**CE251 JAVA PROGRAMMING**

**CHAROTAR UNIVERSITY OF SCIENCE & TECHNOLOGY**

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

**Department of Computer Science & Engineering**

**PRACTICAL – 1(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 Netbeans IDE, or BlueJ and Console Programming.**

**CODE:**

* **Object Oriented Concepts:**
* Class
* Object
* Encapsulation
* Inheritance
* Polymorphism
* Abstraction
* **Comparison of Java with other object-oriented programming languages.**

|  |  |
| --- | --- |
| **Java** | **C++** |
| No use of pointers. Supports references, thread and interfaces. | Use of pointers, structures, union. |
| Wide range of classes for various high level services. | Comparatively available with low level functionalities. |
| Multiple inheritance is partially done through interfaces. | Provide both single and multiple inheritance. |
| It doesn’t support this feature. | Supports operator overloading. |
| Platform independent, byte code generated works on every OS. | Platform dependent, must be recompiled for different platform. |

|  |  |
| --- | --- |
| **Java** | **Python** |
| Java is both compiled and interpreted language, which is first compiled and then interpreted into a byte code. | Python is an interpreted programming language. |
| Define particular block by curly braces, end statements by ; | No need of semi colons and curly braces, uses indentation. |
| Strongly typed, need to define the exact datatype of variables. | Dynamic, no need to define the exact datatype of variables. |
| Multiple inheritance is partially done through interfaces. | Provide both single and multiple inheritance. |
| Java is much faster than python in terms of speed. | Expected to run slower than Java programs. |

* **Introduction to JDK, JRE, JVM, Javadoc:**
* **JDK** – **Java Development Kit** (in short JDK) is Kit which provides the environment to **develop and execute(run)** the Java program. JDK is a kit (or package) which includes two things
  1. Development Tools (to provide an environment to develop your java programs)
  2. JRE (to execute your java program).
* **JRE** – **Java Runtime Environment** (to say JRE) is an installation package which provides environment to **only run(not develop)** the java program(or application)onto your machine. JRE is only used by them who only wants to run the Java Programs i.e. end users of your system.
* **JVM** - **Java Virtual machine** (JVM) is a very important part of both JDK and JRE because it is contained or inbuilt in both. Whatever Java program you run using JRE or JDK goes into JVM and JVM is responsible for **executing the java program line by line** hence it is also known as interpreter.
* **JVM** - **Java Virtual machine** (JVM) is a very important part of both JDK and JRE because it is contained or inbuilt in both. Whatever Java program you run using JRE or 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 tool is a document generator tool in Java programming language for generating standard documentation in HTML format. It generates API documentation. It parses the declarations ad documentation in a set of source file describing classes, methods, constructors, and fields.**

* **Command line argument.**
* The java command-line argument is an argument i.e. passed at the time of running the java program.
* The arguments passed from the console can be received in the java program and it can be used as an input.
* **Introduction to Eclipse or NetBeans** **IDE, or BlueJand Console Programming.**
* **Introduction to Eclipse:**  We use application Eclipse to edit, test, and run Java programs. It is called an IDE (Integrated Development Environment. In this little module, we introduce you to the basics of Eclipse. You will not completely understand the Java code we write; just concentrate on how we are using Eclipse. Besides showing you how to use the IDE in a basic fashion, we give instructions on how to set various preferences, e.g. ensuring that line numbers appear in the editing panel.
* **Introduction to NetBeans:**  NetBeans IDE is a free, open source, integrated development environment (IDE) that enables you to develop desktop, mobile and web applications. The IDE supports application development in various languages, including Java, HTML5, PHP and C++. The IDE provides integrated support for the complete development cycle, from project creation through debugging, profiling and deployment. The IDE runs on Windows, Linux, Mac OS X, and other UNIX-based systems.
* **Introduction to BlueJ:**  BlueJ is a windows based platform for Java Development Kit (JDK). It is a free Java environment started in 1999 by Michael Kolling and John Rosenberg at Monash University, Australia, as a successor to Blue. It requires to install JDK version 1.3 or more before installing BlueJ. It can be freely downloaded from its official website of BlueJ.
* **Introduction to Console Programming:** A console application is a program which runs in an command prompt window. An example of a console application is below: Console programs do not have the flash, nor the event-driven capabilities of a Windows application, however, they still have their place.

**PRACTICAL – 1(2)**

**AIM:**

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

**SOURCE CODE:**

class Practical\_1\_2

{

public static *void* main(String[] *args*)

    {

*int* var=10;

        System.out.println("\""+var+" is the value of var.\"");

    }

}

**OUTPUT:**

****

**CONCLUSION:**

* In this practical we have assigned the value to the variable and then try to print the value by using the System.out.println statement.
* To print the assigned value of variable we will use the “+”symbol.

**PRACTICAL – 1(3)**

**AIM:**

|  |
| --- |
| **Write a console program to declare and initialize a double variable with some value such as 1234.5678. Then retrieve the integral part of the value and store it in a variable of type long, and the first four digits of the fractional part and store them in an integer of type short. Display the value of the double variable by outputting the two values stored as integers.** |

**SOURCE CODE:**

class Practical\_1\_3 {

    public static *void* main(String[] *args*) {

*double* d = 1234.5678;

*long* y = (*long*) (d);

*short* s;

        System.out.println(y);

        s = (*short*) ((d - y) \* 10000);

        System.out.println(s);

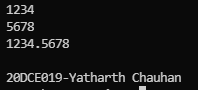
        System.out.println(y + "." + s);

        System.out.println("\n20DCE019-Yatharth Chauhan");

    }

}

**OUTPUT:**

****

**CONCLUSION:**

* We learnt type conversion in JAVA and used

“+” operator to concatenate two words/strings.

