**CHAROTAR UNIVERSITY OF SCIENCE & TECHNOLOGY**

FACULTY OF TECHNOLOGY AND ENGINEERING

**Devang Patel Institute of Advance Technology & Research**

Semester: III

Academic year: 2019-20

**PART-I**

**Data Types, Variables, String, Control Statements, Operators, Arrays**

**PRACTICAL – 1**

**AIM:**

Introduction to Object Oriented Concepts, comparison of Java with other object oriented programming languages. Introduction to JDK, JRE, JVM, javadoc, command line argument.

Introduction to Eclipse or Netbean IDE and Console Programming.

**Object Oriented Programming (OOPs) Concept in Java**

Object-oriented programming: As the name suggests, Object-Oriented Programming or OOPs refers to languages that uses objects in programming. Object-oriented programming aims to implement real-world entities like inheritance, hiding, polymorphism etc in programming. The main aim of OOP is to bind together the data and the functions that operate on them so that no other part of the code can access this data except that function.

OOPs Concepts:

* Polymorphism
* Inheritance
* Encapsulation
* Abstraction
* Class
* Object
* Method
* Message Passing

**Comparison of Java with other Object Oriented Programming Languages**

|  |  |
| --- | --- |
| **Java** | **C/C++** |
| Java is platform independent | C/C++ is a platform dependent |
| It supports 50 keywords | It supports 32 keywords |
| Does not support operator overloading and multiple inheritance | It supports operator overloading and multiple inheritance |
| Does not support storage classes like auto, external etc. | It supports storage classes like auto, external etc. |

**JDK**

The Java Development Kit (JDK) is kit which provides the environment to develop and execute the java program. JDK is a kit which includes two things.

1. Development tools(to provide an environment to develop your java programs)
2. JRE (to execute your java programs).

**JRE**

A runtime environment is an installation package which provides environment to only run the java program onto your machine. JRE contains the Java class libraries, the Java class loader, and the Java Virtual Machine. In this system:

The class loader is responsible for correctly loading classes and connecting them with the core Java class libraries.

**JVM**

A Java virtual machine (JVM) is a very important part of both JRE and JDK because it is contained or inbuilt in both. Whatever java program you run using JRE and JDK goes into JVM and JVM is responsible for executing the java program line by line hence it is also known as interpreter.

**JAVADOC**

Javadoc is an extensible documentation generation system which reads specially formatted comments in java source code and generates compiled documentation. It is typically used to produce API documentation in the form of HTML web pages.

**Command Line Argument**

Command line argument is the argument passed to a program at the time when you run it. To access the command line argument inside a java program is quite easy, they are stored as string array passed to the **args** parameter of main() method.

* Command Line Arguments can be used to specify configuration information while launching your application.
* There is no restriction on the number of java command line arguments. You can specify any number of arguments
* Information is passed as Strings.
* They are captured into the String args of your main method

**Introduction to Eclipse or Netbeans IDE and Console Programming:**

Netbeans and Eclipse IDEs are free, open source, that enables you to develop desktop, mobile and web applications.

A console application is a computer program designed to be used via a text-only computer interface, such as a text terminal.

**CONCLUSION:**

In this Practical I learnt about some basic concepts and terminologies of Java Programming.

**PRACTICAL - 2**

**AIM:**

Given a string, return a string made of the first 2 chars (if present), however include first char only if it is 'o' and include the second only if it is 'z', so "ozymandias" → "oz".

startOz("ozymandias") → "oz"

startOz("bzoo") → "z"

startOz("oxx") → "o"

**PROGRAM CODE:**

class p2

{

void startOZ(String str)

{

if(str.substring(0,1).contains("o"))

{

System.out.print("o");

}

if(str.substring(1,2).contains("z"))

{

System.out.print("z");

}

System.out.println("");

}

public static void main(String args[])

{

p2 p = new p2();

p.startOZ("ozymandias");

p.startOZ("oxx");

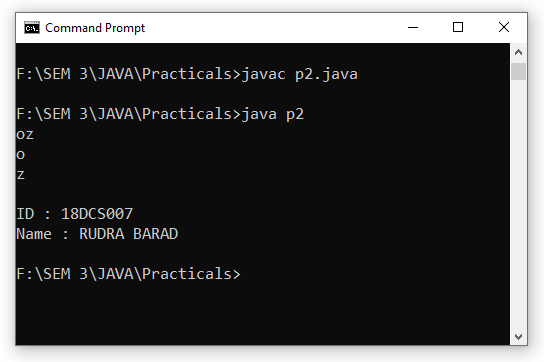
p.startOZ("azen");

System.out.println("\nID : 18DCS007\nName : RUDRA BARAD");

}

}

**OUTPUT:**



**CONCLUSION:**

In this Practical I learnt about substring() and contains() methods of String.

**PRACTICAL - 3**

**AIM:**

Given two non-negative int values, return true if they have the same last digit, such as with 27 and 57. Note that the % "mod" operator computes remainders, so 17 % 10 is 7.

lastDigit(7, 17) → true

lastDigit(6, 17) → false

lastDigit(3, 113) → true

**PROGRAM CODE:**

class p3

{

boolean lastDigit(int a,int b)

{

System.out.println("\nNumbers a and b are : "+a+" "+b);

a=a%10;

b=b%10;

System.out.println("Last Digit Of a : "+a);

System.out.println("Last Digit Of b : "+b);

return(a==b);

}

public static void main(String args[])

{

p3 p = new p3();

boolean b1 = p.lastDigit(7,17);

System.out.println(b1);

boolean b2 = p.lastDigit(6,17);

System.out.println(b2);

boolean b3 = p.lastDigit(3,113);

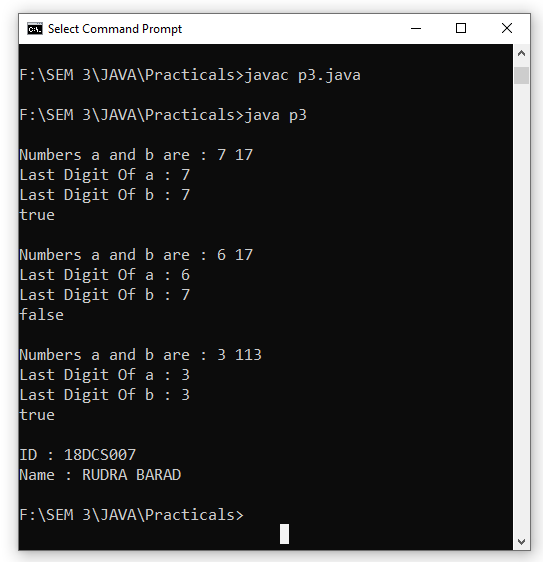
System.out.println(b3);

System.out.println("\nID : 18DCS007\nName : RUDRA BARAD");

}

}

**OUTPUT:**



**CONCLUSION:**

In this Practicals I incorporated some concepts of Java by constructing method of return type **boolean** and using different operators like Divide ( / ) and Modulus ( % ).

**PRACTICAL - 4**

**AIM:**

Given an array of int, return true if the sequence of numbers 1, 2, 3 appears in the array somewhere.

array123 ( [1, 1, 2, 3, 1] ) → true

array123 ( [1, 1, 2, 4, 1] ) → false

array123 ( [1, 1, 2, 1, 2, 3] ) → true

**PROGRAM CODE:**

class p4

{

boolean array123(int arr[])

{

boolean flag = false;

int l = arr.length;

for(int i=0;i<(l-1);i++)

{

if(arr[i]==1 && arr[i+1]==2 && arr[i+2]==3)

{ flag = true; }

} return flag;

}

public static void main(String args[])

{

p4 p = new p4();

int arr1[] = {1,1,2,3,1};

int arr2[] = {1,1,2,4,1};

int arr3[] = {1,1,2,1,2,3};

boolean b1 = p.array123(arr1);

System.out.println(b1);

boolean b2 = p.array123(arr2);

System.out.println(b2);

boolean b3 = p.array123(arr3);

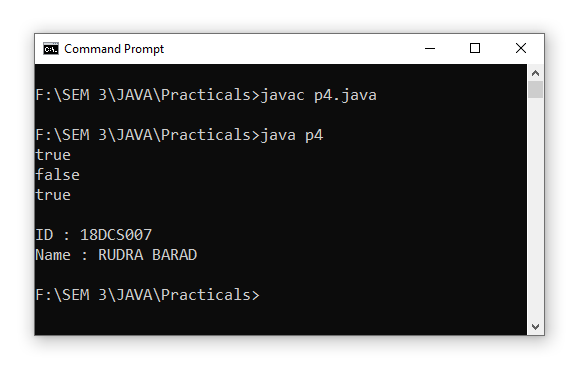
System.out.println(b3);

System.out.println("\nID : 18DCS007\nName : RUDRA BARAD");

}

}

**OUTPUT:**



**CONCLUSION:**

In this Practical we array as argument and some methods like length( ).

**PRACTICAL - 5**

**AIM:**

Given 2 strings, a and b, return the number of the positions where they contain the same length 2 substring. So "xxcaazz" and "xxbaaz" yields 3, since the "xx", "aa", and "az" substrings appear in the same place in both strings.

stringMatch("xxcaazz", "xxbaaz") → 3

stringMatch("abc", "abc") → 2

stringMatch("abc", "axc") → 0

**PROGRAM CODE:**

class p5

{

int stringMatch(String a,String b)

{

int i,l,count=0;

if(a.length() < b.length())

l=a.length();

else

l=b.length();

for(i=0;i<(l-1);i++)

{

String sub\_a = a.substring(i,i+2);

String sub\_b = b.substring(i,i+2);

if(sub\_a.equals(sub\_b))

count++;

}

return count;

}

public static void main(String args[])

{

p5 p = new p5();

int temp1 = p.stringMatch("xxcaaz","xxbaaz");

System.out.println(temp1);

int temp2 = p.stringMatch("abc","abc");

System.out.println(temp2);

int temp3 = p.stringMatch("abc","axc");

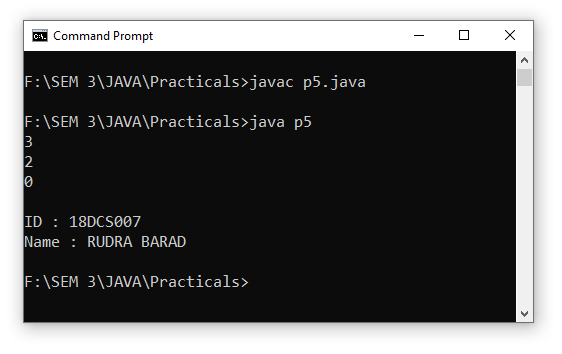
System.out.println(temp3);

System.out.println("\nID : 18DCS007\nName : RUDRA BARAD");

}

}

**OUTPUT:**



**CONCLUSION:**

In this Practical I learnt about different functions of String like length( ), substring( ), equals( ), etc.

**PRACTICAL - 6**

**AIM:**

Computing Body Mass Index

You can use nested if statements to write a program that interprets body mass index.

Body Mass Index (BMI) is a measure of health based on height and weight. It can be calculated by taking your weight in kilograms and dividing it by the square of your height in meters. The interpretation of BMI for people 20 years or older is as follows:

BMI Interpretation

BMI < 18.5 Underweight

18.5 ≤ BMI < 25.0 Normal

25.0 ≤ BMI < 30.0 Overweight

30.0 ≤ BMI Obese

Write a program that prompts the user to enter a weight in pounds and height in

inches and displays the BMI. Note that one pound is 0.45359237 kilograms and

one inch is 0.0254 meters. Listing 3.4 gives the program.

Output-

Enter weight in pounds: 146

Enter height in inches: 70

BMI is 20.948603801493316

Person is Normal

**PROGRAM CODE:**

import java.util.Scanner;

class p6

{

public static void main(String args[])

{

Scanner sc = new Scanner(System.in);

System.out.print("Enter the Weight in Pounds : ");

double weight = sc.nextDouble();

System.out.print("Enter the Height in Inches : ");

double height = sc.nextDouble();

double bmi;

bmi = (0.45359237\*weight)/((0.0254\*height)\*(0.0254\*height));

System.out.println("BMI is : "+bmi);

if(bmi<18.5)

System.out.println("\nPerson is UNDERWEIGHT");

else if(bmi>=18.5 && bmi<25.0)

System.out.println("\nPerson is NORMAL");

else if(bmi>=25.0 && bmi<30.0)

System.out.println("\nPerson is OVERWEIGHT");

else if(bmi>=30.0)

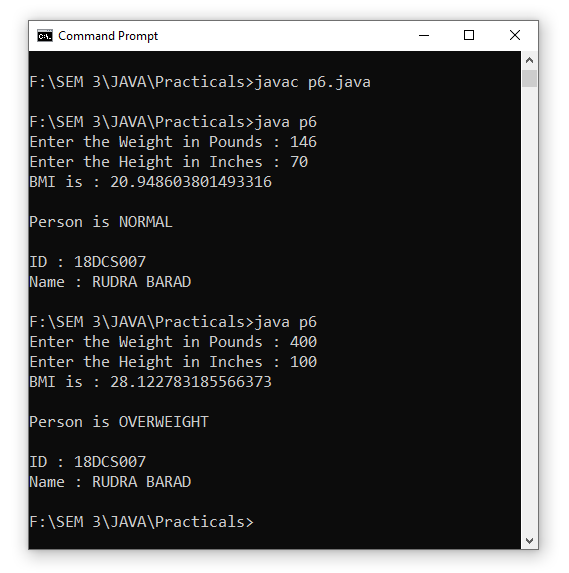
System.out.println("\nPerson is OBESE");

System.out.println("\nID : 18DCS007\nName : RUDRA BARAD");

}

}

**OUTPUT:**



**CONCLUSION:**

In this Practical I learnt how to import different classes and how to use Scanner Class for inputting values from user. I also used nested if statements for applying conditions.

**PRACTICAL - 7**

**AIM:**

Lottery

The lottery program involves generating random numbers, comparing digits, and using Boolean operators.

Suppose you want to develop a program to play lottery. The program randomly generates a lottery of a two-digit number, prompts the user to enter a two-digit number, and determines whether the user wins according to the following rules:

1. If the user input matches the lottery number in the exact order, the award is $10,000.

2. If all digits in the user input match all digits in the lottery number, the award is $3,000.

3. If one digit in the user input matches a digit in the lottery number, the award is $1,000.

Note that the digits of a two-digit number may be 0. If a number is less than 10, we assume the number is preceded by a 0 to form a two-digit number. For example, number 8 is treated as 08 and number 0 is treated as 00 in the program.

Listing 3.8 gives the complete program.

Output: Test Cases

Case-1 Enter your lottery pick (two digits): 15

The lottery number is 15

Exact match: you win $10,000

Case-2 Enter your lottery pick (two digits): 45

The lottery number is 54

Match all digits: you win $3,000

Case-3 Enter your lottery pick: 23

The lottery number is 34

Match one digit: you win $1,000

Case-4 Enter your lottery pick: 23

The lottery number is 14

Sorry: no match

**PROGRAM CODE:**

import java.util.\*;

class p7

{

public static void main(String args[])

{

Scanner sc = new Scanner(System.in);

System.out.print("Enter Your Lottery Ticket (2 Digit) : ");

int x = sc.nextInt();

Random rand = new Random();

int lot = rand.nextInt(100);

System.out.println("Lottery Number Is : "+lot);

if(x==lot)

{

System.out.println("\nCongratulations ! You got Exact Match. \nYou Won : $10,000 ");

}

else

{

int x1 = x/10;

int x2 = x%10;

int count = 0;

int l1 = lot/10;

int l2 = lot%10;

if(x1==l1)

count++;

else if(x1==l2)

count++;

else if(x2==l1)

count++;

else if(x2==l2)

count++;

if(count==1)

System.out.println("\nCongratulations ! You got 1 Match. \nYou Won : $1,000 ");

if(count==2)

System.out.println("\nCongratulations ! You Match 2 Digits. \nYou Won : $3,000 ");

else if(count==0)

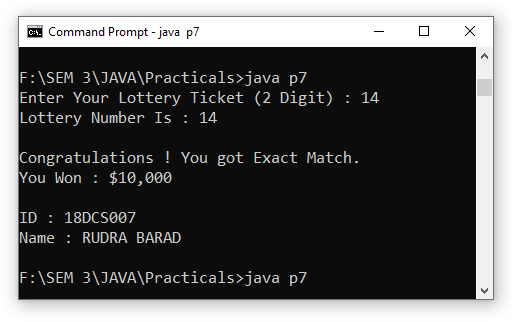
System.out.println("\nSorry ! You didn't got Any Match. \nYou Won : $00,000 ");

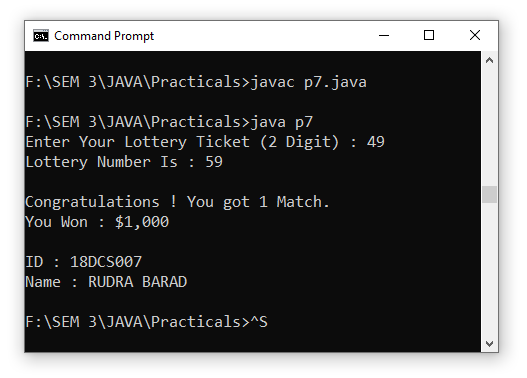
}

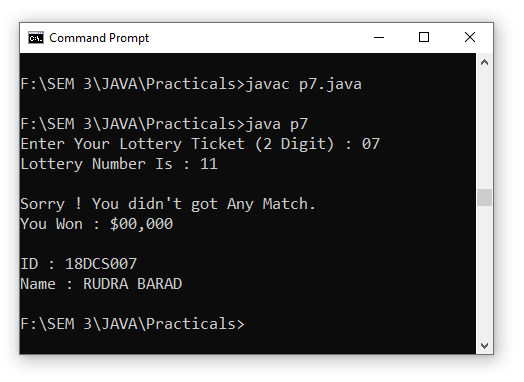
System.out.println("\nID : 18DCS007\nName : RUDRA BARAD");

}

}

**OUTPUT:**





**CONCLUSION:**

In this Practical I got to know about Random Class and How to generate a random 2 digit integer. I also used Scanner Class as well as If and Else If ladder for giving conditions, so that I can get Proper Output for Lottery.

