep and implementing it to hold the thread process for determined task.

**PRACTICAL 7.5**

**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:**

import java.lang.\*;

public class prac7\_5 extends Thread{

public static void main(String[] args) {

prac7\_5 t1 = new prac7\_5();

prac7\_5 t2 = new prac7\_5();

prac7\_5 t3 = new prac7\_5();

System.out.println("Convert Priority");

System.out.println("Thread 1 "+t1.getPriority());

System.out.println("Thread 2 "+t2.getPriority());

System.out.println("Thread 3 "+t3.getPriority());

t1.setPriority(3);

t2.setPriority(5);

t3.setPriority(7);

System.out.println("After Priority");

System.out.println("Thread 1 "+t1.getPriority());

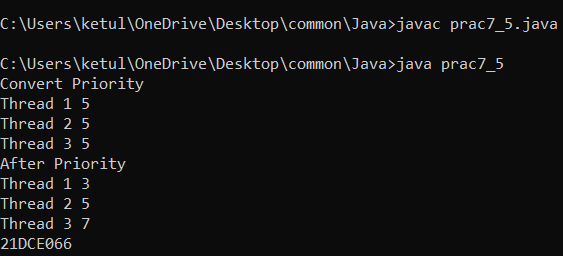
System.out.println("Thread 2 "+t2.getPriority());

System.out.println("Thread 3 "+t3.getPriority());

System.out.println("21DCE066");}

}

**OUTPUT:**

****

**CONCLUSION:**

In this practical we learnt the concept of thread.setPriority which can change or set the priority of the thread.

**PRACTICAL 7.6**

**AIM:**

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

**PROGRAM:**

import java.util.ArrayList;

class prac7\_66{

    public static void main(String[] args) throws InterruptedException {

        final ProducerConsumer p1 = new ProducerConsumer();

        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 ProducerConsumer {

            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 consumed Product :" +val);

                        notify();

                        Thread.sleep(1000);

                    }

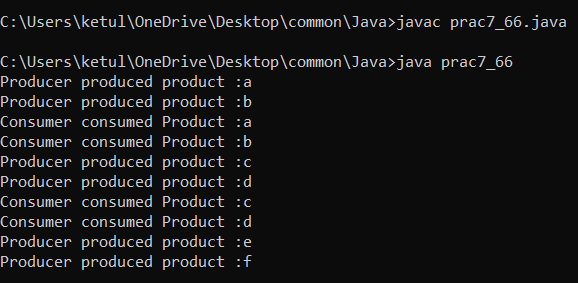
                }

            }

        }

        }

**OUTPUT:**



**CONCLUSION:**

In this practical we learnt the synchronized of the thread and implementing to the example.

**PART-VIII Collection Framework and Generic**

**PRACTICAL 8.1**

**AIM:**

**Design a Custom Stack using ArrayList class, which implements following Functionalities of stack. My Stack**

**-list ArrayList<Object>: A list to store elements.**

**+isEmpty: boolean: Returns true if this stack is empty.**

**+getSize(): int: Returns number of elements in this stack.**

**+peek(): Object: Returns top element in this stack without removing it.**

**+pop(): Object: Returns and Removes the top elements in this stack.**

**+push(o: object): Adds new element to the top of this stack.**

**PROGRAM:**

import java.util.\*;

class mystack{

    ArrayList<Object>l;

    String s;

    mystack(Object element[]){

        l = new ArrayList<Object>();

        for(int i=0;i<element.length;i++){

            l.add(element[i]);

        }

    }

    boolean isEmpty(){

        return l.isEmpty();

    }

    int getsize(){

        return l.size();

    }

    Object peek(){

        return l.get(l.size()-1);

    }

    Object pop(){

        return l.remove(l.size()-1);

    }

    void push(Object o){

        l.add(o);

    }

    void print(){

        for(int i=0;i<l.size();i++){

            System.out.println(l.get(i));

        }

    }

}

public class prac8\_1 {

    public static void main(String[] args) {

        Integer arr[] = new Integer[]{11,2,31};

        mystack s = new mystack(arr);

        s.push(6);

        s.push(18);

        System.out.println("the stack is ");

        s.print();

        System.out.println("Size of the stack "+s.getsize());

        System.out.println("Is List Empty- "+s.isEmpty());

        System.out.println("The last added(peek/top) element is "+s.peek());

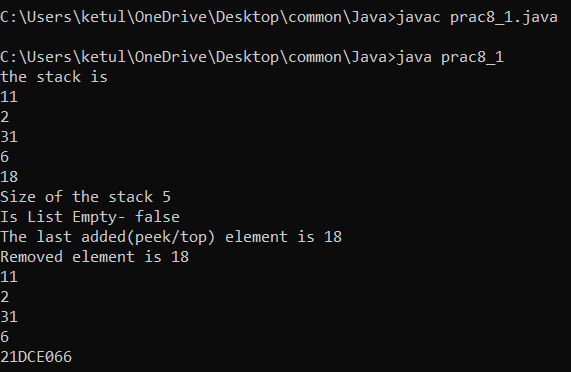
        System.out.println("Removed element is "+s.pop());

        s.print();

        System.out.println("21DCE066");}

}

**OUTPUT:**

****

**CONCLUSION:**

In this practical we learn the concept of Array List and LI-FO concept and to remove element from the stack.