* In this practical we have stored the value in double variable and then by taking the modulo and adding the long variable we store the variable as short and try to the value of it.

**PRACTICAL – 1(4)**

**AIM:**

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

**SOURCE CODE:**

class Practical\_1\_4 {

    public static *void* main(String[] *args*) {

*int* arr[][] = { { 1 }, { 1, 2 }, { 1, 2, 3 } };

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

            System.out.println(arr[i].length);

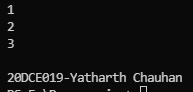
            System.out.println("\n20DCE019-Yatharth Chauhan");

        }

    }

}

**OUTPUT:**

****

**CONCLUSION:**

* In this practical we learnt the concept of array in java

**PRACTICAL – 1(5)**

**AIM:**

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

**SOURCE CODE:**

import java.util.Scanner;

public class Practical\_1\_5 {

    public static *void* main(String[] *args*) {

*int* rs;

*short* code[] = { 1, 2, 3, 4, 5 };

*float* tax[] = { 8, 12, 5, 7.5f, 3 };

*int* price[] = { 149, 299, 49, 449, 799 };

        System.out.println("1. Motor (Rs.149 + Tax = Rs.8)");

        System.out.println("2. fan (Rs.299 + Tax = Rs.12)");

        System.out.println("3. Tube (Rs.49 + Tax = Rs.5)");

        System.out.println("4. Wire (Rs.449 + Tax = Rs. 7.5)");

        System.out.println("5. Other Product (Rs.799 + Tax = Rs.3)");

        System.out.println("\nEnter Product Code: ");

        Scanner sc = new Scanner(System.in);

*int* i = sc.nextInt();

        switch (i) {

          case 1:

          rs = (*int*) (price[i - 1] + price[i - 1] \* tax[i - 1] / 100);

          break;

          case 2:

          rs = (*int*) (price[i - 1] + price[i - 1] \* tax[i - 1] / 100);

          break;

          case 3:

          rs = (*int*) (price[i - 1] + price[i - 1] \* tax[i - 1] / 100);

          break;

          case 4:

          rs = (*int*) (price[i - 1] + price[i - 1] \* tax[i - 1] / 100);

          break;

          default:

          rs = (*int*) (price[i - 1] + price[i - 1] \* tax[i - 1] / 100);

        }

        System.out.println("\n<----Bill---->");

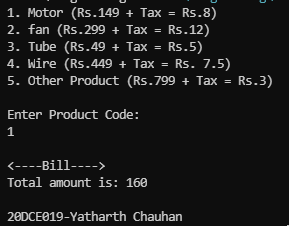
        System.out.println("Total amount is: " + rs);

        System.out.println("\n20DCE019-Yatharth Chauhan");

    }

}

**OUTPUT:**

****

**CONCLUSION:**

* In this Practical We learnt the use of switch statement.

**PRACTICAL – 1(6)**

**Aim:**

**Write a program to show output like:**

**\* \* \* \* \***

**\* \* \* \***

**\* \* \***

**\* \***

**\***

**SOURCE CODE:**

public class Practical\_1\_6 {

        public static *void* main(String[] *args*)

        {

*int* i,j;

            for(i=5;i>0;i--)

            {

                for(j=0;j<i;j++)

                {

                    System.out.print(" \* ");

                }

                System.out.println(" ");

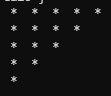
            }

            System.out.println("\n20DCE019-Yatharth Chauhan");

        }

}

**OUTPUT:**

****

**CONCLUSION:**

* In this practical, we learnt to use the for loop for printing a patten as shown**.**

**PRACTICAL – 2(1)**

**Aim: Given a string and a non-negative int n, we'll say that the front of the string is the first 3 chars, or whatever is there if the string is less than length 3. Return n copies of the front;**

**front\_times('Chocolate', 2) → 'ChoCho'**

**front\_times('Chocolate', 3) → 'ChoChoCho'**

**front\_times('Abc', 3) → 'AbcAbcAbc'**

**SOURCE CODE:**

import java.util.Scanner;

public class Practical\_2\_1 {

    public static *void* main(String[] *args*) {

        String res = " ", res1 = " ", res2 = " ";

        res = front\_times("Chocolate", 2);

        res1 = front\_times("Chocolate", 2);

        res2 = front\_times("Abc", 3);

        System.out.println(res);

        System.out.println(res1);

        System.out.println(res2);

        System.out.println("\n20DCE019-Yatharth Chauhan");

    }

    public static String front\_times(String *str*, *int* *n*) {

*int* fLen = 3;

        String front = *str*.substring(0, fLen);

        String res = " ";

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

            res = res + front;

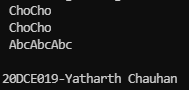
        }

        return res;

    }

}

**OUTPUT:**

****

**CONCLUSION:**

* In this practical, we came to know about repetition of string in java.

**PRACTICAL – 2(2)**

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

**array\_count9([1, 2, 9]) → 1**

**array\_count9([1, 9, 9]) → 2**

**array\_count9([1, 9, 9, 3, 9]) → 3**

**SOURCE CODE:**

import java.util.Scanner;

public class Practical\_2\_3 {

    public static *void* main(String[] *args*) {

        System.out.println("Enter Array Elements: ");

        Scanner sc = new Scanner(System.in);

*int* array[] = new *int*[5];

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

        {

            array[i] = sc.nextInt();

        }

*int* count = array\_count9(array);

        System.out.println("No. of lines in the given array is: "  + count);

        System.out.println("\n20DCE019-Yatharth Chauhan");

    }

    public static *int* array\_count9(*int* *ar*[]) {

*int* count = 0;

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

            if(*ar*[i] == 9)

            {

                count++;

            }

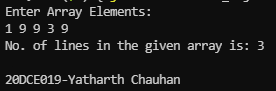
        }

        return count;

    }

}

**OUTPUT:**

****

**CONCLUSION:**

* In this practical we learnt about Arrays and java methods that we can use to solve a given problem.