**PRACTICAL - 8**

**AIM:**

The problem is to write a program that will grade multiple-choice tests. Assume there are eight students and ten questions, and the answers are stored in a two dimensional array. Each row records a student’s answers to the questions, as shown in the following array.

Students’ Answers to the Questions:

0 1 2 3 4 5 6 7 8 9

Student 0 A B A C C D E E A D

Student 1 D B A B C A E E A D

Student 2 E D D A C B E E A D

Student 3 C B A E D C E E A D

Student 4 A B D C C D E E A D

Student 5 B B E C C D E E A D

Student 6 B B A C C D E E A D

Student 7 E B E C C D E E A D

The key is stored in a one-dimensional array:

Key to the Questions:

0 1 2 3 4 5 6 7 8 9

Key D B D C C D A E A D

Your program grades the test and displays the result. It compares each student’s

answers with the key, counts the number of correct answers, and displays it.

**PROGRAM CODE:**

import java.util.\*;

class p8

{

public static void main(String args[])

{

char stu[][] = new char[8][10];

char ans[] = {'D','B','D','C','C','D','A','E','A','D'};

int i,j,count=0;

Scanner sc = new Scanner(System.in);

for(i=0;i<8;i++)

{

System.out.println("Enter The Marks Of Student "+i);

for(j=0;j<10;j++)

{

stu[i][j] = sc.next().charAt(0);

}

}

System.out.println(" ");

for(i=0;i<8;i++)

{

count = 0;

System.out.print("Correct Answers by Student "+i+" are : ");

for(j=0;j<10;j++)

{

if(stu[i][j]==ans[j])

count++;

}

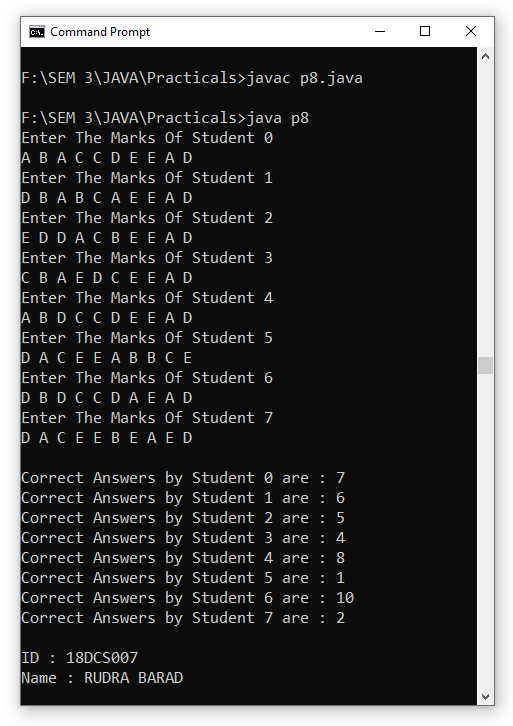
System.out.print(count+"\n");

}

System.out.println("\nID : 18DCS007\nName : RUDRA BARAD");

}

}

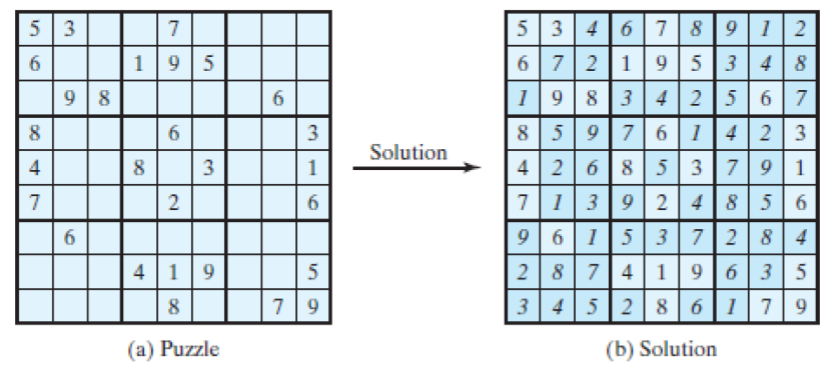
**OUTPUT:**

**CONCLUSION:**

In this Practical I learnt how to Use 2-Dimensional Array in Java. I also used next( ), and charAt( ) for inputting char values.

**PRACTICAL - 9**

**AIM:**

The problem is to check whether a given Sudoku solution is correct.

**PROGRAM CODE:**

import java.util.\*;

class p9

{

public static void main(String args[])

{

Scanner sc = new Scanner(System.in);

int s[][] = new int[9][9];

s[0][0]=5;

s[0][1]=3;

s[0][4]=7;

s[1][0]=6;

s[1][3]=1;

s[1][4]=9;

s[1][5]=5;

s[2][1]=9;

s[2][2]=8;

s[2][7]=6;

s[3][0]=8;

s[3][4]=6;

s[3][8]=3;

s[4][0]=4;

s[4][3]=8;

s[4][5]=3;

s[4][8]=1;

s[5][0]=7;

s[5][0]=7;

s[5][4]=2;

s[5][8]=6;

s[6][1]=6;

s[7][3]=4;

s[7][4]=1;

s[7][5]=9;

s[7][8]=5;

s[8][4]=8;

s[8][7]=7;

s[8][8]=9;

for(int i=0;i<9;i++)

{

System.out.println(" ");

for(int j=0;j<9;j++)

{

System.out.print(" "+s[i][j]);

}

System.out.println(" ");

}

System.out.println("Here's the Incomplete Sudoko !Complete The Sudoko \n");

for(int i=0;i<9;i++)

{

for(int j=0;j<9;j++)

{

s[i][j] = sc.nextInt();

}

}

int sum=0,flag=0;

for(int i=0;i<9;i++)

{

for(int j=0;j<9;j++)

{

sum+=s[i][j];

}

if(sum==45)

flag = 1;

else

flag = 0;

sum=0;

}

for(int i=0;i<9;i++)

{

for(int j=0;j<9;j++)

{

sum+=s[j][i];

}

if(sum==45)

flag = 1;

else

flag = 0;

sum=0;

}

if(flag==1)

System.out.println("\nCongratulations ! Your Sudoku is 100% CORRECT :) ");

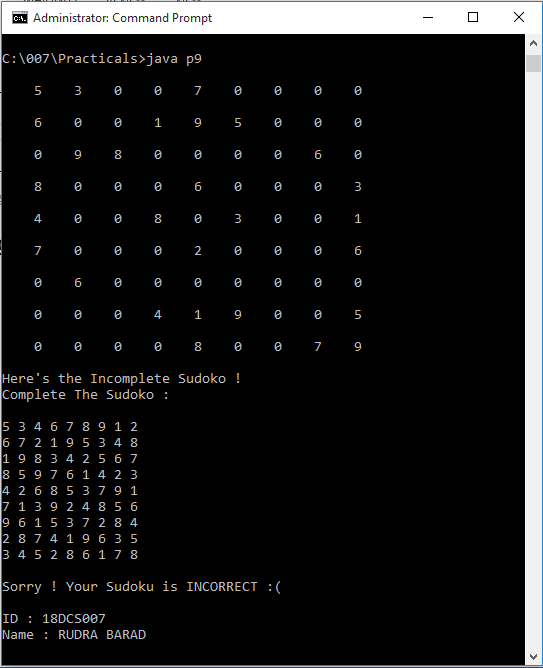
else

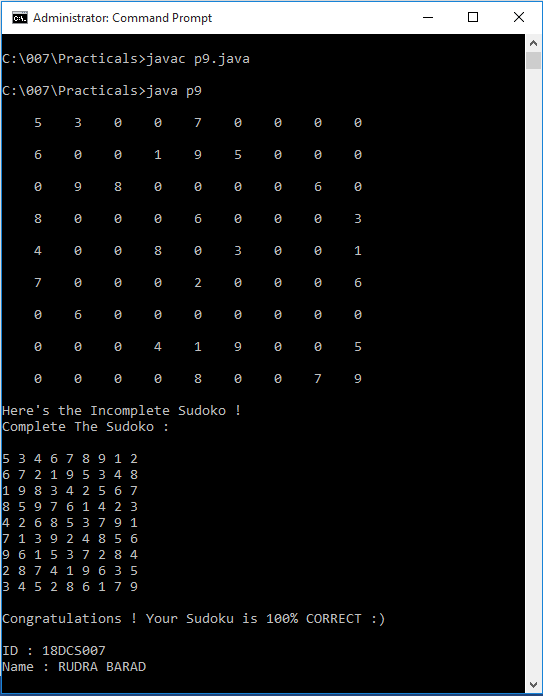
System.out.println("\nSorry ! Your Sudoku is INCORRECT :( ");

System.out.println("\nID : 18DCS007\nName : RUDRA BARAD");

}

}

**OUTPUT:**



**CONCLUSION:**

In this Practical I learnt How to check the solution of Sudoku and also used nextInt( ) for getting Int values from User. Condition that I used to check the solution of Sudoku is that Sum of Each Rows and Coulumn must be 45 as Each row contains unique 0-9 numbers, which gives sum of 45.

**PRACTICAL - 10**

**AIM:**

Finding a Closest Pair

The GPS navigation system is becoming increasingly popular. The system uses the graph and geometric algorithms to calculate distances and map a route. The practical presents a geometric problem for finding a closest pair of point.

**Output:**

Enter the number of points: 8

Enter 8 points: (-1, 3) (-1, -1) (1, 1) (2, 0.5) (2, -1) (3, 3) (4, 2) (4, -0.5)

The closest two points are (1.0, 1.0) and (2.0, 0.5)

**PROGRAM CODE:**

import java.util.\*;

class p10

{

static double calculateDistance(double x1, double x2, double y1, double y2)

{

return(Math.sqrt( Math.pow(y1-x1,2) + Math.pow(y2-x2,2) ) );

}

public static void main(String args[])

{

Scanner sc = new Scanner(System.in);

System.out.print("\nEnter The Amount Of Number You want to check : ");

int num = sc.nextInt();

double n1=0,n2=0,n3=0,n4=0;

double arr[][] = new double[num][2];

int i,j;

double dist,min;

System.out.println("\nEnter "+num+" set of Numbers \n");

for(i=0;i<num;i++)

{

System.out.print("SET "+(i+1)+" : ");

for(j=0;j<2;j++)

{

arr[i][j] = sc.nextDouble();

}

}

min = calculateDistance(arr[0][0],arr[0][1],arr[1][0],arr[1][1]);

System.out.println(" ");

for(i=0;i<(num-1);i++)

{

for(j=(i+1);j<(num);j++)

{

dist = calculateDistance(arr[i][0],arr[i][1],arr[j][0],arr[j][1]);

if ( dist < min )

{

min = dist;

n1 = arr[i][0];

n2 = arr[i][1];

n3 = arr[i+1][0];

n4 = arr[i+1][1];

}

}

}

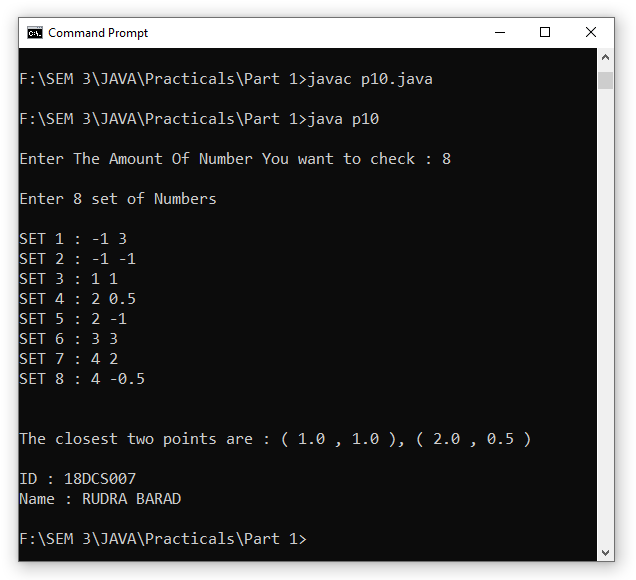
System.out.println("\nThe closest two points are : ( "+n1+" , "+n2+" ), ( "+n3+" , "+n4+" ) ");

System.out.println("\nID : 18DCS007\nName : RUDRA BARAD");

}

}

**OUTPUT:**



**CONCLUSION:**

In this Practical I learnt about how to apply algorithms to calculate distances in programming for finding the closest points. I also learnt how to implement Math functions like pow( ) and sqrt( ), etc.

**PART-II**

**Object Oriented Programming : Classes, Methods, Inheritance**

**PRACTICAL – 1**

**AIM:** Design a class named Circle containing following attributes and behavior.

* One double data field named radius. The default value is 1.
* A no-argument constructor that creates a default circle.
* A Single argument constructor that creates a Circle with the specified radius.
* A method named getArea() that returns area of the Circle.
* A method named getPerimeter() that returns perimeter of it.

**PROGRAM CODE:**

class circle

{

double radius = 1;

circle()

{

System.out.println("\nNo Argument Default Constructor called.");

}

circle(double r)

{

System.out.println("\nSingle Argument Constructor called.");

radius = r;

System.out.println("\nRadius of Circle is : "+radius);

}

double getArea()

{

return(3.14\*radius\*radius);

}

double getPerimeter()

{

return(2\*3.14\*radius);

}

public static void main(String args[])

{

circle c1 = new circle();

circle c2 = new circle(10.00);

double a = c2.getArea();

double p = c2.getPerimeter();

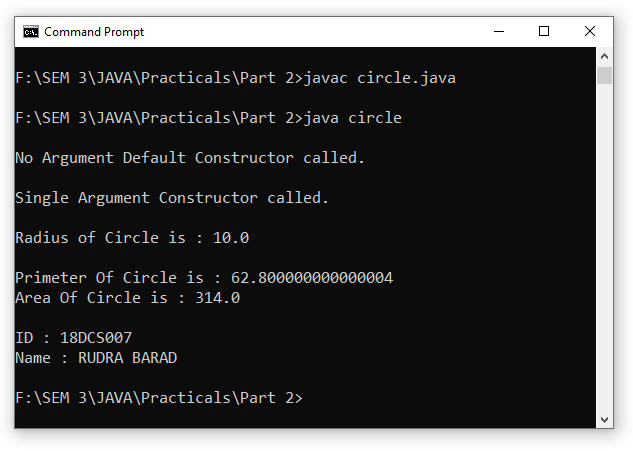
System.out.println("\nPrimeter Of Circle is : "+p);

System.out.println("Area Of Circle is : "+a);

System.out.println("\nID : 18DCS007\nName : RUDRA BARAD");

}

}

**OUTPUT:**

**CONCLUSION:**

In this practical I made class named Circle in which I initialized a double variable named radius, No- Argument Constructor and Single Argument Constructor which defines Radius of Circle. I also defined two methods namely getPerimeter( ) and getArea( ).

**PRACTICAL – 2**

**AIM:** Design a class named Account that contains:

* A private int data field named id for the account (default 0).
* A private double data field named balance for the account (default 500₹).
* A private double data field named annualInterestRate that stores the current interest rate. ( Default 7% ). Assume all accounts have the same interest rate.
* A private Date data field named dateCreated that stores the date when the account was created.
* A no-arg constructor that creates a default account.
* A constructor that creates an account with the specified id and initial balance.
* The accessor and mutator methods for id, balance, and annualInterestRate.
* The accessor method for dateCreated.
* A method named getMonthlyInterestRate() that returns the monthly interest rate.
* A method named getMonthlyInterest() that returns the monthly interest.
* A method named withdraw that withdraws a specified amount from the account.
* A method named deposit that deposits a specified amount to the account.