**PRACTICAL 8.2**

**AIM:**

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

**PROGRAM:**

import java.util.\*;

class Student implements Comparable<Student>{

    String Name;

    int javaMarks;

    Student() {

        Name = null;

        javaMarks = 0;

    }

    Student(String name,int javaMarks) {

        this.Name = name;

        this.javaMarks = javaMarks;

    }

    public int compareTo(Student s){

        return this.javaMarks - s.javaMarks;

    }

    public String toString() {

        return String.format("[%s, %d]", Name, javaMarks);

    }

}

public class prac8\_2{

        public static void main(String[] args){

        Student[] s = new Student[4];

        s[0] = new Student("Mind",68);

        s[1] = new Student("your",54);

        s[2] = new Student("own",99);

        s[3] = new Student("Code",83);

        System.out.println("Before sorting: " + Arrays.toString(s));

        System.out.println("");

        Arrays.sort(s);

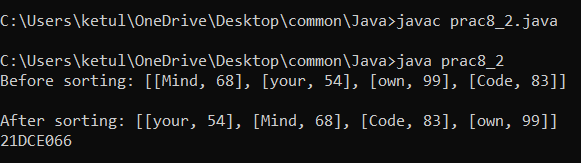
        System.out.println("After sorting: " + Arrays.toString(s));

        System.out.println("21DCE066");

    }

}

**OUTPUT:**

****

**CONCLUSION:**

From this practical we learnt sorting of array and format of the String.

**PRACTICAL 8.3**

**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:**

import java.util.\*;

class prac8\_3{

    public static void main(String args[]){

        Map<String, Integer>map = new HashMap<>();

        Scanner sc = new Scanner(System.in);

        System.out.println("Enter a string:");

        String sentence = sc.nextLine();

        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) {

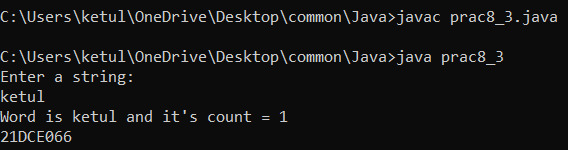
            System.out.println("Word is "+ str + " and it's count = " + map.get(str));

        }

        System.out.println("21DCE066");}

}

**OUTPUT:**

****

**CONCLUSION:**

From this practical we learnt the concept of Map and Set Classes to count the words in the sentence via map and tree set.

**PRACTICAL 8.4**

**AIM:**

**Write a code which counts the number of the keywords in a Java source file. Store all the keywords in a HashSet and use the contains () method to test if a word is in the keyword set.**

**PROGRAM:**

import java.util.\*;

import java.io.\*;

public class prac8\_4 {

    public static void main(String[] args) {

        Set<String> hash\_Set = new HashSet<String>();

        hash\_Set.add("abstract");

        hash\_Set.add("assert");

        hash\_Set.add("do");

        hash\_Set.add("boolean");

        hash\_Set.add("double");

        hash\_Set.add("break");

        hash\_Set.add("else");

        hash\_Set.add("byte");

        hash\_Set.add("enum");

        hash\_Set.add("case");

        hash\_Set.add("extends");

        hash\_Set.add("catch");

        hash\_Set.add("this");

        hash\_Set.add("throw");

        hash\_Set.add("throws");

        hash\_Set.add("try");

        hash\_Set.add("void");

        hash\_Set.add("volatile");

        hash\_Set.add("while");

        hash\_Set.add("String");

        int count = 0;

        ArrayList<String>keywords = new ArrayList<String>();

        try {

            File SourceFile= new File("file1.txt");

            Scanner myReader = new Scanner(SourceFile);

            while (myReader.hasNext()) {

                String word = myReader.next();

                if (hash\_Set.contains(word)) {

                    count += 1;

                    keywords.add(word);

                }

            }

            myReader.close();

        }

        catch (FileNotFoundException e)

        {

            e.getMessage();

        }

        finally {

            System.out.println("\nFounded Keywords in given file : "+ count+"\n");

            System.out.println(keywords);

            System.out.println("\nOperation Performed ! ");

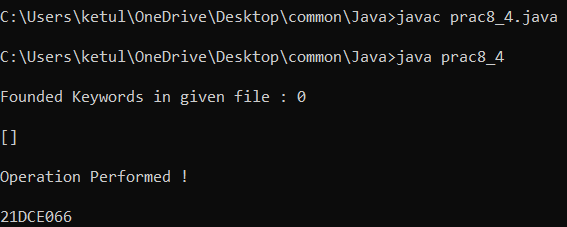
            System.out.println("\n21DCE066");

        }

        System.exit(0);}

}

**OUTPUT:**

****

**CONCLUSION:**

By performing this practical we learnt about the concept of Hash Set and Contains. By reading from the file the contain check the word from the predefined Hash set and gives the result of the words occurred from the Hash Set.