**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**

**SOURCE CODE:**

import java.util.Scanner;

public class Practical\_2\_3 {

    public static *void* main(String[] *args*) {

        System.out.println("Enter Array Elements: ");

        Scanner sc = new Scanner(System.in);

*int* array[] = new *int*[5];

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

            array[i] = sc.nextInt();

        }

*boolean* count = array\_count9(array, array.length);

        System.out.println("count is: " + count);

        System.out.println("\n20DCE019-Yatharth Chauhan");

    }

    public static *boolean* array\_count9(*int* *ar*[], *int* *n*) {

*boolean* count = false;

        if (*n* > 4) {

*n* = 4;

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

                if (*ar*[i] == 9) {

                    count = true;

                }

            }

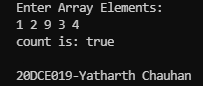
        }

        return count;

    }

}

**OUTPUT:**

****

**CONCLUSION:**

* In this practical we learnt about Arrays and java methods that we can use for arrays to achieve our goal.

**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'**

**SOURCE CODE:**

p public class Practical\_2\_4 {

    public static *void* main(String[] *args*) {

        String str1 = new String("The");

        String str2 = new String("AAbb");

        String str3 = new String("Hi--There");

        String str4 = new String(doublechar(str1));

        String str5 = new String(doublechar(str2));

        String str6 = new String(doublechar(str3));

        System.out.println(str4);

        System.out.println(str5);

        System.out.println(str6);

        System.out.println("\n20DCE019-Yatharth Chauhan");

    }

    public static String doublechar(String *str*) {

        String result = "";

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

            result = result + *str*.charAt(i) + *str*.charAt(i);

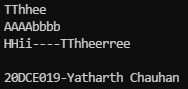
        }

        return result;

    }

}

**OUTPUT:**

****

**CONCLUSION:**

* In this practical we used pre-defined strings and then used doublechar method which takes an argument as string and return a string which has each elements of it doubled.

**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”.**

**SOURCE CODE:**

import java.util.Scanner;

public class Practical\_2\_5 {

    public static *void* main(String[] *args*) {

        System.out.println("Enter The String");

        Scanner sc = new Scanner(System.in);

        String S = sc.nextLine();

        String x[] = S.split("  ");

        String string\_r = new String();

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

            StringBuffer str = new StringBuffer(x[i]);

            string\_r = string\_r + str.reverse();

        }

        System.out.println("\nOriginal String is: " + S);

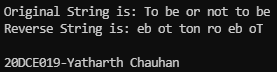
        System.out.println("Reverse String is: " + string\_r);

        System.out.println("\n20DCE019-Yatharth Chauhan");

    }

}

**OUTPUT:**

****

**CONCLUSION:**

* In this practical we inputted a string through Scanner class. Then we used split method to split the string with “ ” and then reversed each word using for loop.

**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**

**SOURCE CODE:**

import java.util.Arrays;

import java.util.Scanner;

public class Practical\_2\_6 {

    public static *void* main(String[] *args*) {

        String str\_1;

        Scanner sc = new Scanner(System.in);

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

        str\_1 = sc.nextLine();

        System.out.println("\nLength of String is: " + str\_1.length());

        System.out.println("Lower Case String is: " + str\_1.toLowerCase());

        System.out.println("Upper Case String is: " + str\_1.toUpperCase());

        StringBuilder rev = new StringBuilder();

        rev.append(str\_1);

        rev.reverse();

        System.out.println("Reverse String is: " + rev);

*char*[] chars = str\_1.toCharArray();

        Arrays.sort(chars);

        String sorted = new String(chars);

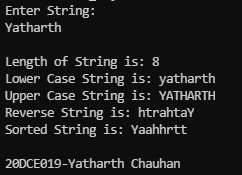
        System.out.println("Sorted String is: " + sorted);

        System.out.println("\n20DCE019-Yatharth Chauhan");

    }

}

**OUTPUT:**

****

**CONCLUSION:**

* In this practical we learned how to use length() method, toUpperCase(), toLowerCase() method, reverse() method & Array.

**PRACTICAL – 2(7)**

**AiM: Perform following Functionalities of the string: “CHARUSATUniversity”**

* **Find length**
* **Replace ‘H’ by ‘N’.**
* **Convert all character in Uppercase**
* **Extract and print “CHARUSAT” from given string.**

**SOURCE CODE:**

public class Practical\_2\_7 {

    public static *void* main(String[] *args*) {

        String str\_1 = "CHARUSAT University";

        System.out.println("Length of String is: "

 + str\_1.length());

        String r = str\_1.replace("H", "N");

        System.out.println("Updated New String is

(Replace H by N): " + r);

        System.out.println("Upper Case String is: "

 + str\_1.toUpperCase());

        String str\_2 = str\_1.substring(0, 8);

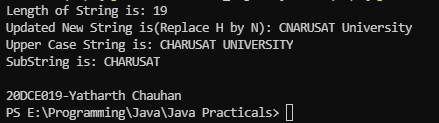
        System.out.println("SubString is: " + str\_2);

        System.out.println("\n20DCE019-Yatharth Chauhan");

    }

}

**OUTPUT:**

****

**CONCLUSION:** In this practical we can print the string with any element replaced, upper case of any string and we can also print the substring of any given string.