**PROGRAM CODE:**

import java.util.\*;

class account

{

private int id = 0;

private double balance = 500;

private double annualInterestRate = 7;

private Date dateCreated;

Date d1 = new Date();

Scanner sc = new Scanner(System.in);

double total,w,d;

account()

{

System.out.println("\nNo Argument Default Constructor called.");

id = 0;

balance = 500;

System.out.println("Initial Id : "+id+"\nInitial Balance is : "+balance);

}

void getValues()

{

System.out.print("\nEnter The ID for Account : ");

id = sc.nextInt();

System.out.print("Enter The Balance for Account : ");

balance = sc.nextDouble();

System.out.print("Enter The Annual Interest Rate for Account : ");

annualInterestRate = sc.nextDouble();

dateCreated = d1;

}

void setValues()

{

System.out.println("\nID for Account is : "+id);

System.out.println("Balance for Account is : Rs "+balance);

System.out.println("Annual Interest Rate for Account is : "+annualInterestRate+" %");

System.out.println("Date When Account was Created is : "+d1);

}

double getMonthlyInterestRate()

{

return(annualInterestRate/12);

}

double getMonthlyInterest()

{

total = (annualInterestRate\*balance)/100;

System.out.println("Annual Interest is : Rs "+total);

return(total/12);

}

void withdraw()

{

System.out.println("\nCurrent Balance for Account is : Rs "+balance);

System.out.print("Enter The Amount Of Balance You Want to Withdraw : ");

w = sc.nextDouble();

balance = balance - w;

System.out.println("\nAfter Withdrawing Balance is : Rs "+balance);

}

void deposit()

{

System.out.println("\nCurrent Balance for Account is : Rs "+balance);

System.out.print("Enter The Amount Of Balance You Want to Deposit : ");

d = sc.nextDouble();

balance = balance + d;

System.out.println("\nAfter Depositing Balance is : Rs "+balance);

}

public static void main(String args[])

{

account a1 = new account();

a1.getValues();

a1.setValues();

double month\_rate = a1.getMonthlyInterestRate();

System.out.println("\nTherefore, Monthly Interest Rate is : "+month\_rate+" %");

double month\_interest = a1.getMonthlyInterest();

System.out.println("\nSo, the Monthly Interest is : Rs "+month\_interest+" ");

a1.deposit();

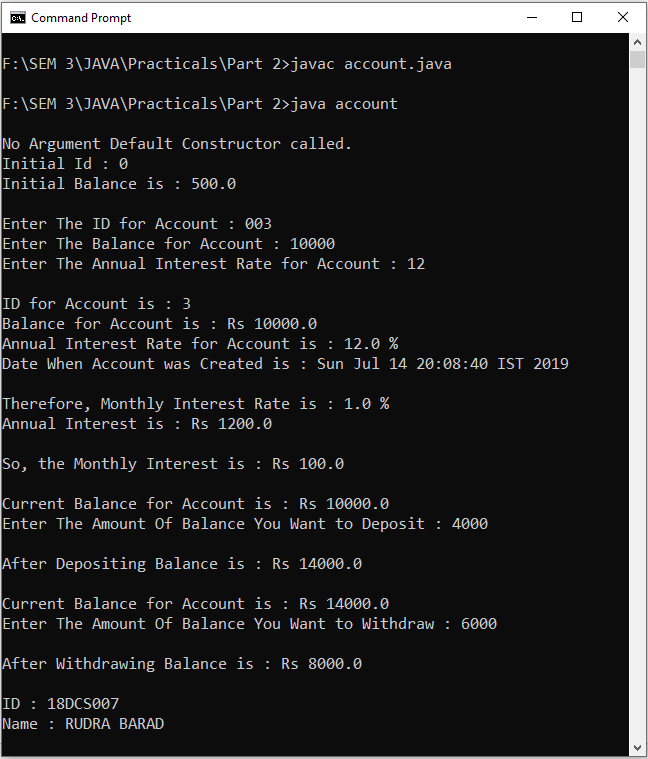
a1.withdraw();

System.out.println("\nID : 18DCS007\nName : RUDRA BARAD");

}

}

**OUTPUT:**



**CONCLUSION:**

In this Practical I created class named Account, Initialized private int variable as id, private variable double named as balance, private double variable named as annualInterestRate and private Date variable as dateCreated. I also built a Non-Argument Constructor and two methods for entering and displaying data. Also made methods that return monthly interest rate and monthly rate and also 2 methods which keeps record or money withdrawed and deposited. I learnt how to implement Date Class too.

**PRACTICAL – 3**

**AIM:**

Use the Account class created as above to simulate an ATM machine.

Create 10 accounts with id **AC001…..AC010** with initial balance 300₹. The system prompts the users to enter an id. If the id is entered incorrectly, ask the user to enter a correct id. Once an id is accepted, display menu with multiple choices.

1. Balance inquiry

2. Withdraw money [Maintain minimum balance 300₹]

3. Deposit money

4. Money Transfer

5. Create Account

6. Deactivate Account

7. Exit

**Hint**: Use **ArrayList**, which is can shrink and expand with compared to Array.

**PROGRAM CODE:**

import java.lang.\*;

import java.util.\*;

class Account

{

String id;

int balance;

Account(String i)

{

id = i;

balance = 300;

}

void BalanceInq()

{

System.out.println("ID :- " + id);

System.out.println("Balance :- " + balance);

}

void withdraw()

{

int temp;

Scanner sc = new Scanner(System.in);

System.out.print("Enter amount to withdraw :- ");

temp = sc.nextInt();

if(balance - temp < 300)

System.out.println("Minimum 300 balance is needed");

else

{

balance = balance - temp;

System.out.println("Current balance :- " + balance);

}

}

void deposit()

{

int temp;

Scanner sc = new Scanner(System.in);

System.out.print("Enter amount to deposite :- ");

temp = sc.nextInt();

balance = balance + temp;

System.out.println("Current balance :- " + balance);

}

void moneyTransfer(ArrayList ar)

{

int temp;

String id;

Scanner sc = new Scanner(System.in);

Account ac = null;

System.out.print("Enter ID to transfer :- ");

id = sc.next();

Iterator itr = ar.iterator();

while(itr.hasNext())

{

Account a = (Account) itr.next();

if(id.equals(a.id))

ac = a;

}

if(ac == null)

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

else

{

System.out.print("Enter amount to transfer :- ");

temp = sc.nextInt();

if(temp > balance)

System.out.println("Not enough balane to transfer");

else

{

ac.balance = ac.balance + temp;

balance = balance - temp;

}

}

}

void Create(ArrayList arr)

{

String str;

Scanner sc = new Scanner(System.in);

System.out.print("Enter ID :- ");

str = sc.next();

Account ac = new Account(str);

arr.add(ac);

}

void delete(ArrayList arr)

{

String str;

Scanner sc = new Scanner(System.in);

Account ac = null;

System.out.print("Enter ID :- ");

str = sc.next();

Iterator itr = arr.iterator();

while(itr.hasNext())

{

Account a = (Account) itr.next();

if(str.equals(a.id))

ac = a;

}

if(ac == null)

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

else

{

arr.remove(ac);

System.out.println("Account removed successfully");

}

}

}

public class java13

{

public static void main(String[] args)

{

ArrayList<Account> arr = new ArrayList<Account>();

Scanner sc = new Scanner(System.in);

String str;

int op;

Account ac = null;

Account a = new Account(null);

arr.add(new Account("AC001"));

arr.add(new Account("AC002"));

arr.add(new Account("AC003"));

arr.add(new Account("AC004"));

arr.add(new Account("AC005"));

arr.add(new Account("AC006"));

arr.add(new Account("AC007"));

arr.add(new Account("AC008"));

arr.add(new Account("AC009"));

arr.add(new Account("AC010"));

System.out.print("Enter your ID :- ");

str = sc.next();

Iterator itr = arr.iterator();

while(itr.hasNext())

{

a = (Account) itr.next();

if(str.equals(a.id))

ac = a;

}

if(ac == null)

{

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

System.exit(0);

}

else

{

do

{

System.out.println("Press 1 - for Balance Inquiry");

System.out.println("Press 2 - for Withdraw");

System.out.println("Press 3 - for Deposit");

System.out.println("Press 4 - for Money Transfer");

System.out.println("Press 5 - for Create Account");

System.out.println("Press 6 - for Remove Account");

System.out.println("Press 0 - for Exit");

System.out.print("Enter your choice => ");

op = sc.nextInt();

switch(op)

{

case 1:

ac.BalanceInq();

break;

case 2:

ac.withdraw();

break;

case 3:

ac.deposit();

break;

case 4:

ac.moneyTransfer(arr);

break;

case 5:

ac.Create(arr);

break;

case 6:

ac.delete(arr);

break;

case 0:

System.out.println("Thank you");

break;

default:

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

}

}while(op != 0);

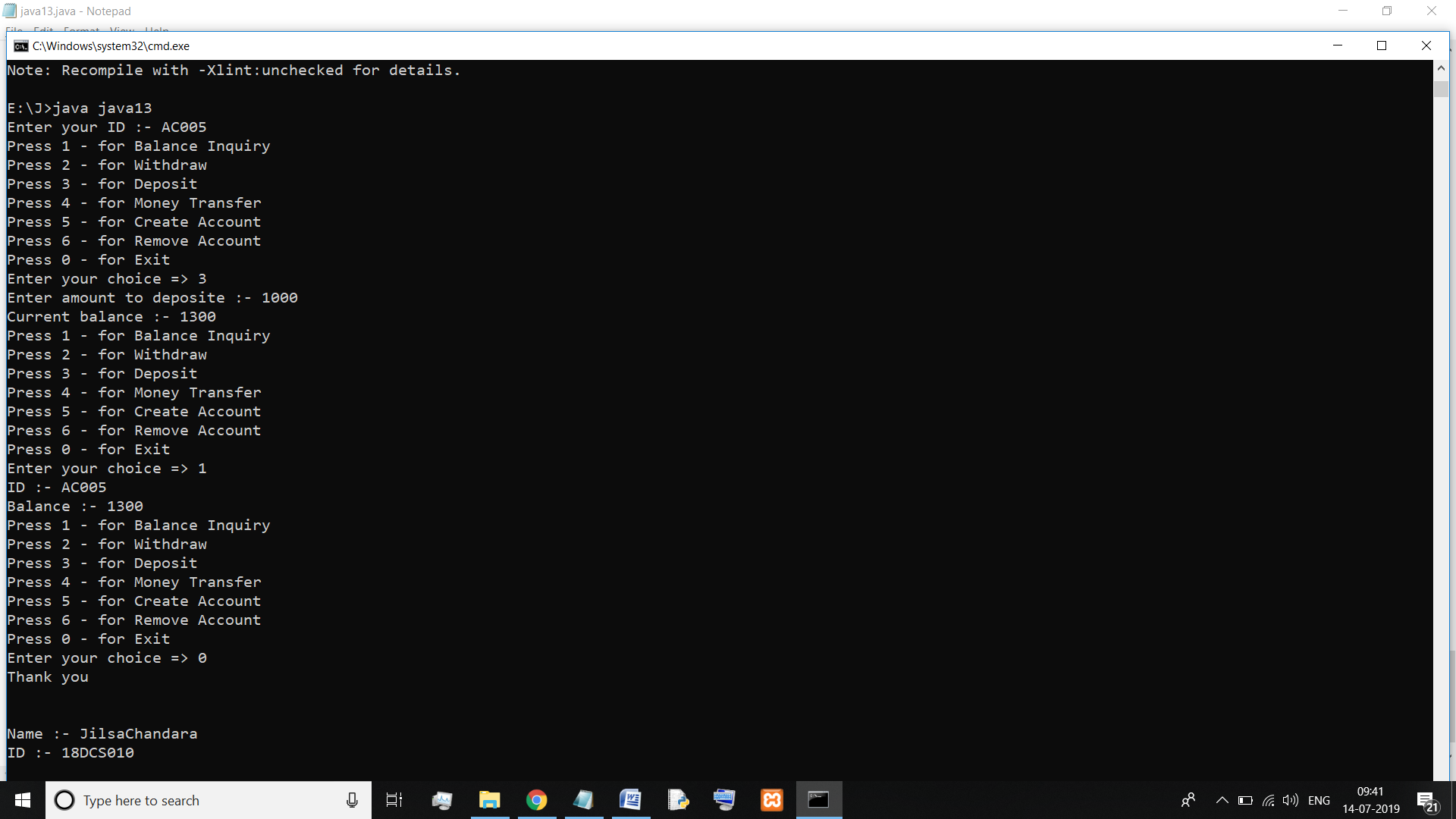
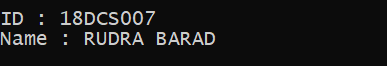
}

System.out.println("\n\ID :- 18DCS007\n nName :- RUDRA BARAD ");

}

}

**OUTPUT:**



**CONCLUSION:**

In this practical, we learned about menu driven program and ArrayList

**PRACTICAL – 4**

**AIM:** Write a java program that implements educational hierarchy using inheritance.

**PROGRAM CODE:**

import java.util.\*;

class office

{

int empNo;

String empName;

double salary;

Scanner sc = new Scanner(System.in);

void getValue()

{

System.out.print("\nEnter the Employee Name : ");

empName = sc.nextLine();

System.out.print("Enter the Employee Number : ");

empNo = sc.nextInt();

System.out.print("Enter the Salary of Employee :");

salary = sc.nextDouble();

}

}

class teaching extends office

{

void setValue()

{

String designation = "TEACHING";

System.out.println("\nDesignation is : "+designation);

System.out.println("\nEmployee Number for Teaching is : "+empNo);

System.out.println("Employee Name for Teaching is : "+empName);

System.out.println("Salary for Teaching is :"+salary+"\n");

}

}

class non\_teaching extends office

{

void setValue()

{

String designation = "NON - TEACHING";

System.out.println("\nDesignation is : "+designation);

System.out.println("\nEmployee Number for Non-Teaching is : "+empNo);

System.out.println("Employee Name for Non-Teaching is : "+empName);

System.out.println("Salary for Non-Teaching is :"+salary);

}

public static void main(String args[])

{

teaching t = new teaching();

t.getValue();

t.setValue();

non\_teaching nt = new non\_teaching();

nt.getValue();

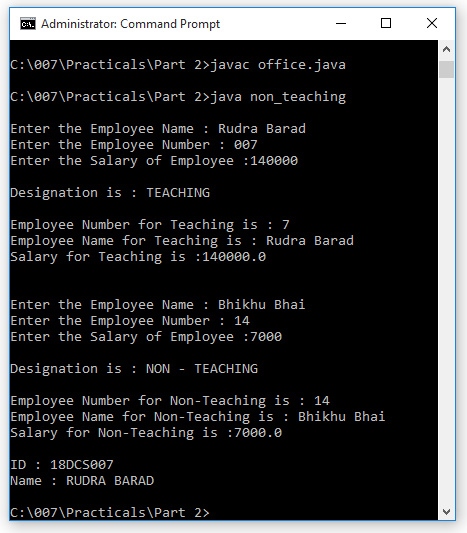
nt.setValue();

System.out.println("\nID : 18DCS007\nName : RUDRA BARAD");

}

}

**OUTPUT:**



**CONCLUSION:**

In this practical, we learned about inheritance and used Hierarchical inheritance here

**PRACTICAL – 5**

**AIM:** Develop a Program that illustrate method overloading concept.

**PROGRAM CODE:**

class overloading

{

void add(int a, int b)

{

int sum = a + b;

System.out.println("\nAddition of Two int Numbers "+a+" and "+b+" : "+sum);

}

void add(double a, double b)

{

double sum = a + b;

System.out.println("Addition of Two double Numbers "+a+" and "+b+" : "+sum);

}

void add(int a, double b)

{

double sum = a + b;

System.out.println("Addition of One int and One Double Number "+a+" and "+b+" : "+sum);

}

void add(char a, char b)

{

int sum = a + b;

System.out.println("Addition of Two chars "+a+" and "+b+" : "+sum);

}

void add(char a, int b)

{

int sum = a + b;

System.out.println("Addition of One char and One int "+a+" and "+b+" : "+sum);

}

public static void main(String args[])

{

System.out.println("\nMETHOD OVERLOADING EXAMPLE");

overloading o = new overloading();

o.add(3,6);

o.add(22.4,27.6);

o.add(2,5.0);

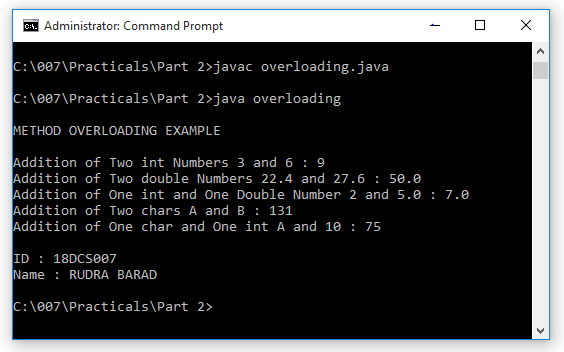
o.add('A','B');

o.add('A',10);

System.out.println("\nID : 18DCS007\nName : RUDRA BARAD");

} }

**OUTPUT:**



**CONCLUSION:**

In this practical, we learned about method overloading.

**PART-III**

**Package & Interface**

**PRACTICAL – 1**

**AIM:**

WAP that illustrate the use of interface reference. Interface Luminious Object has two method lightOn() and lightOff(). There is one class Solid extended by 2 classes Cube and Cone. There is one class LuminiousCone extends Cone and implements Luminoius Interface. LumminuiousCube extends Cube and implements Luminious Interface. Create a object of LuminiousCone and LuminousCube and use the concept of interface reference to invoke the methods of interface.

**PROGRAM CODE:**

interface Luminious

{

void LightOn();

void LightOff();

}

class Solid

{}

class Cone extends Solid

{}

class Cube extends Solid

{}

class LuminiousCone extends Cone implements Luminious

{

public void LightOn()

{

System.out.println("Light on from Cone");

}

public void LightOff()

{

System.out.println("Light off from Cone");

}

}

class LuminiousCube extends Cube implements Luminious

{

public void LightOn()

{

System.out.println("Light on from Cube");

}

public void LightOff()

{

System.out.println("Light off from Cube");

}

}

public class java16

{

public static void main(String[] agrs)

{

Luminious LCube = new LuminiousCube();

LCube.LightOn();

LCube.LightOff();

Luminious LCone = new LuminiousCone();

LCone.LightOn();

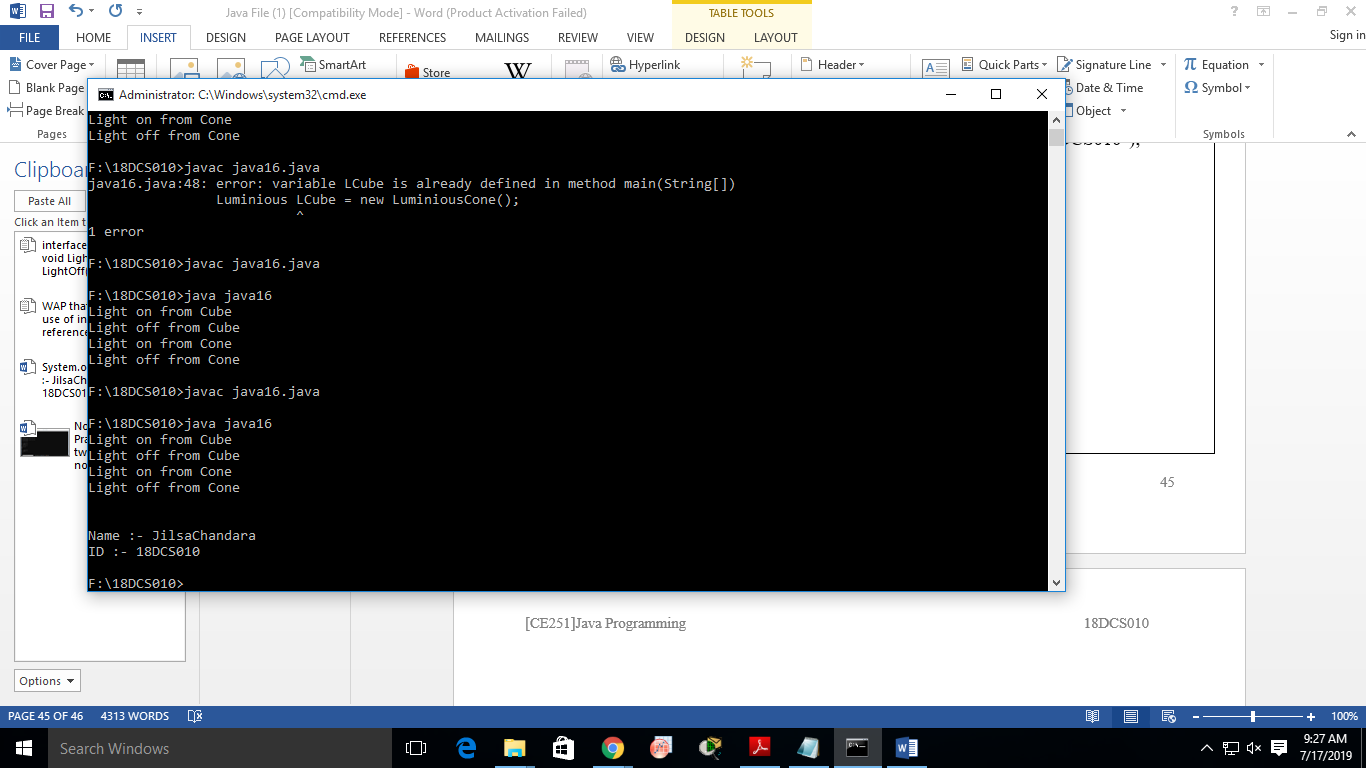
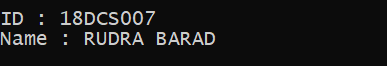
LCone.LightOff();

System.out.println("\nID : 18DCS007\nName : RUDRA BARAD");

}

}

**OUTPUT:**



**CONCLUSION:**

In this practical, we learned about interfaces and abstract methods.

**PRACTICAL – 2**

**AIM:**

WAP that illustrate the interface inheritance.

Interface P is extended by P1 and P2 interfaces.

Interface P12 extends both P1 and P2. Each interface declares one method and one constant. Create one class that implemetns P12. By using the object of the class invokes each of its method and displays constant.

**PROGRAM CODE:**

interface P

{

final int p = 1;

void p\_method();

}

interface P1 extends P

{

final int p1 = 2;

void p1\_method();

}

interface P2 extends P

{

final int p2 = 3;

void p2\_method();

}

interface P12 extends P1, P2

{

final int p12 = 4;

void p12\_method();

}

class Class implements P12

{

public void p\_method()

{

System.out.println("Hello from P");

System.out.println("P => " + p);

}

public void p1\_method()

{

System.out.println("Hello from P1");

System.out.println("P1 => " + p1);

}

public void p2\_method()

{

System.out.println("Hello from P2");

System.out.println("P2 => " + p2);

}

public void p12\_method()

{

System.out.println("Hello from P12");

System.out.println("P12 => " + p12);

}

}

public class java17

{

public static void main(String[] args)

{

P12 pr = new Class();

pr.p\_method();

pr.p1\_method();

pr.p2\_method();

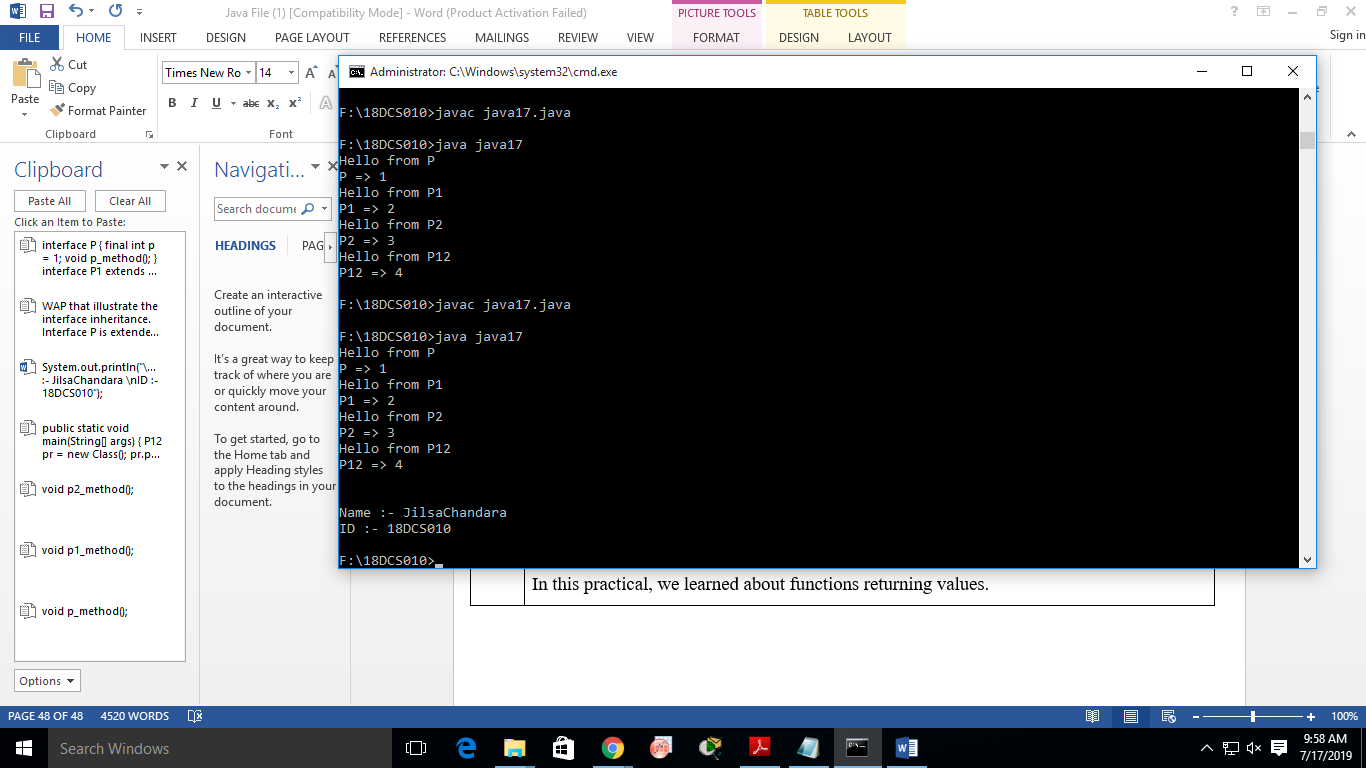
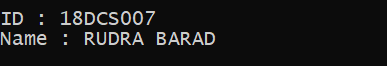
pr.p12\_method();

System.out.println("\nID : 18DCS007\nName : RUDRA BARAD");

}

}

**OUTPUT:**



**CONCLUSION:**

In this practical, we learned about inheritance of interfaces.

**PRACTICAL – 3**

**AIM:**

Create an abstract class Robot that has the concretre subclasses , RobotA, RobotB, RobotC. Class RobotA1 extends RobotA, RobotB1 extends RobotB and RobotC1 extends RobotC. There is interface Motion that declares 3 methods forward(), reverse() and stop(), implemented by RobotB and RobotC. Sound interface declare method beep() implemented by RobotA1, RobotB1 and RobotC1. Create an instance method of each class and invoke beep() and stop() method by all objects.

**PROGRAM CODE:**

abstract class Robot

{}

interface Motion

{

void forward();

void reverse();

void stop();

}

interface Sound

{

void Beep();

}

class RobotA extends Robot

{}

class RobotB extends Robot

{}

class RobotC extends Robot

{}

class RobotA1 extends RobotA implements Sound

{

public void stop()

{

System.out.println("Stop in RobotA1");

}

public void Beep()

{

System.out.println("Beep in RobotA1");

}

}

class RobotB1 extends RobotB implements Sound, Motion

{

public void forward()

{}

public void reverse()

{}

public void stop()

{

System.out.println("Stop in RobotB1");

}

public void Beep()

{

System.out.println("Beep in RobotB1");

}

}

class RobotC1 extends RobotC implements Sound, Motion

{

public void forward()

{}

public void reverse()

{}

public void stop()

{

System.out.println("Stop in RobotC1");

}

public void Beep()

{

System.out.println("Beep in RobotC1");

}

}

public class java18

{

public static void main(String[] args)

{

RobotA1 ra = new RobotA1();

RobotB1 rb = new RobotB1();

RobotC1 rc = new RobotC1();

ra.Beep();

ra.stop();

rb.Beep();

rb.stop();

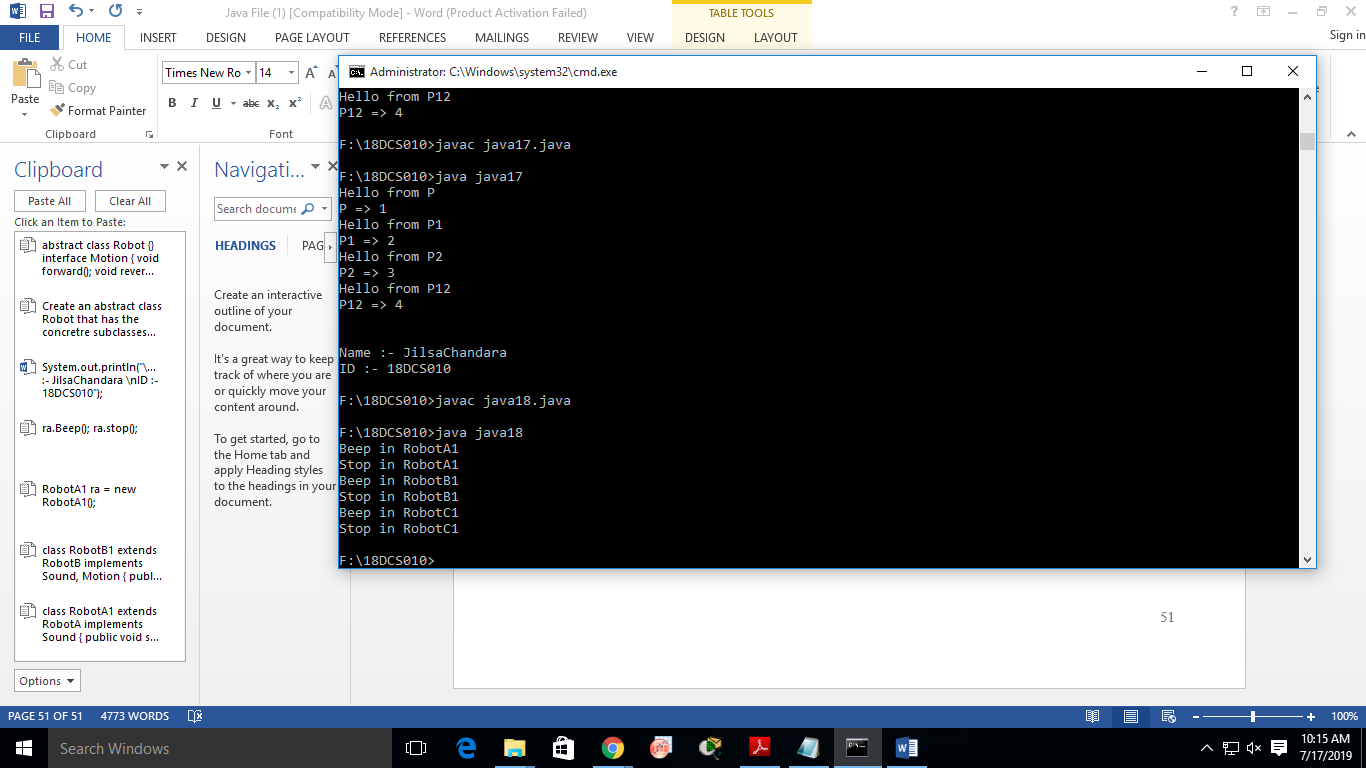
rc.Beep();

rc.stop();

}

}

**OUTPUT:**



**CONCLUSION:**

In this practical, we learned about implementing interface in inheritance of classes.

**PRACTICAL – 4**

**AIM:**

Write a java program to create an abstract class named Shape that contains two integers and an empty method named printArea(). Provide three classes named Rectangle, Triangle and Circle such that each one of the classes extends the class Shape. Each one of the classes contain only the method printArea( ) that prints the area of the given shape.