**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=100Rupees.)**

**SOURCE CODE:**

import java.util.Scanner;

public class Practical\_3\_1 {

    public static *void* main(*String* *args*[]) {

*Scanner* sc = new Scanner(System.in);

        System.out.println("Enter Pound Value: ");

*int* p;

        p = sc.nextInt();

*int* rs;

        rs = p \* 100;

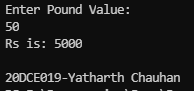
        System.out.println("Rs is: " + rs);

        System.out.println("\n20DCE019-Yatharth Chauhan");

    }

}

**OUTPUT:**

****

**CONCLUSION:** In this program we learnt about scanner class, how to import it and make its object and use it. And converted pound into rupees.

**PRACTICAL – 3(2)**

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

**­­­**

**SOURCE CODE:**

public class Practical\_3\_2 {

    public static *void* main(*String*[] *args*) {

*TriangleArea* x1 = new TriangleArea();

*TriangleArea* x2 = new TriangleArea(7.0);

*TriangleArea* x3 = new TriangleArea(2.3f, 6.0f);

        System.out.println("Area of square: " + x1.area);

        System.out.println("Area of circle: " + x2.area);

        System.out.println("Area of triangle: " + x3.area);

        System.out.println("\n20DCE019-Yatharth Chauhan");

    }

}

class TriangleArea {

*double* area;

    TriangleArea() {

        area = 10;

    }

    TriangleArea(*double* *radius*) {

        area = 3.14 \* radius \* radius;

    }

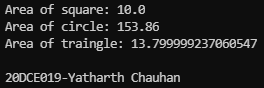
    TriangleArea(*float* *base*, *float* *altitude*) {

        area = base \* altitude;

    }

}

**OUTPUT:**

****

**CONCLUSION:** In this program we created a new class named triangle, and made three different types of constructor and called them by making objects.

**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 dis play each Employee’s**

**yearly salary again.**

**SOURCE CODE:**

import java.util.Scanner;

public class Practical\_3\_3 {

    public static *void* main(String *args*[]) {

        Employee x1 = new Employee();

        x1.set();

        x1.get();

        x1.raise(10);

    }

}

class Employee {

    String first\_name;

    String last\_name;

*double* salary;

    public *void* set() {

        Scanner sc = new Scanner(System.in);

        System.out.println("Enter Your First Name: ");

       first\_name = sc.nextLine();

        System.out.println("Enter Your Last Name: ");

        last\_name = sc.nextLine();

        System.out.println("Enter Your Salary: ");

        salary = sc.nextDouble();

        if (salary < 0) {

            salary = 0.0;

        }

        sc.close();

    }

    public *void* get() {

        System.out.println("\nName of Employee: " + first\_name + " " + last\_name);

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

    }

    public *void* raise(*double* *percentage*) {

*double* Raise;

        Raise = *percentage* \* salary;

        Raise /= 100;

        salary += Raise;

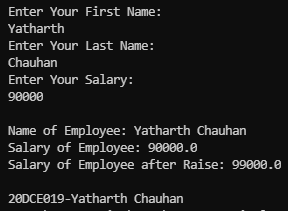
        System.out.println("Salary of Employee after Raise: " + salary);

        System.out.println("\n20DCE019-Yatharth Chauhan");

    }

}

**OUTPUT:**

****

**CONCLUSION:** In this program we learnt basic concepts of making class, defining methods inside them and calling it in other classes. We learnt how to use methods of other class.

**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.**

**SOURCE CODE:**

import java.util.Scanner;

public class Practical\_3\_4 {

    public static *void* main(String[] *args*) {

        date obj = new date();

        obj.set();

        obj.get();

    }

}

class date {

*int* date, month, year;

    public *void* set() {

        Scanner sc = new Scanner(System.in);

        System.out.println("Enter The Day: ");

        date = sc.nextInt();

        System.out.println("Enter The Month: ");

        month = sc.nextInt();

        System.out.println("Enter The Year: ");

        year = sc.nextInt();

        sc.close();

    }

    public *void* get() {

        // System.out.println("\n");

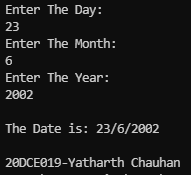
        System.out.println("\nThe Date is: " + date + "/" + month + "/" + year);

        System.out.println("\n20DCE019-Yatharth Chauhan");

    }

}

**OUTPUT:**

****

**CONCLUSION:** In this program we practiced other examples of class and methods here we took input in other class and called them using objects.

**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.**

**SOURCE CODE:**

import java.util.Scanner;

public class Practical\_3\_5 {

    public static *void* main(String *args*[]) {

        Scanner sc = new Scanner(System.in);

*int* l, b;

        System.out.println("Enter The Length of Rectangle: ");

        l = sc.nextInt();

        System.out.println("Enter The Breadth of Rectangle: ");

        b = sc.nextInt();

        area object = new area(l, b);

        System.out.println("\nArea of rectangle is: " + object.returnarea());

        System.out.println("\n20DCE019-Yatharth Chauhan");

    }

}

class area {

*int* length, breadth;

    area(*int* *l*, *int* *b*) {

        length = *l*;

        breadth = *b*;

    }

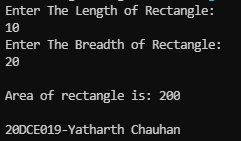
    public *int* returnarea() {

        return length \* breadth;

    }

}

**OUTPUT:**

****

**CONCLUSION:** In this program we used parameters in the function and we passed the values from one class to another, did some operations and then we returned its value to previous class.