**PROGRAM CODE:**

abstract class Shape

{

int a, b;

abstract void printArea();

}

class Rectangle extends Shape

{

public Rectangle(int x, int y)

{

a = x;

b = y;

}

public void printArea()

{

System.out.println("Rectangle Area => " + (a \* b));

}

}

class Triangle extends Shape

{

public Triangle(int x, int y)

{

a = x;

b = y;

}

public void printArea()

{

System.out.println("Triangle Area => " + (0.5 \* a \* b));

}

}

class Circle extends Shape

{

public Circle(int x)

{

a = x;

}

public void printArea()

{

System.out.println("Circle Area => " + (3.14 \* a \* a));

}

}

public class java19

{

public static void main(String[] args)

{

Rectangle R = new Rectangle(10,20);

Triangle T = new Triangle(5,10);

Circle C = new Circle(8);

R.printArea();

T.printArea();

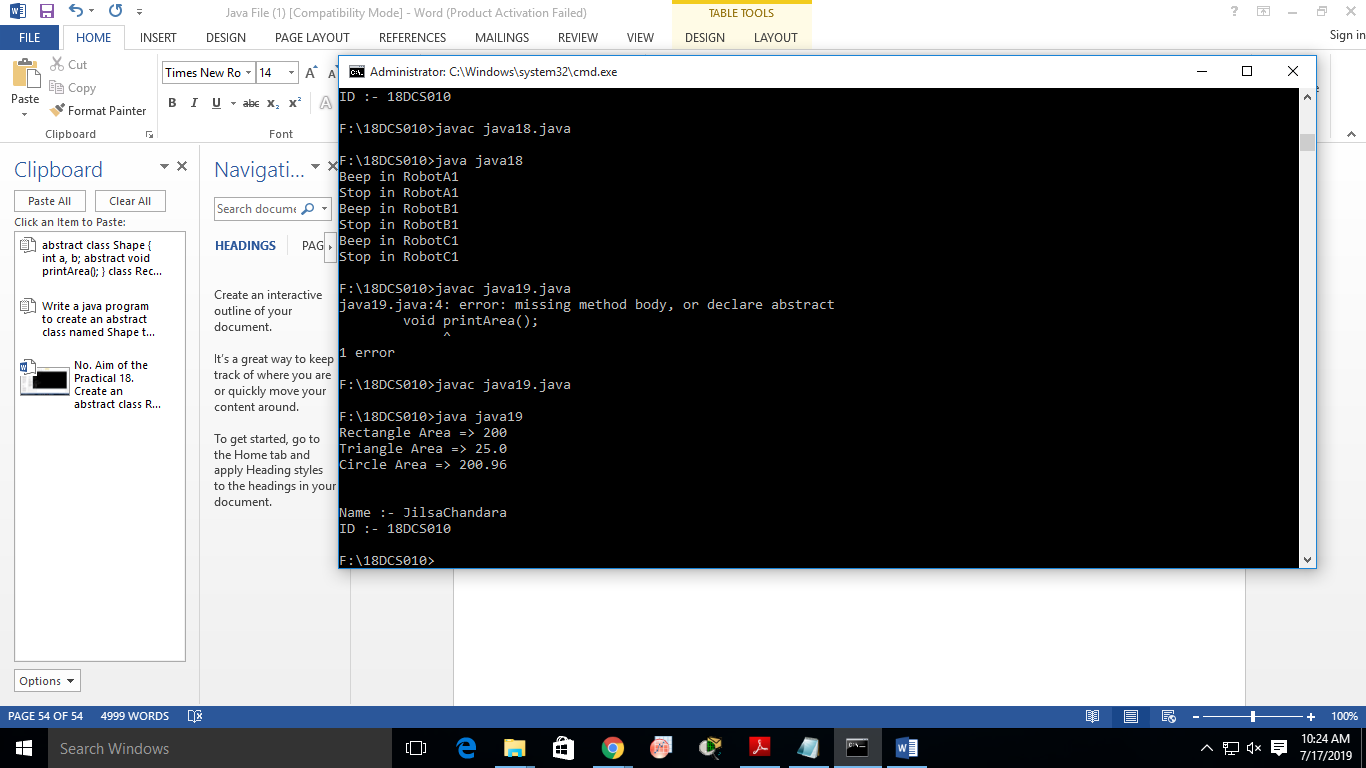
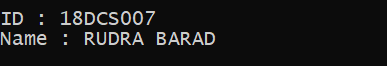
C.printArea();

System.out.println("\nID : 18DCS007\nName : RUDRA BARAD");

}

}

**OUTPUT:**



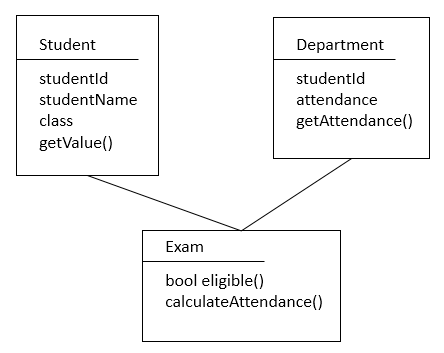
**CONCLUSION:**

In this practical, we learned about abstract class and override their methods.

**PRACTICAL – 5**

**AIM:**

Write a java program to find the details of the students eligible to enroll for the examination ( Students, Department combinedly give the eligibility criteria for the enrollement class) using interfaces.



**PROGRAM CODE:**

import java.util.Scanner;

interface Student

{

int studentID = 3;

String studentName = "ABC";

String Class = "CSE" ;

void getValue();

}

class Department implements Student

{

int attendance;

public void getAttendance()

{

Scanner sc = new Scanner(System.in);

System.out.print("Enter Attendance => ");

attendance = sc.nextInt();

}

public void getValue()

{

System.out.println("ID => " + studentID + "\tName => " + studentName + "\tClass => " + Class);

}

}

class Exam extends Department

{

public void calculateAttendance()

{

attendance = attendance;

}

public boolean eligible()

{

if(attendance < 60)

return false;

else

return true;

}

}

public class java20

{

public static void main(String[] args)

{

Exam e = new Exam();

e.getValue();

e.getAttendance();

e.calculateAttendance();

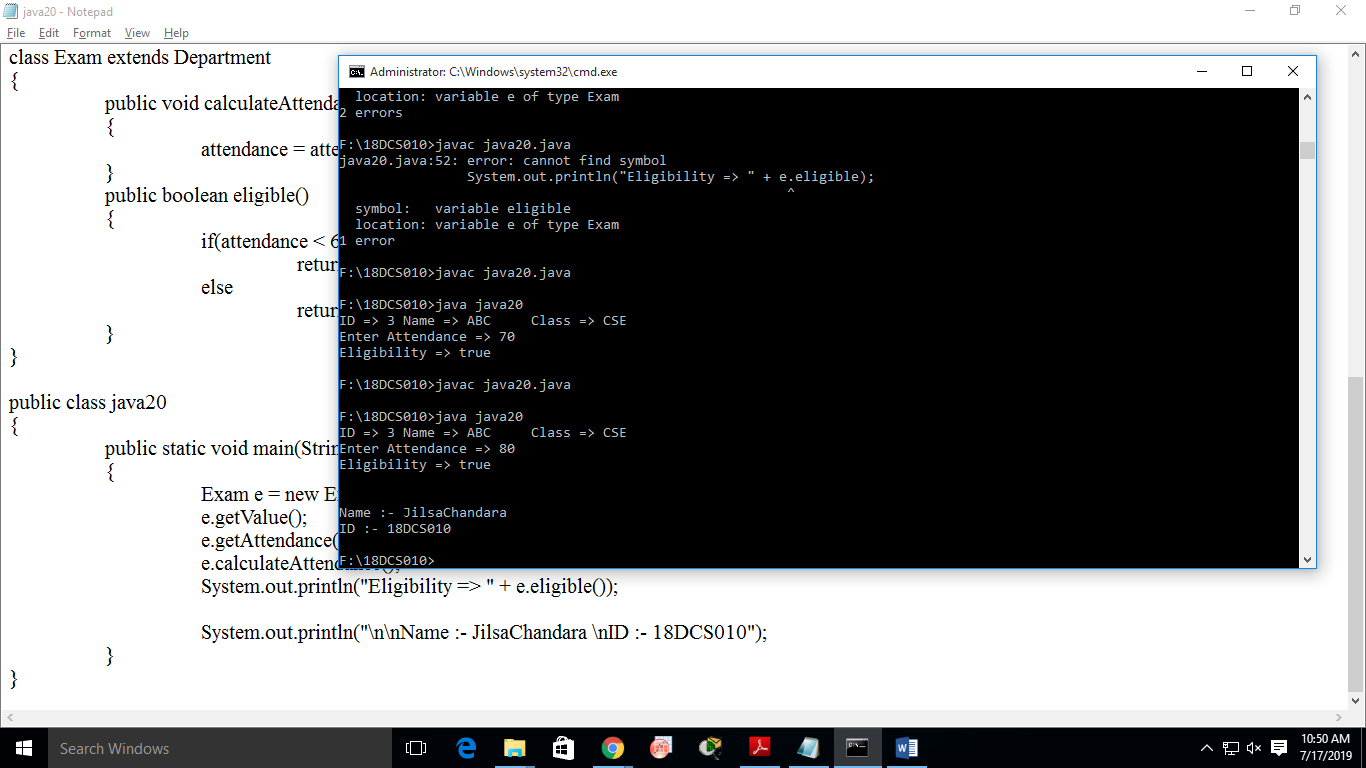
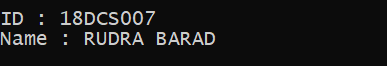
System.out.println("Eligibility => " + e.eligible());

System.out.println("\nID : 18DCS007\nName : RUDRA BARAD");

}

}

**OUTPUT:**



**CONCLUSION:**

In this practical, we learned about implementing interface and extending class in the same class.

**PRACTICAL – 6**

**AIM:**

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

**PROGRAM CODE:**

package p;

public class Maths

{

public void add(int a, int b)

{

System.out.println("Addition => " + (a + b));

}

}

import p.Maths;

public class java21

{

public static void main(String[] args)

{

Maths m = new Maths();

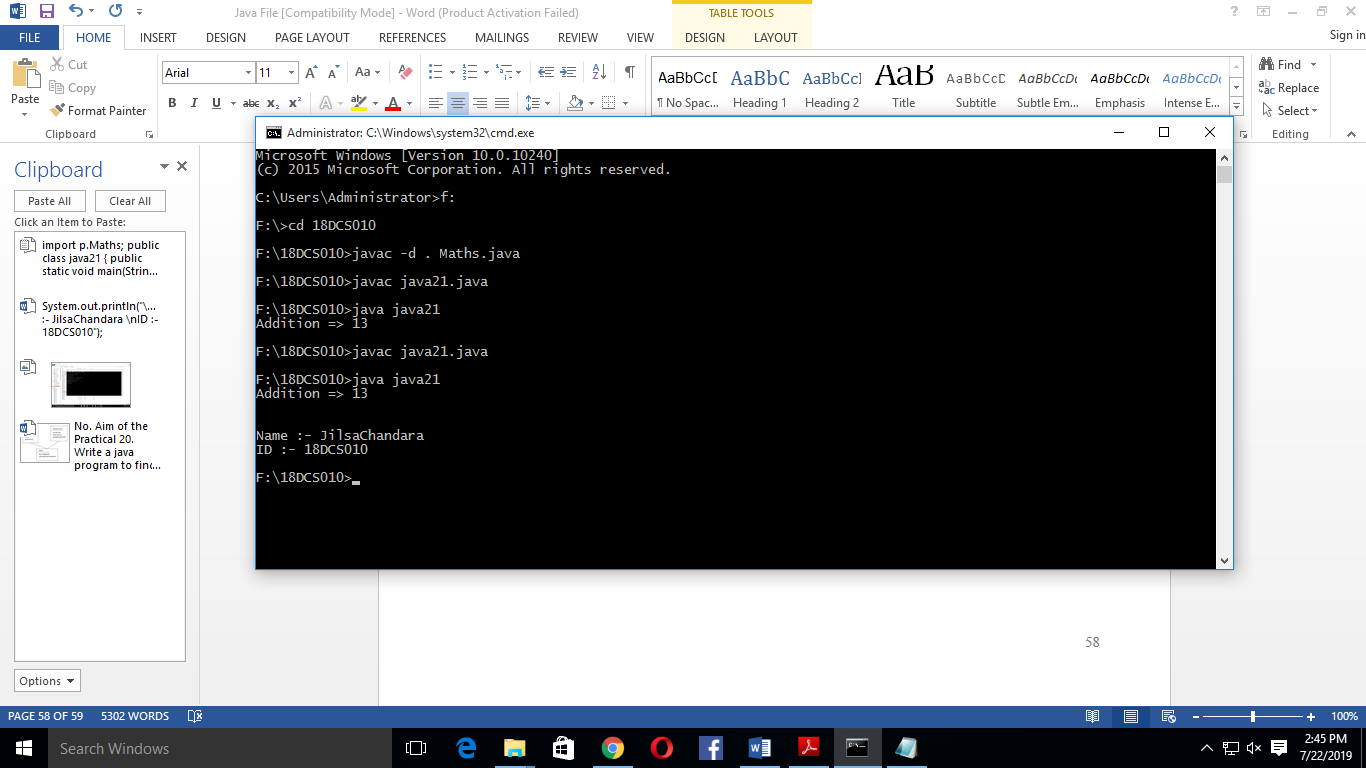
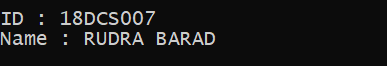
m.add(5,8);

System.out.println("\nID : 18DCS007\nName : RUDRA BARAD");

}

}

**OUTPUT:**



**CONCLUSION:**

In this practical, we learned about creating user defined package and importing them.

**PRACTICAL – 7**

**AIM:**

Write a program that demonstrates use of packages & import statements.

**PROGRAM CODE:**

package p;

public class Maths

{

public void add(int a, int b)

{

System.out.println("Addition => " + (a + b));

}

}

import p.Maths;

public class java21

{

public static void main(String[] args)

{

Maths m = new Maths();

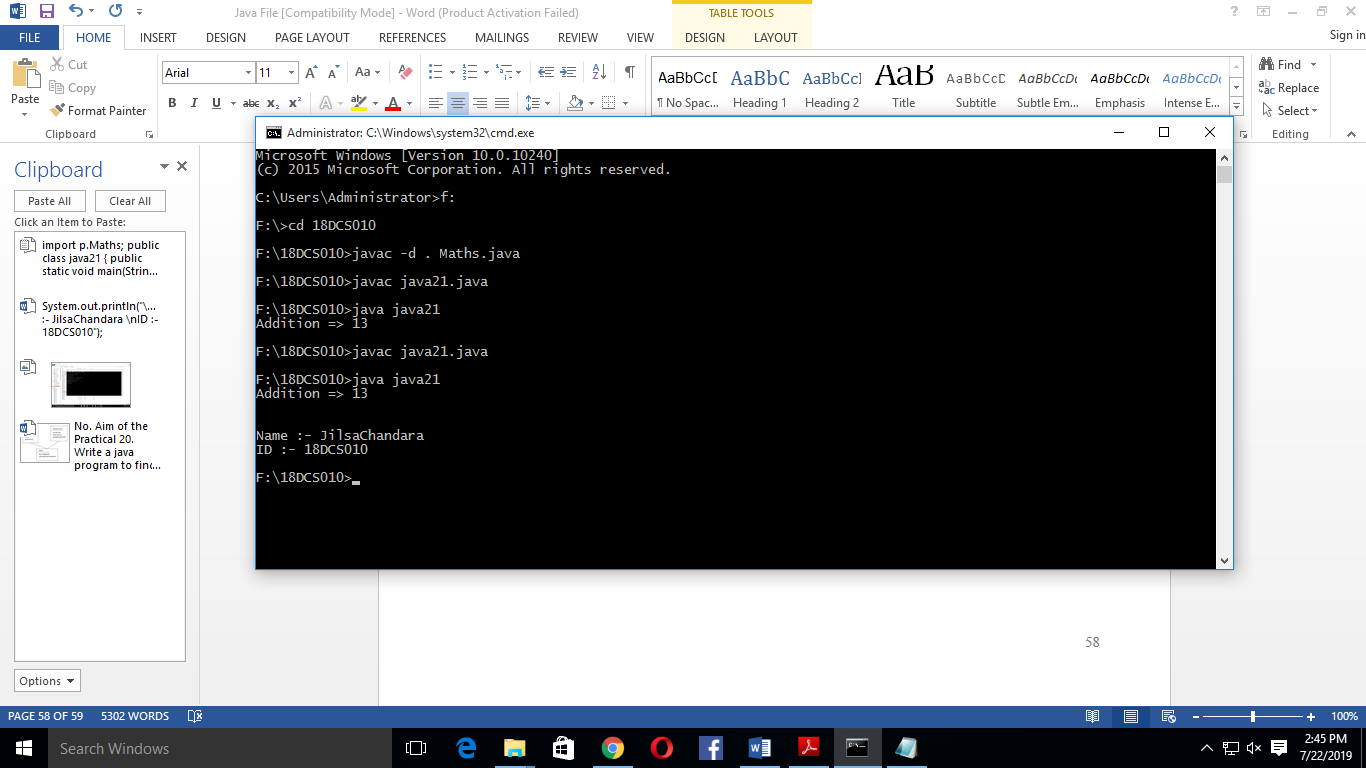
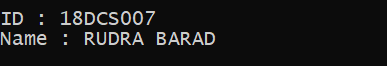
m.add(5,8);

System.out.println("\nID : 18DCS007\nName : RUDRA BARAD");

}

}

**OUTPUT:**



**CONCLUSION:**

In this practical, we learned about using the class of another file to main file. We used the concept of package and imported the class.