**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.**

**SOURCE CODE:**

public class Practical\_3\_6 {

    public static *void* main(String[] *args*) {

        complex c1 = new complex(5, 3);

        c1.printcomplexnumber();

        complex c2 = new complex(4, 1);

        c2.printcomplexnumber();

        complex c3 = new complex();

        c3 = c3.addcomplex(c1, c2);

        System.out.print("\nsum is: ");

        c3.printcomplexnumber();

        c3 = c3.subcomplex(c1, c2);

        System.out.print("substraction is: ");

        c3.printcomplexnumber();

        c3 = c3.productcomplex(c1, c2);

        System.out.print("multiplication is: ");

        c3.printcomplexnumber();

        c3 = c3.dividecomplex(c1, c2);

        System.out.print("division is: ");

        c3.printcomplexnumber();

        System.out.println("\n20DCE019-Yatharth Chauhan");

    }

}

class complex {

*int* real, imagenery;

    complex() {

    }

    complex(*int* *tempReal*, *int* *imagnery*) {

        real = *tempReal*;

        imagenery = *imagnery*;

    }

*void* printcomplexnumber() {

        System.out.println(+real + " + " + imagenery + "i");

    }

    complex addcomplex(complex *c1*, complex *c2*) {

        complex temp = new complex();

        temp.real = *c1*.real + *c2*.real;

        temp.imagenery = *c1*.imagenery + *c2*.imagenery;

        return temp;

    }

    complex subcomplex(complex *c1*, complex *c2*) {

        complex temp = new complex();

        temp.real = *c1*.real - *c2*.real;

        temp.imagenery = *c1*.imagenery - *c2*.imagenery;

        return temp;

    }

    complex productcomplex(complex *c1*, complex *c2*) {

        complex temp = new complex();

        temp.real = *c1*.real \* *c2*.real;

        temp.imagenery = *c1*.imagenery \* *c2*.imagenery;

        return temp;

    }

    complex dividecomplex(complex *c1*, complex *c2*) {

        complex temp = new complex();

        temp.real = *c1*.real / *c2*.real;

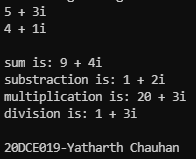
        temp.imagenery = *c1*.imagenery / *c2*.imagenery;

        return temp;

    }

}

**OUTPUT:**

****

**CONCLUSION:** In this practical we have print the sum, difference and product of two complex number

**PRACTICAL – 3(7)**

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

        Public class MethodOverloading

    {

               Private void methodoverloading()

               {

                                //No argument,private method

                }

               Private int methodOverloaded(int i)

               {                            //code

}

               Protected  int methOdoverloaded(double d)

                {

**//code**

}

Public void methodOverloading(int i, double d)

                 {

                                              //code

 }

}

**PROGRAM CODE:**

public class Practical\_3\_7 {

    private *void* methodoverloaded() {

        System.out.println("private void mathodoverloaded():");

    }

    private *int* methodoverloaded(*int* *i*) {

        System.out.println("private int mathodoverloaded(int):");

        return 0;

    }

    private *void* methodoverloaded(*double* *i*) {

        System.out.println("private int mathodoverloaded(double):");

    }

    public *void* methodoverloaded(*int* *i*, *double* *j*) {

        System.out.println("public void mathodoverloaded(int,double):");

        System.out.println("\n20DCE019-Yatharth Chauhan");

    }

    public static *void* main(String[] *args*) {

        Practical\_3\_7 s = new Practical\_3\_7();

        s.methodoverloaded();

        s.methodoverloaded(2);

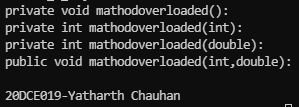
        s.methodoverloaded(12.22);

        s.methodoverloaded(2, 22.22);

    }

}

**OUTPUT:**

****

**CONCLUSION:** We get to know about the method overloading in Java.

**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**

**SOURCE CODE:**

class parent {

*void* A() {

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

    }

}

class child extends parent {

*void* B() {

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

    }

}

public class Practical\_4\_1 {

    public static *void* main(String[] *args*) {

        parent p = new parent();

        child c = new child();

        p.A();

        c.B();

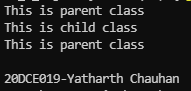
        p.A();

        System.out.println("\n20DCE019-Yatharth Chauhan");

    }

}

**OUTPUT:**

****

**CONCLUSION:** In this practical we can use object of parent and child class to call its own methods as well as we can call parent class method using object of child class by the use 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.**

**SOURCE CODE:**

class member {

*String* name, phonenum, address;

*int* salary, age;

*void* printSalary() {

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

    }

}

class employee extends member {

*String* specialization;

}

class manager extends member {

*String* department;

}

public class Practical\_4\_2 {

    public static *void* main(*String* *args*[]) {

        employee e1 = new employee();

        manager m1 = new manager();

        e1.name = "Yatharth Chauhan";

        e1.age = 19;

        e1.phonenum = "1234567899";

        e1.address = "Bharuch";

        e1.salary = 90000;

        e1.specialization = "java";

        m1.name = "Dev";

        m1.age = 19;

        m1.phonenum = "9876543211";

        m1.address = "Surat";

        m1.salary = 80000;

        m1.department = "Android Development";

        System.out.println("Employee detail:-");

        System.out.println("Name:-" + e1.name + "\nAge:-" + e1.age + "\nPhone number:-" + e1.phonenum + "\nAddress:-"

                + e1.address + "\nSpecialization:-" + e1.specialization);

        e1.printSalary();

        System.out.println();

        System.out.println("manager detail:-");

        System.out.println("Name:-" + m1.name + "\nAge:-" + m1.age + "\nPhone number:-" + m1.phonenum + "\nAddress:-"

                + m1.address + "\nDepartment:-" + m1.department);

        m1.printSalary();

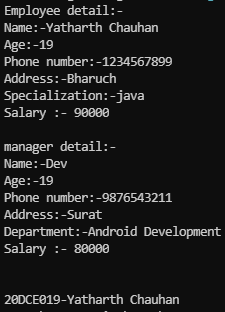
        System.out.println();

        System.out.println("\n20DCE019-Yatharth Chauhan");

    }

}

**OUTPUT:**

****

**Conclusion:** In this program I learnt about hierarchical inheritance. Using constructor and invoking methods of base class.