**PRACTICAL – 8**

**AIM:**

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

**PROGRAM CODE:**

interface I

{

default void show()

{

System.out.println("Show from Interface I");

}

}

interface i1 extends I

{ }

interface i2 extends I

{ }

public class java23 implements i1, i2

{

public static void main(String[] args)

{

java23 obj = new java23();

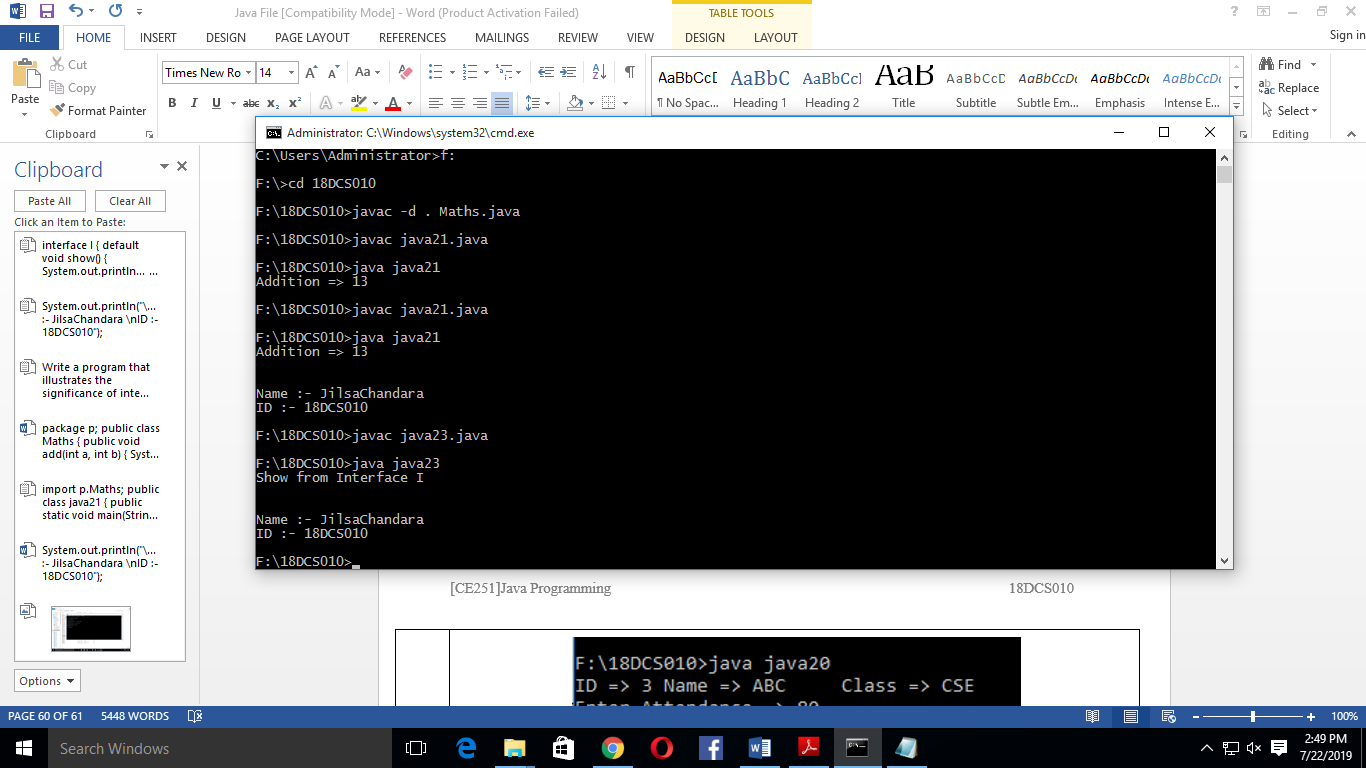
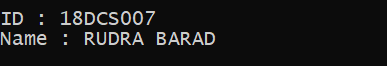
obj.show();

System.out.println("\nID : 18DCS007\nName : RUDRA BARAD");

}

}

**OUTPUT:**



**CONCLUSION:**

In this practical, we learned about default methods in interface and how it works.

**PART-IV**

**Exception Handling**

**PRACTICAL – 1**

**AIM:**

WAP to show the try - catch block to catch the different types of exception.

**PROGRAM CODE:**

class p1

{

public static void main(String args[])

{

try

{

int i = 10;

System.out.println(i/0);

}

catch(Exception e)

{

System.out.println("\n"+e);

}

try

{

int arr[] = new int[4];

arr[7]=7;

System.out.println(arr[7]);

}

catch(Exception e)

{

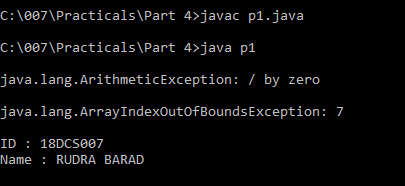
System.out.println("\n"+e);

}

System.out.println("\nID : 18DCS007\nName : RUDRA BARAD");

}

}

**OUTPUT:**

**CONCLUSION:**

In this practical, we learned about exception handling and try-catch blocks.

**PRACTICAL – 2**

**AIM:**

WAP to generate user defined exception using “throw” and “throws” keyword.

**PROGRAM CODE:**

class my\_exception extends Exception

{

String sop;

public my\_exception(String str)

{

sop = str;

}

public String toString()

{

return ("\nType Of Exception : "+sop);

}

}

class p2

{

public static void disp()throws NullPointerException, ArrayIndexOutOfBoundsException

{

String s = null;

System.out.println(s.length());

}

public static void main(String args[])

{

try

{

disp();

}

catch(Exception e)

{

System.out.println("\n"+e);

}

try

{

throw new my\_exception("USER DEFINED EXCEPTION");

}

catch(my\_exception e)

{

System.out.println(e);

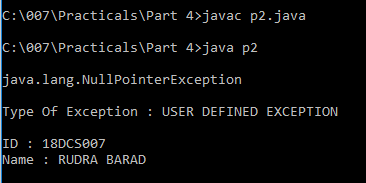
}

System.out.println("\nID : 18DCS007\nName : RUDRA BARAD");

}

}

**OUTPUT:**



**CONCLUSION:**

In this practical, we learned about throw and throws statements and how to use them.

**PRACTICAL – 3**

**AIM:**

Write a program that raises two exceptions. Specify two ‘catch’ clauses for the two exceptions. Each ‘catch’ block handles a different type of exception. For example the exception could be ‘ArithmeticException’ and ‘ArrayIndexOutOfBoundsException’. Display a message in the ‘finally’ block.

**PROGRAM CODE:**

class p3

{

public static void main(String args[])

{

try

{

int i = 10;

System.out.println(i/0);

}

catch(ArithmeticException e)

{

System.out.println("\n"+e);

}

try

{

int arr[] = new int[4];

arr[7]=7;

System.out.println(arr[7]);

}

catch(ArrayIndexOutOfBoundsException e)

{

System.out.println("\n"+e);

}

finally

{

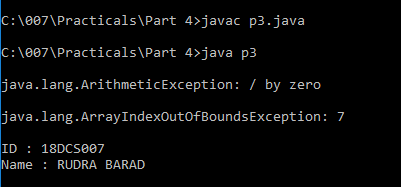
System.out.println("\nID : 18DCS007\nName : RUDRA BARAD");

}

}

}

**OUTPUT:**



**CONCLUSION:**

In this practical, we learned about multiple try-catch block to catch different types of exceptions.

**PART-V**

**File Handling & Streams**

**PRACTICAL – 1**

**AIM:**

WAP to show how to create a file with different mode and methods of File class

to find path, directory etc.

**PROGRAM CODE:**

import java.io.File;

class p1

{

public static void main(String args[])

{

try

{

System.out.println("\nUSING CREATE NEW FILE :");

File obj1 = new File("File 1.text");

if(obj1.createNewFile())

{

System.out.println("\nFile Is Created : " + obj1.getName());

}

else

{

System.out.println("\nFile 1 Already Exist !");

}

System.out.println("\nAbsolute Path OF File 1 is : "+obj1.getAbsolutePath());

obj1=obj1.getAbsoluteFile().getParentFile();

System.out.println("Parent Directory Of File 1 is : "+obj1);

System.out.println("Length OF File 1 is : "+obj1.length());

}

catch(Exception e)

{

System.out.println("Exception is : " + e);

}

try

{

System.out.println("\nUSING EXIST :");

File obj2 = new File("File 2.text");

obj2.createNewFile();

if(obj2.exists())

{

System.out.println("\nFile 2 Already Exist !");

}

else

{

System.out.println("\nFile Is Created : " + obj2.getName());

}

}

catch(Exception e)

{

System.out.println("Exception is : " + e);

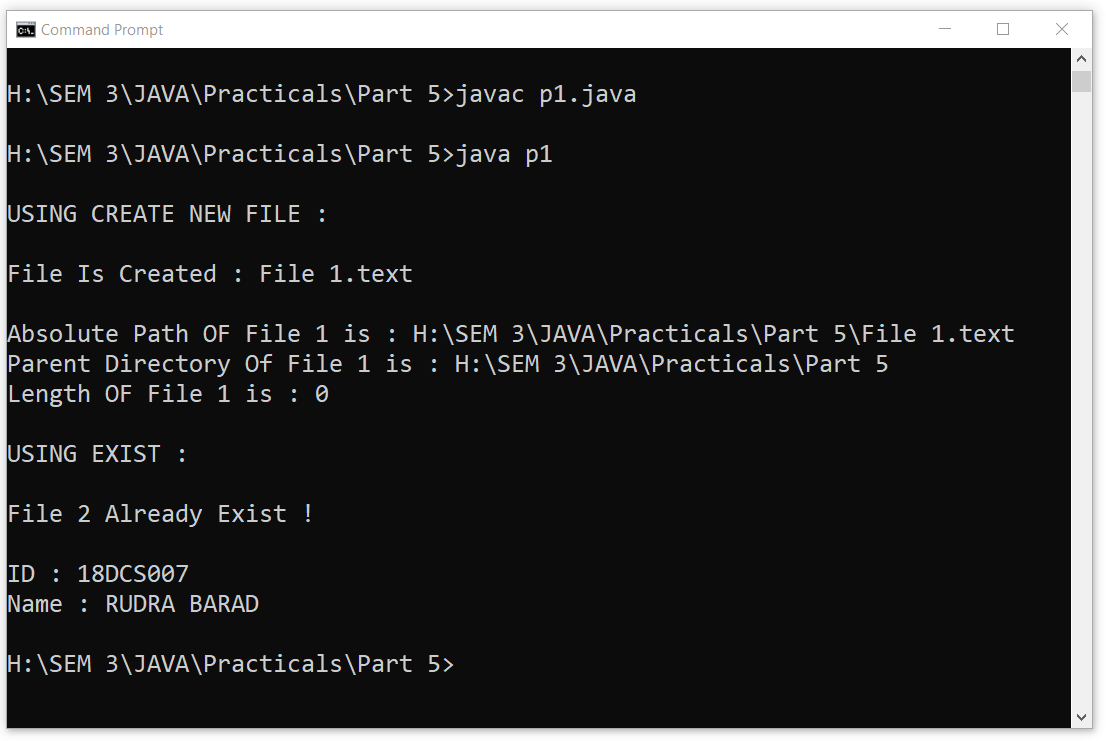
}

System.out.println("\nID : 18DCS007\nName : RUDRA BARAD");

}

}

**OUTPUT:**



**CONCLUSION:**

In this practical we learned how to create file in any mode and print its path and directory.

**PRACTICAL – 2**

**AIM:**

Write a program to show a tree view of files and directories under a specified

drive/volume.

**PROGRAM CODE:**

import java.io.File;

class p2

{

public static void displayIt(File f )

{

System.out.println(f.getAbsoluteFile());

if(f.isDirectory())

{

String[] subNote=f.list();

for(String filename : subNote)

{

displayIt(new File (f,filename));

}

}

}

public static void main(String args[])

{

try

{

System.out.println("\n TREE VIEW OF FILES & DIRECTORIES UNDER SPECIFIC DRIVE : ");

displayIt(new File("C: "));

}

catch(Exception e)

{

System.out.println("Exception is : " + e);

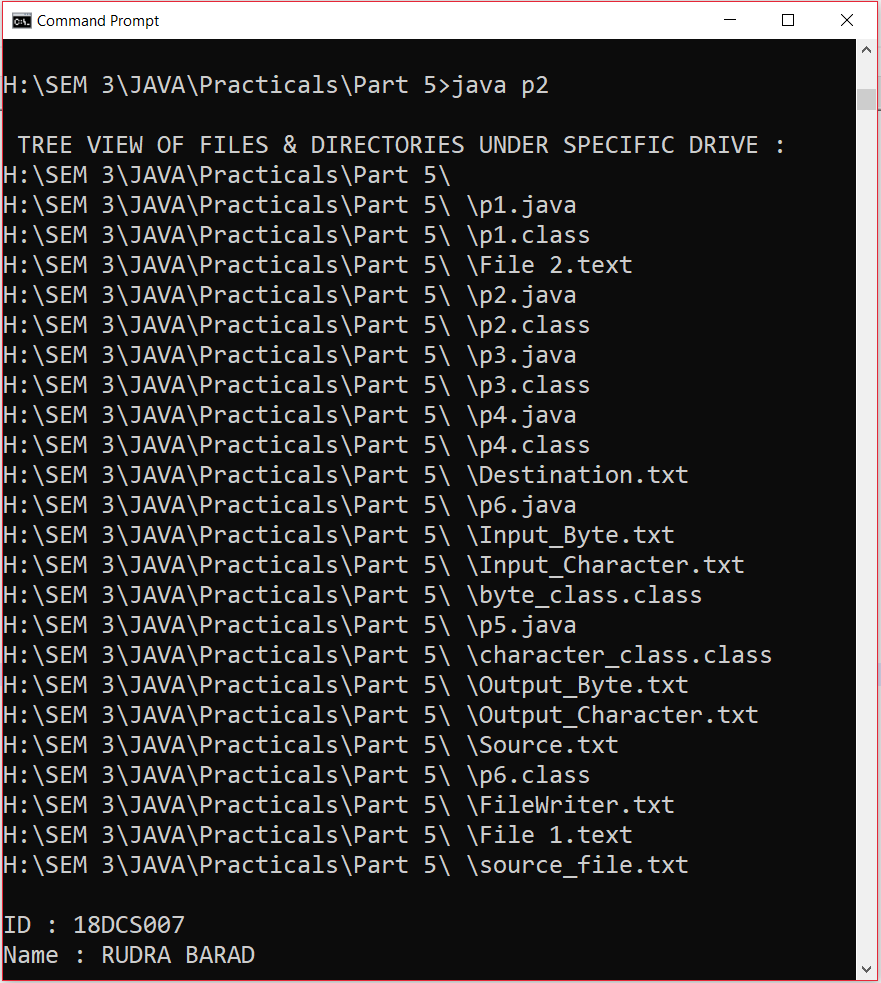
}

System.out.println("\nID : 18DCS007\nName : RUDRA BARAD");

}

}

**OUTPUT:**



**CONCLUSION:**

In this practical we learned how to show directories in tree view.

**PRACTICAL – 3**

**AIM:**

Write a Java program that reads on file name from the user, then displays

information about whether the file exists, whether the file is readable, whether

the file is writable, the type of file and the length of the file in bytes?

**PROGRAM CODE:**

import java.util.\*;

import java.io.File;

class p3

{

public static void main(String args[])

{

try

{

Scanner sc = new Scanner(System.in);

System.out.print("\nEnter The Name Of File You Want To Create : ");

String name = sc.next();

File obj = new File(name);

obj.createNewFile();

if(obj.exists())

{

System.out.println("\nFILE EXISTS :)");

System.out.println("Is File Readable : "+obj.canRead());

System.out.println("Is File Writable : "+obj.canWrite());

System.out.println("Is File Executable : "+obj.canExecute());

System.out.println("\nAfter Changing The Permissions to FALSE ");

obj.setReadable(false);

System.out.println("Is File Readable : "+obj.canRead());

obj.setWritable(false);

System.out.println("Is File Writable : "+obj.canWrite());

obj.setExecutable(false);

System.out.println("Is File Executable : "+obj.canExecute());

}

else

{

System.out.println("File Doesn't Exist :(");

}

}

catch(Exception e)

{

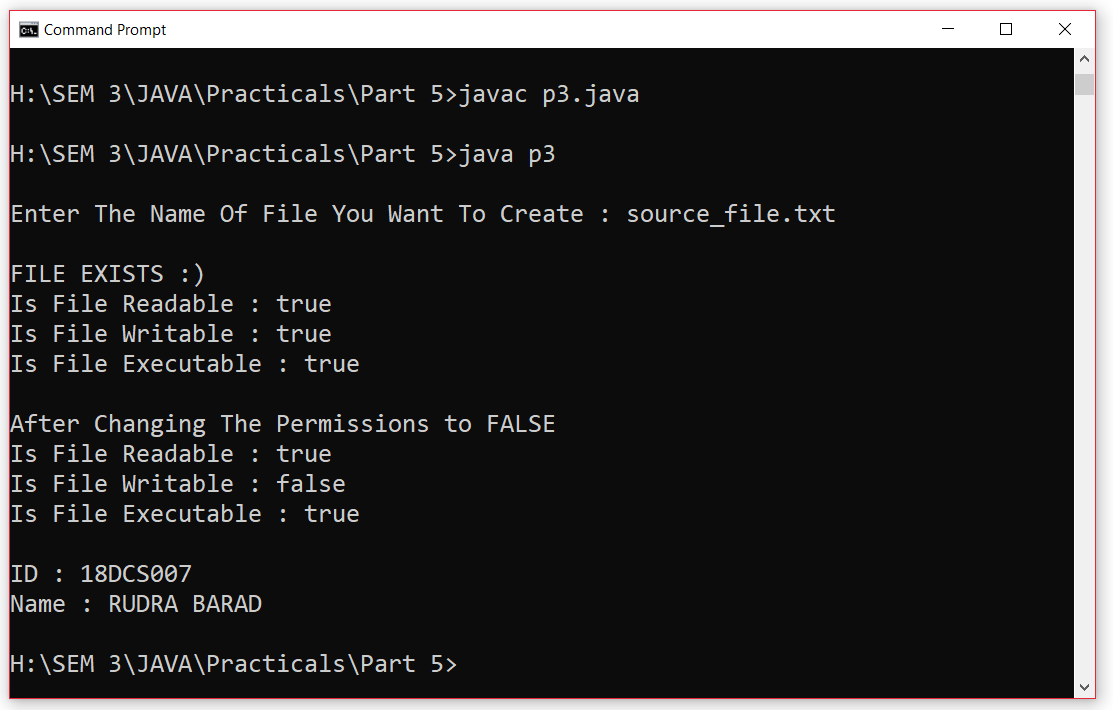
System.out.println("Exception is : " + e);

}

System.out.println("\nID : 18DCS007\nName : RUDRA BARAD");

}

}

**OUTPUT:**

**CONCLUSION:**

In this practical we created a file and gave it name from console and checked its others

attributes.

**PRACTICAL – 4**

**AIM:**

Write a program to transfer data from one file to another file so that if the

destination file does not exist, it is created.

**PROGRAM CODE:**

import java.io.FileInputStream;

import java.io.FileOutputStream;

class p4

{

public static void main(String args[])

{

try

{

FileInputStream fin = new FileInputStream("Source.txt");

FileOutputStream fout = new FileOutputStream("Destination.txt");

int c=0;

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

{

fout.write(c);

}

fin.close();

fout.close();

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

}

catch(Exception e)

{

System.out.println("Exception is : " + e);

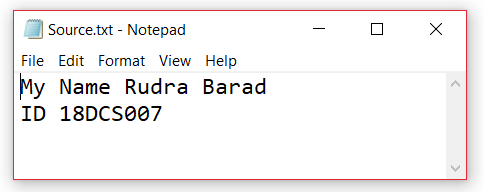
}

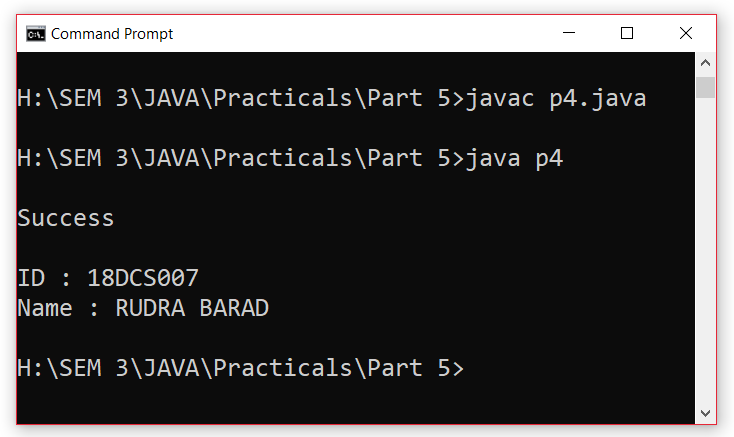
System.out.println("\nID : 18DCS007\nName : RUDRA BARAD");

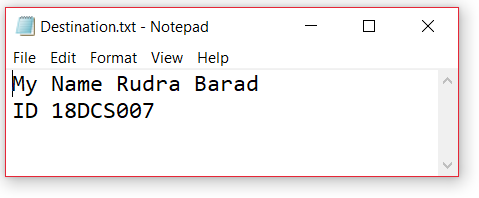
}

}

**OUTPUT:**







**CONCLUSION:**

In this practical we learned to copy a text from one file to another.

**PRACTICAL – 5**

**AIM:**

WAP to show use of character and byte stream.

**PROGRAM CODE:**

import java.io.\*;

class byte\_class

{

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

{

FileInputStream fin = null;

FileOutputStream fout = null;

try

{

fin = new FileInputStream("Input\_Byte.txt");

fout = new FileOutputStream("Output\_Byte.txt");

int b=0;

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

{

fout.write(b);

}

}

finally

{

if(fin != null)

fin.close();

if(fout != null)

fout.close();

System.out.println("\nSuccessfull For BYTE STREAM.");

}

}

}

class character\_class

{

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

{

FileReader fr = null;

FileWriter fw = null;

try

{

fr = new FileReader("Input\_Character.txt");

fw = new FileWriter("Output\_Character.txt");

int c=0;

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

{

fw.write(c);

}

}

finally

{

if(fr != null)

fr.close();

if(fw != null)

fw.close();

System.out.println("\nSuccessfull For CHARACTER STREAM.");

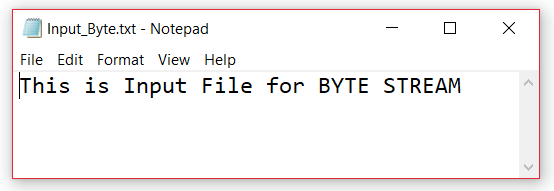
}

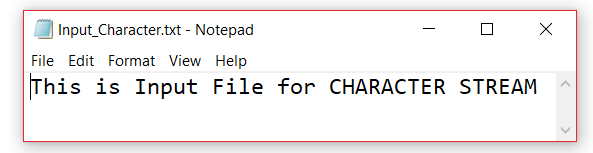
System.out.println("\nID : 18DCS007\nName : RUDRA BARAD");

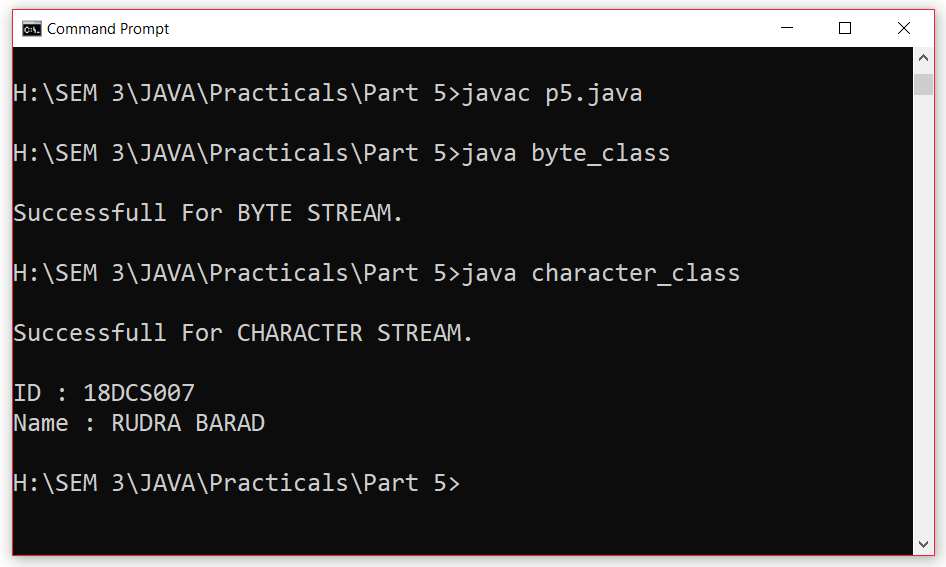
}

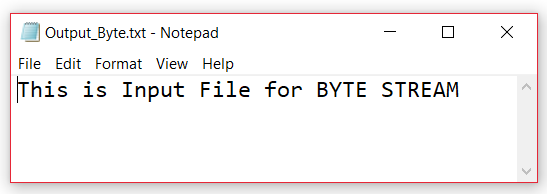
}

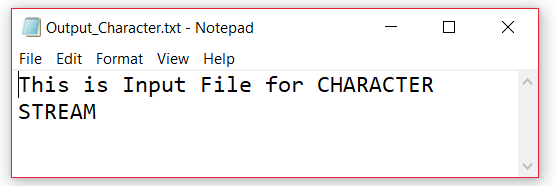
**OUTPUT:**











**CONCLUSION:**

We learned how to use Byte Stream as well as Character Stream.

**PRACTICAL – 6**

**AIM:**

WAP to read console input and write them into a file.

(BufferedReader /BufferedWriter).

**PROGRAM CODE:**

import java.io.\*;

class p6

{

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

{

try

{

FileWriter fw = new FileWriter("FileWriter.txt");

BufferedWriter bfwr = new BufferedWriter(fw);

bfwr.write("This is First Line");

bfwr.newLine();

bfwr.write("This is Second Line");

bfwr.newLine();

bfwr.write("This is Third Line");

bfwr.newLine();

bfwr.close();

FileReader fr = new FileReader("FileWriter.txt");

BufferedReader bfr = new BufferedReader(fr);

System.out.println(bfr.readLine());

System.out.println(bfr.readLine());

System.out.println(bfr.readLine());

}

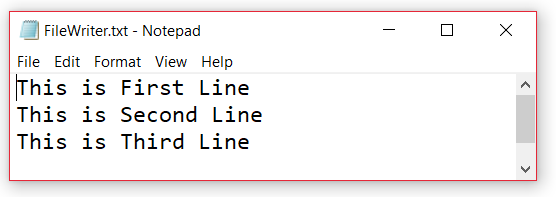
catch(Exception e)

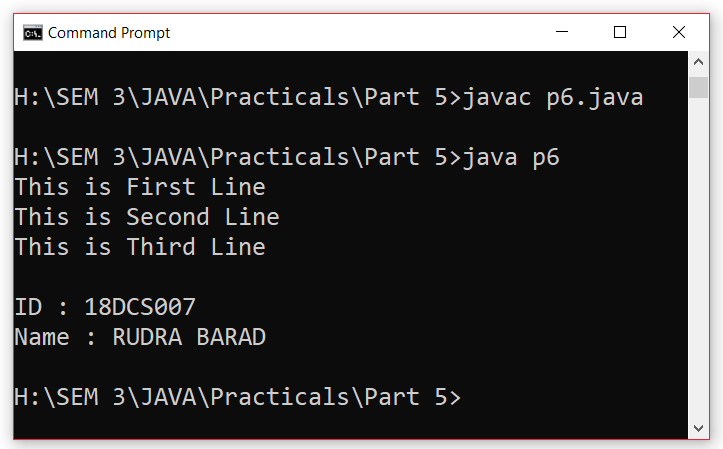
{

System.out.println("Exception is : " + e);

}

System.out.println("\nID : 18DCS007\nName : RUDRA BARAD");

**OUTPUT:**



**CONCLUSION:**

I learnt how to read from a file and write input from console using File Reader and File Writer.

**PRACTICAL – 7**

**AIM:**

WAP to demonstrate methods of wrapper class.

**PROGRAM CODE:**

public class java33

{

public static void main(String[] args)

{

Integer i = new Integer(40);

System.out.println("String of integer => " + i.toString());

System.out.println("Binary of integer => " + Integer.toBinaryString(i));

Double d = new Double(40.0);

System.out.println("String of double => " + d.toString());

System.out.println("Byte Code of double => " + d.byteValue());

Character c = new Character('c');

System.out.println("Is C letter => " + Character.isLetter(c));

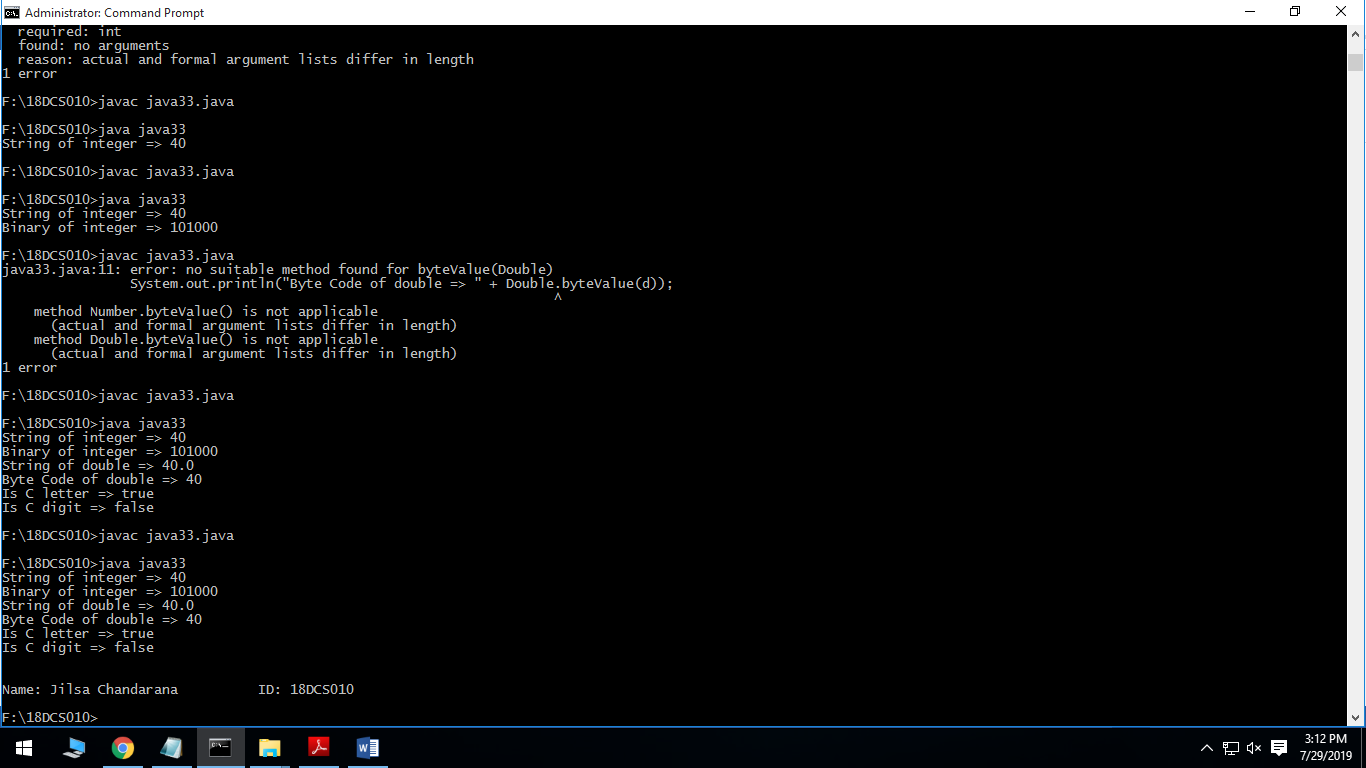
System.out.println("Is C digit => " + Character.isDigit(c));

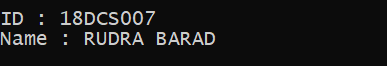
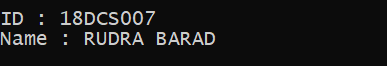
System.out.println("Name : RUDRA BARAD\t\t\tID : 18DCS007");

}

}

**OUTPUT:**





**CONCLUSION:**

In this practical, we learned about wrapper classes and called its methods.