**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.**

**SOURCE CODE:**

public class Practical\_4\_3 {

    public static *void* main(String[] *args*) {

        Square[] obj = { new Square(), new Square(2.0f), new Square(5.0f) };

*int* i = 0;

        for (i = 0; i < 3; i++) {

            System.out.println("object" + (i + 1) + ":\n");

            obj[i].display();

        }

    }

}

class Rectangle {

*float* lenght, breadth;

    public Rectangle() {

    }

    public Rectangle(*float* *a*, *float* *b*) {

        lenght = *a*;

        breadth = *b*;

    }

}

class Square extends Rectangle {

*float* side;

    public Square() {

        super();

    }

    public Square(*float* *a*) {

        super(5, 5);

        side = 5;

    }

    public *void* display() {

        System.out.println("Area of rectangle:" + (lenght \* breadth));

        System.out.println("Perimeter of Rectangle:" + (2 \* (lenght + breadth)));

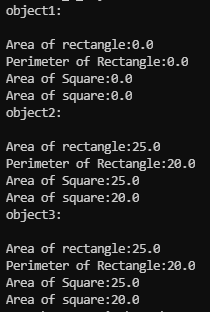
        System.out.println("Area of Square:" + (side \* side));

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

    }

}

**OUTPUT:**

****

**CONCLUSION:** In this program I learnt about invoking constructor of parent class in the constructor of child class. This is possible with the help of super keyword. It can also be used to access data members and member functions of parent class.

**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.**

**SOURCE CODE:**

class shape {

    public *void* print\_shape() {

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

    }

}

class rectangle extends shape {

    public *void* print\_rectangle() {

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

    }

}

class circle extends shape {

    public *void* ciecle() {

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

    }

}

class square extends rectangle {

    public *void* sqaure() {

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

    }

}

public class Practical\_4\_4 {

    public static *void* main(String[] *args*) {

        square sq = new square();

        sq.print\_shape();

        sq.print\_rectangle();

        System.out.println("\n20DCE019-Yatharth Chauhan");

    }

}

**OUTPUT:**

****

**CONCLUSION:** In this program I learnt about multiple inheritance and using super keyword to provoke methods of parent class.

**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.**

**SOURCE CODE:**

class degree {

*void* getDegree() {

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

    }

}

class undergrad {

*void* getDegree() {

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

    }

}

class postgrad {

*void* getDegree() {

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

    }

}

public class Practical\_4\_5 {

    public static *void* main(String *args*[]) {

        degree d1 = new degree();

        undergrad u1 = new undergrad();

        postgrad p1 = new postgrad();

        d1.getDegree();

        u1.getDegree();

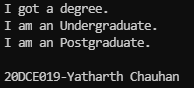
        p1.getDegree();

        System.out.println("\n20DCE019-Yatharth Chauhan");

    }

}

**OUTPUT:**

****

**CONCLUSION:** I learnt how Call the method by creating an object from this program.

**PRACTICAL – 4(6)**

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

**Called MyCalculator which implements the interface. divisorSum function just takes an integer as input and return the sum of all its divisors. For example divisors of 6 are 1, 2, 3 and 6, so divisor sum should return 12. The value of n will be at most 1000.**

**SOURCE CODE:**

import java.util.Scanner;

interface AdvancedArithmetic {

*int* divisorSum(*int* *n*);

}

class MyCalculator implements *AdvancedArithmetic* {

    public *int* divisorSum(*int* *n*) {

        if (n <= 1) {

            return n;

        }

*int* res = n + 1;

        for (*int* i = 2; i < n; i++) {

            if (n % i == 0) {

                res += i;

            }

        }

        return res;

    }

}

class Practical\_4\_6 {

    public static *void* main(*String* *args*[]) {

*int* m;

*Scanner* sc = new Scanner(System.in);

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

        m = sc.nextInt();

*AdvancedArithmetic* a1 = new MyCalculator();

        System.out.print("Sum of divisors of " + m + " : ");

        System.out.println(a1.divisorSum(m));

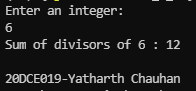
        System.out.println("\n20DCE019-Yatharth Chauhan");

        sc.close();

    }

}

**OUTPUT:**

****

**CONCLUSION:** In this practical we learnt Concept of interface.

**PRACTICAL – 4(7 and 8)**

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

**Radius and a color) or rectangles (with a length, width, and color). You also**

**want to beable 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.**

**SOURCE CODE:**

import java.util.Scanner;

class Practical\_4\_7 {

    public static *void* main(String *args*[]) {

        Scanner in = new Scanner(System.in);

*int* n;

        System.out.println("Enter Your Choice\n1. Circle\n2. Rectangle\n");

        n = in.nextInt();

        switch (n) {

            case 1:

                Circle c = new Circle();

                c.get();

                c.input();

                c.output();

                break;

            case 2:

                Rectangle o = new Rectangle();

                o.get();

                o.input();

                o.output();

                break;

            default:

                System.out.println("\nInvalid Choice");

        }

        in.close();

    }

}

class signs {

*double* area;

    String colour, text;

    public *void* get() {

        Scanner in = new Scanner(System.in);

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

        colour = in.nextLine();

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

        text = in.nextLine();

        in.close();

    }

}

interface Shape {

*void* input();

*void* output();

}

class Circle extends signs implements Shape {

*double* radius;

    public *void* input() {

        Scanner in = new Scanner(System.in);

        System.out.print("\nEnter Radius: ");

        radius = in.nextDouble();

        area = 3.14 \* radius \* radius;

        in.close();