**PART-VI**

**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 p1

{

public static void main(String[]args)

{

example1 e1= new example1();

e1.start();

example2 e2=new example2();

Thread t= new Thread(e2);

t.start();

}

}

class example1 extends Thread

{

public void run()

{

System.out.println("By extending thread class");

}

}

class example2 implements Runnable

{

public void run()

{

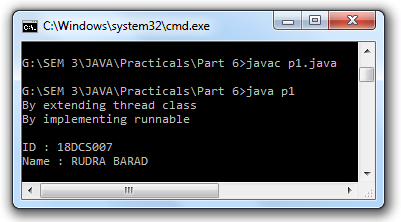
System.out.println("By implementing runnable ");

System.out.println("\nID : 18DCS007\nName : RUDRA BARAD");

}

}

**OUTPUT:**



**CONCLUSION:**

In this practical we learned how to make thread by implementing runnable interface and extending thread class

**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 p2

{

public static void main(String[] args)

{

Scanner sc=new Scanner(System.in);

System.out.print("Enter the number :");

int N=sc.nextInt();

System.out.print("Enter the number of threads :");

int n=sc.nextInt();

Summation sm=new Summation();

int k,l=n,h=n;

if(N<n)

{

for(int i=1;i<l;i++)

{

h--;

if(N==h)

{

n=h;

break;

}

}

}

k=N/n;

int j=1,z=k;

if(N%n!=0)

{

int m=(N-(k\*n));

new MyThread(sm,(k\*n+1),N,m).start();

}

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

{

new MyThread(sm,j,k,z).start();

j=k+1;

k=k+z;

}

try

{

Thread.sleep(1000);

sm.show();

}

catch(Exception ex){}

}

}

class MyThread extends Thread

{

int start,end,iter;

Summation sm;

MyThread(Summation sm,int s,int e,int i)

{

this.sm=sm;

start=s;

end=e;

iter=i;

}

public void run()

{

sm.add(start,end,iter);

}

}

class Summation

{

public int sum=0;

public synchronized void add(int s,int e,int iter)

{

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

{

sum+=s;

s++;

}

}

public void show()

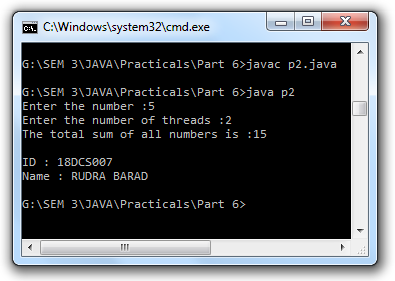
{

System.out.println("The total sum of all numbers is :"+sum);

System.out.println("\nID : 18DCS007\nName : RUDRA BARAD");

}

}

**OUTPUT:**

**CONCLUSION:**

In This practical we learned how to divide and perform task with multiple 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:**

public class p3

{

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

{

Thread randomThread = new Thread(new RandomGenThread());

randomThread.start();

System.out.println();

}

}

class RandomGenThread implements Runnable

{

double num;

public void run()

{

try

{

SquareThread sqt = new SquareThread();

Thread squareThread = new Thread(sqt);

CubeThread cbt = new CubeThread();

Thread threadCube = new Thread(cbt);

squareThread.start();

threadCube.start();

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

{

System.out.println("t1-"+i);

if(i%2 == 0)

{

sqt.setNum(new Double(i));

}

else

{

cbt.setNum(new Double(i));

}

Thread.sleep(1000);

}

}

catch (InterruptedException e)

{

e.printStackTrace();

}

}

}

class SquareThread implements Runnable

{

Double num;

public void run()

{

try

{

int i=0;

do

{

i++;

if(num != null&&num %2 ==0)

{

System.out.println("t2--->square of "+num+"="+(num\*num));

num = null;

}

Thread.sleep(1000);

}while(i<=5);

}

catch (Exception e)

{

e.printStackTrace();

}

}

public Double getNum()

{

return num;

}

public void setNum(Double num)

{

this.num = num;

}

}

class CubeThread implements Runnable

{

Double num;

public void run()

{

try

{

int i=0;

do

{

i++;

if(num != null&&num%2 !=0)

{

System.out.println("t3-->Cube of "+num+"="+(num\*num\*num));

num=null;

}

Thread.sleep(1000);

}

while(i<=5);

System.out.println("\nID : 18DCS007\nName : RUDRA BARAD");

}

catch (Exception e)

{

e.printStackTrace();

}

}

public Double getNum()

{

return num;

}

public void setNum(Double num)

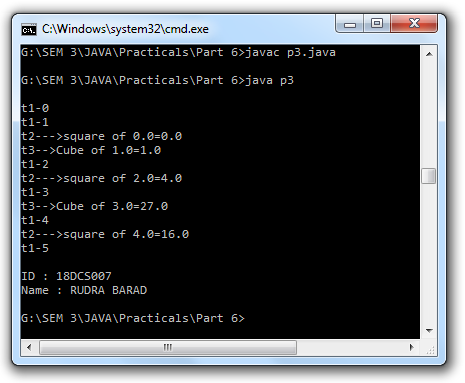
{

this.num = num;

}

}

**OUTPUT:**



**CONCLUSION:**

In this program we learned how to use multiple threads in a program.

**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:**

class p4

{

public static void main(String[]args)

{

example1 e1= new example1();

e1.start();

}

}

class example1 extends Thread

{

public void run()

{

int a=5;

try

{

for(int i=0;i<=10;i++)

{

System.out.println(a);

a++;

Thread.sleep(1000);

}

System.out.println("\nID : 18DCS007\nName : RUDRA BARAD");

}

catch(Exception e)

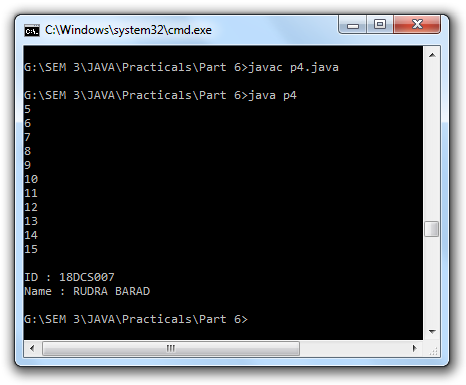
{

System.out.println(e);

}

}

}

**OUTPUT:**

**CONCLUSION:**

In this program we learned the working of 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 p5

{

public static void main(String[]args)

{

First f =new First();

Second s= new Second();

Third t= new Third();

System.out.println(f.getPriority());

System.out.println(s.getPriority());

System.out.println(t.getPriority());

f.setPriority(3);

s.setPriority(5);

t.setPriority(7);

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

System.out.println(f.getPriority());

System.out.println(s.getPriority());

System.out.println(t.getPriority());

f.start();

s.start();

t.start();

System.out.println("\nID : 18DCS007\nName : RUDRA BARAD");

}

}

class First extends Thread

{

public void run()

{

System.out.println("First");

}

}

class Second extends Thread

{

public void run()

{

System.out.println("second");

}

}

class Third extends Thread

{

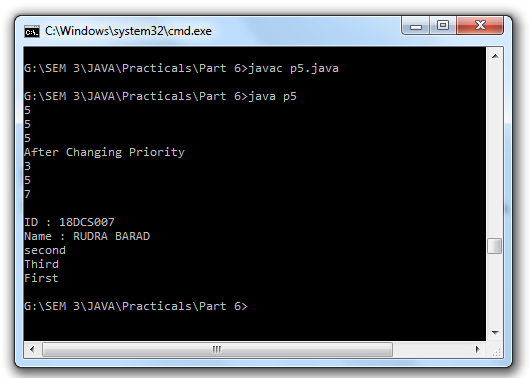
public void run()

{

System.out.println("Third");

}

}

**OUTPUT:**

**CONCLUSION:**

In this program we learned about checking and changing priority of threads.

**PRACTICAL – 6**

**AIM:**

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

**PROGRAM CODE:**

public class p6

{

public static void main(String[] args)

{

Shop c = new Shop();

Producer p1 = new Producer(c, 1);

Consumer c1 = new Consumer(c, 1);

p1.start();

c1.start();

}

}

class Shop

{

private int materials;

private boolean available = false;

public synchronized int get()

{

while (available == false)

{

try

{

wait();

}

catch (InterruptedException ie)

{

}

}

available = false;

notifyAll();

return materials;

}

public synchronized void put(int value)

{

while (available == true)

{

try

{

wait();

}

catch (InterruptedException ie)

{

ie.printStackTrace();

}

}

materials = value;

available = true;

notifyAll();

}

}

class Consumer extends Thread

{

private Shop Shop;

private int number;

public Consumer(Shop c, int number)

{

Shop = c;

this.number = number;

}

public void run()

{

int value = 0;

for (int i = 0; i < 10; i++)

{

value = Shop.get();

System.out.println("Consumed value " + this.number+ " got: " + value);

}

}

}

class Producer extends Thread

{

private Shop Shop;

private int number;

public Producer(Shop c, int number)

{

Shop = c;

this.number = number;

}

public void run()

{

for (int i = 0; i < 10; i++)

{

Shop.put(i);

System.out.println("Produced value " + this.number+ " put: " + i);

try

{

sleep((int)(Math.random() \* 100));

}

catch (InterruptedException ie)

{

ie.printStackTrace();

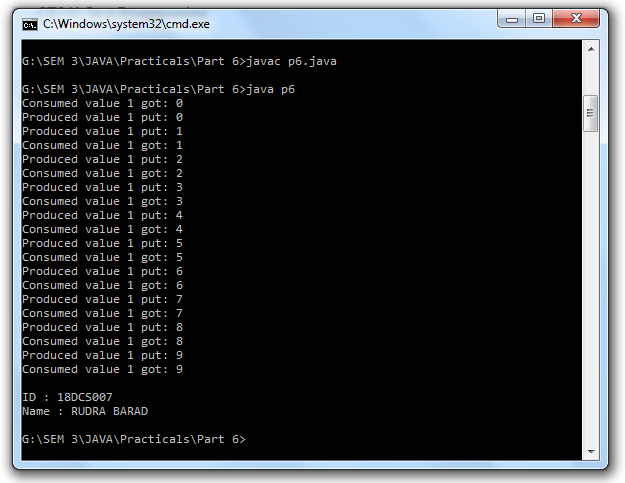
}

}

System.out.println("\nID : 18DCS007\nName : RUDRA BARAD");

}

}

**OUTPUT:**

**CONCLUSION:**

In this program we learned the concept of synchronization and solved the consumer product problem using it.

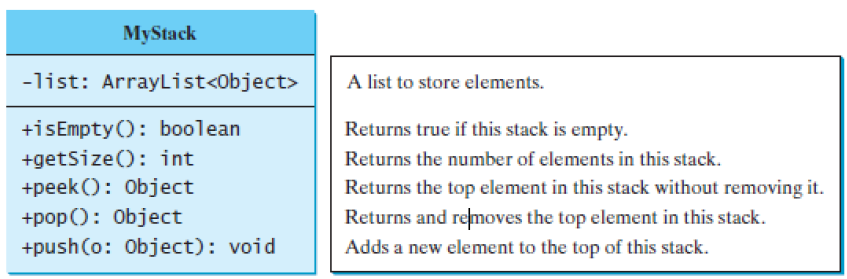
**PART-VII**

**Collection Framework and Generic**

**PRACTICAL – 1**

**AIM:**

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



**PROGRAM CODE:**

import java.util.ArrayList;

class p1

{

public static void main(String[] args)

{

ArrayList<Integer> a = new ArrayList<Integer>();

System.out.println("Is Arraylist empty => " + a.isEmpty());

a.add(1);

a.add(2);

a.add(3);

a.add(4);

a.add(5);

System.out.println("Items added \nSize of Arraylist => " + a.size());

System.out.println("First element to remove => " + a.get(0));

a.remove(0);

System.out.println("After removing first element => " + a.get(0));

System.out.println("Adding 0 on the top => ");

a.add(0,0);

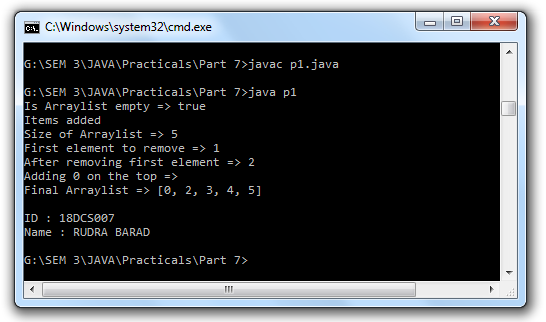
System.out.println("Final Arraylist => " + a);

System.out.println("\nID : 18DCS007\nName : RUDRA BARAD");

}

}

**OUTPUT:**



**CONCLUSION:**

In this practical, we learned about ArrayList and functions of ArrayList.

**PRACTICAL – 2**

**AIM:**

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

**PROGRAM CODE:**

import java.util.\*;

class p2

{

public static void main(String[] args)

{

ArrayList<Integer> a = new ArrayList<Integer>();

for(int i = 10; i > 0; i--)

{

a.add(i);

}

System.out.println("Unsorted Array: "+a);

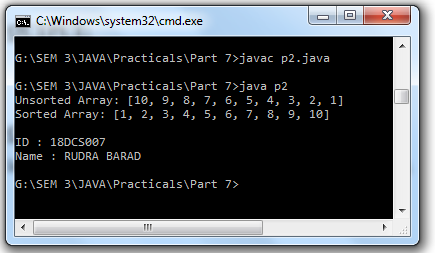
Collections.sort(a);

System.out.println("Sorted Array: " + a);

System.out.println("\nID : 18DCS007\nName : RUDRA BARAD");

}

}

**OUTPUT:**

**CONCLUSION:**

In this practical, we learned about sorting method of ArrayList.

**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.\*;

public class p3

{

public static void main(String[] args)

{

String text = "A B C E D A B C E D B C E A B D E D";

String[] list = text.split(" ");

Map map = new HashMap();

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

{

int cnt = 0;

for(int j = i; j < list.length; j++)

{

if(list[i].equals(list[j]))

{

cnt++;

}

}

map.put(list[i], cnt);

}

Set set = map.entrySet();

Iterator itr = set.iterator();

while(itr.hasNext())

{

Map.Entry entry = (Map.Entry) itr.next();

System.out.println("Key => " + entry.getKey() + "\t\t Value => " +

entry.getValue());

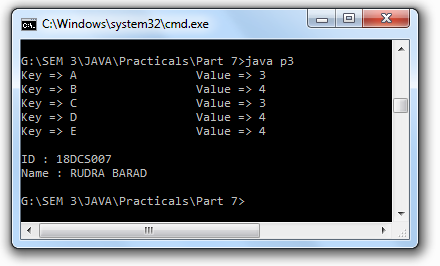
}

System.out.println("\nID : 18DCS007\nName : RUDRA BARAD");

}

}

**OUTPUT:**



**CONCLUSION:**

In this practical, we learned about Map and Set classes along with Iterator.

**PRACTICAL – 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 CODE:**

import java.util.\*;

import java.io.\*;

public class p4

{

public static void main(String[] args)

{

Map map = new HashMap();

map.put(1,"abstract");

map.put(2,"assert");

map.put(3,"Boolean");

map.put(4,"break");

map.put(5,"byte");

map.put(6,"case");

map.put(7,"catch");

map.put(8,"char");

map.put(9,"class");

map.put(10,"const");

map.put(11,"continue");

map.put(12,"default");

map.put(13,"do");

map.put(14,"double");

map.put(15,"else");

map.put(16,"enum");

map.put(17,"extends");

map.put(18,"final");

map.put(19,"finally");

map.put(20,"float");

map.put(21,"for");

map.put(22,"goto");

map.put(23,"if");

map.put(24,"implements");

map.put(25,"import");

map.put(26,"instanceof");

map.put(27,"int");

map.put(28,"interface");

map.put(29,"long");

map.put(30,"native");

map.put(31,"new");

map.put(32,"package");

map.put(33,"private");

map.put(34,"protected");

map.put(35,"public");

map.put(36,"return");

map.put(37,"short");

map.put(38,"static");

map.put(39,"strictfp");

map.put(40,"super");

map.put(41,"switch");

map.put(42,"synchronized");

map.put(43,"this");

map.put(44,"throw");

map.put(45,"throws");

map.put(46,"transient");

map.put(47,"try");

map.put(48,"void");

map.put(49,"volatile");

map.put(50,"while");

map.put(51," true");

map.put(52," false");

map.put(53,"null");

try

{

int cnt = 0;

BufferedReader reader = new BufferedReader(new FileReader("p3.java"));

String line = reader.readLine();

while(line != null)

{

String[] list = line.split(" ");

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

{

if(map.containsValue(list[i]))

{

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

cnt++;

}

}

line = reader.readLine();

}

System.out.println("Count of keywords => " + cnt);

}

catch(Exception e)

{

System.out.println("Exception => " + e.toString());

}

finally

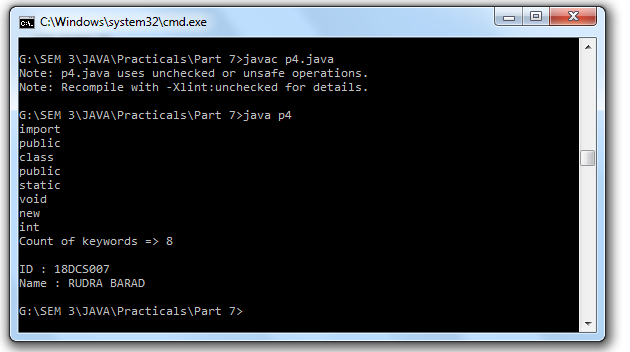
{

System.out.println("\nID : 18DCS007\nName : RUDRA BARAD");

}

}

}

**OUTPUT:**

**CONCLUSION:**

In this practical, we learned about contains method of hashmap.