    }

    public *void* output() {

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

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

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

        System.out.println("Radius " + radius);

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

    }

}

class Rectangle extends signs implements Shape {

*double* length, breadth;

    public *void* input() {

        Scanner in = new Scanner(System.in);

        System.out.println("\nEnter Length and breadth : ");

        length = in.nextDouble();

        breadth = in.nextDouble();

        area = length \* breadth;

        in.close();

    }

    public *void* output() {

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

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

        System.out.println("COLOUR: " + colour);

        System.out.println("Length " + length);

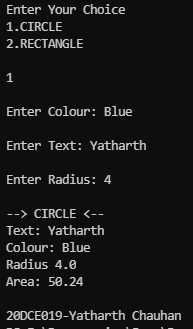
        System.out.println("Breadth " + breadth);

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

    }

}

**OUTPUT:**

****

**CONCLUSION:** In this practical we learnt about significance of interface.

**PRACTICAL – 4(9)**

**Aim: Write a java program which shows importing of classes from other user**

**define packages.**

**SOURCE CODE:**

package package\_4\_9;

public class package\_4\_9 {

    public static *void* hello() {

        System.out.println("This is the hello method in pack class which is imported");

    }

    public static *void* main(String *args*[]) {

    }

}

**MAIN CLASS:**

package package\_4\_9;

public class Practical\_4\_9 {

    public static *void* main(String *args*[]) {

        package\_4\_9.hello();

        System.out.println("\n20DCE019-Yatharth Chauhan");

    }

}

**OUTPUT:**

**Text

Description automatically generated**

**CONCLUSION:** In this practical we learnt about packages.

**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.**

**SOURCE CODE:**

import java.util.Scanner;

public class Practical\_5\_1 {

    public static *void* main(String[] *args*) {

        Scanner sc = new Scanner(System.in);

*int* a, b, c;

        try {

            System.out.println("Enter Integer 1:");

            a = sc.nextInt();

            System.out.println("Enter Integer 2:");

            b = sc.nextInt();

            c = a / b;

            System.out.println("Division is: " + c);

            System.out.println("\n20DCE019-Yatharth Chauhan");

        } catch (Exception e) {

            System.out.println("\nOOPS! An error occured:" + e);

            System.out.println("\n20DCE019-Yatharth Chauhan");

        }

        sc.close();

    }

}

**OUTPUT:**

**Text

Description automatically generated**

**CONCLUSION:** We learnt to pass the values and check exceptions.

**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".**

**String parsed to a numeric variable: Print "Format**

**mismatch". Accessing an invalid index in string: Print**

**"Index is invalid".**

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

**MyException: This is a user defined Exception which you need to create. It**

**takes a parameter param. When an exception of this class is encountered, the**

**handler should print "MyException[param]", here param is the parameter passed to the exception class.**

**Exceptions other than mentioned above: Any other exception except**

**the above ones fall in this category. Print "Exception encountered".**

**Finally, after the exception is handled, print "Exception**

**Handling Completed".**

**Example: For an exception of MyException class if the parameter value is**

**5, the message will look like**

**MyException[5].**

**SOURCE CODE:**

import java.util.\*;

class myexception extends Exception {

    public myexception(String *s*) {

        super(*s*);

    }

}

public class Practical\_5\_2 extends Exception {

    public static *void* main(String[] *args*) {

*int* a, b, ans;

        Scanner sc = new Scanner(System.in);

        try {

            a = sc.nextInt();

            b = sc.nextInt();

            ans = a / b;

            String s = new String();

            s = "Name";

            System.out.println("Answer: " + ans);

            System.out.println(s.charAt(2));

            sc.close();

            throw new myexception("hello");

        } catch (myexception E) {

            System.out.println(E.getMessage());

            System.out.println("My Exception[" + E.getMessage() + "]");

        } catch (ArithmeticException e) {

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

        } catch (InputMismatchException e) {

            System.out.println("format mismatch");

        } catch (StringIndexOutOfBoundsException e) {

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

        } catch (ArrayIndexOutOfBoundsException e) {

            System.out.println("Array index is invalid");

        } finally {

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

        }

        System.out.println("\n20DCE019 - Yatharth Chauhan");

    }

}

**OUTPUT:**

**Text

Description automatically generated**

**CONCLUSION:** In this program earnt how to handle exceptions using nested try and catch block along with finally block.

**PRACTICAL – 5(3)**

**Aim: Write a java program to generate user defined exception using**

**“throw” and “throws” keyword.**

**Also Write a java that differentiate checked and unchecked exceptions.**

**(Mention at least two checked and two unchecked exception in program).**

**SOURCE CODE:**

public class Practical\_5\_3 {

*int* create\_exception() throws ArithmeticException {

*int* x = 10 / 0;

        return x;

    }

    public *void* checkAge(*int* *age*) {

        if (*age* < 18)

            throw new ArithmeticException("Not Eligible to Vote.");

        else

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

    }

    public static *void* main(String[] *args*) {

        Practical\_5\_3 obj = new Practical\_5\_3();

        System.out.println("20DCE019-Yatharth Chauhan\n");

        try {

            obj.create\_exception();

        } catch (ArithmeticException e) {

            System.out.println("An Error has occured.");

            System.out.println("\n20DCE019-Yatharth Chauhan");

        }

*int* age = 10;

        obj.checkAge(age);

    }

}

**OUTPUT:**

**Text

Description automatically generated**

**CONCLUSION:**

1. Here also we have used the concept of exception handling.

2. Here we have made the user defined exception in order to check whether age entered is eligible for voting or not.

**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 LineCounts file1.txt file2.txt**

**file3.txt". Write each file name, along with the number of lines in that file,**

**to standard output. If an error occurs while trying to read from one of the**

**files, you should print an error message for that file, but you should still**

**process all the remaining files.**

**SOURCE CODE:**

import java.io.\*;

import java.io.File;

import java.util.\*;

public class Practical\_6\_1 {

    public static *void* main(String[] *args*) {

*int* x;

        for (x = 0; x < *args*.length; x++) {

*int* y = 0;

            try {

                File fileobj = new File(*args*[x]);

                if (fileobj.exists()) {

                    System.out.println("Searching For " + *args*[x] + "......");

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

                }

                Scanner sc = new Scanner(fileobj);

                while (sc.hasNextLine()) {

                    y++;

                    sc.nextLine();

                }

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

                sc.close();

            } catch (IOException e) {

                System.out.println("An error Occured While Finding " + *args*[x] + "file");

            }

            System.out.println("20DCE019-Yatharth Chauhan\n");

        }

    }

}

**OUTPUT:**

**Text

Description automatically generated**

**CONCLUSION:** In this practical we learnt how to perform file handling along with the syntax of file handling in java.

**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.**

**SOURCE CODE:**

import java.io.BufferedInputStream;

import java.io.FileInputStream;

class Practical\_6\_2 {

    public static *void* main(String *args*[]) {

        try {

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

            BufferedInputStream bis = new BufferedInputStream(fis);

*int* i;

*int* occurs = 0;

            while ((i = bis.read()) != -1) {

*char* a = (*char*) i;

                if (a == *args*[0].charAt(0)) {

                    occurs++;

                }

            }

            System.out.println("Letter e is " + occurs + " of times repeated in the file.");

            bis.close();

        } catch (Exception ex) {

            System.out.println(ex.getMessage());

        }

        System.out.println("\n20DCE019-Yatharth Chauhan");

    }

}

**OUTPUT:**

**Text

Description automatically generated**

**CONCLUSION:** In this practical we learnt how to counts the number of times a particular character, such as e, appears in a file in java.

**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.**

**SOURCE CODE:**

import java.io.\*;

public class Practical\_6\_3 {

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

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

        FileReader fr = new FileReader(f);

        BufferedReader br = new BufferedReader(fr);

        String words[] = null;

        String s = "Yatharth";

        String i;

*boolean* occur = false;

        while ((i = br.readLine()) != null) {

            words = i.split(" ");

            for (String word : words) {

                if (word.equals(s)) {

                    occur = true;

                }

            }

        }

        if (occur) {

            System.out.println("The Given word is present in the file.");

        } else {

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

        }

        fr.close();

        System.out.println("\n20DCE019-Yatharth Chauhan");

    }

}

**OUTPUT:**

**Text

Description automatically generated**

**CONCLUSION:** In this practical we can use the file handling concept to read the file and find out the required portion, line, word with help of bufferReader and Wrapper class.

**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.**

**SOURCE CODE:**

import java.io.\*;

public class Practical\_6\_4 {

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

        try {

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

            FileOutputStream fos = new FileOutputStream("file2.txt");

*int* x;

            while ((x = fis.read()) != -1) {

                fos.write(x);

            }

            System.out.println("Data is copied Successfully");

            fis.close();

            fos.close();

        } catch (Exception e) {

            System.out.println(e);

        }

        System.out.println("\n20DCE019-Yatharth Chauhan");

    }

}

**OUTPUT:**

**Graphical user interface, text

Description automatically generated**

**CONCLUSION:** In this practical we learnt how to copy data from one file into another file in java.

**PRACTICAL – 6(5-A)**

**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.**

**SOURCE CODE:**

import java.io.\*;

class Practical\_6\_5\_A {

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

        FileReader fr = new FileReader("file1.txt");

        BufferedReader br = new BufferedReader(fr);

*int* i;

        while ((i = br.read()) != -1)

        {

            System.out.print((*char*) i);

        }

        br.close();

        fr.close();

        System.out.println("\n\n20DCE019-Yatharth Chauhan");

    }

}

**OUTPUT:**

**Graphical user interface, text

Description automatically generated**

**PRACTICAL – 6(5-B)**

**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.**

**SOURCE CODE:**

import java.io.\*;

class Practical\_6\_5\_B {

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

        FileWriter writer = new FileWriter("file1.txt");

        BufferedWriter buffer = new BufferedWriter(writer);

        buffer.write("I'm Yatharth Chauhan");

        buffer.close();

        System.out.println("Successfull");

        System.out.println("\n20DCE019-Yatharth Chauhan");

    }

}

**OUTPUT:**

**Graphical user interface, text, application

Description automatically generated**

**CONCLUSION:** In this practical we learnt how to use of character and byte stream using BufferedReader/BufferedWriter in java.

**PRACTICAL – 7(1-A)**

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

**A. by extending Thread class**

**B. by using Runnable interface.**

**SOURCE CODE-A:**

public class Practical\_7\_1\_A extends Thread {

    public *void* run() {

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

        System.out.println("\n20DCE019-Yatharth Chauhan");

    }

    public static *void* main(String[] *args*) {

        Practical\_7\_1\_A obj = new Practical\_7\_1\_A();

        obj.start();

    }

}

**SOURCE CODE-B:**

public class Practical\_7\_1\_B implements Runnable {

    public *void* run() {

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

        System.out.println("\n20DCE019-Yatharth Chauhan");

    }

    public static *void* main(String[] *args*) {

        Practical\_7\_1\_B ob = new Practical\_7\_1\_B();

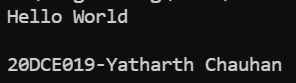
        Thread th = new Thread(ob);

        th.start();

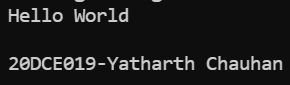
    }

}

**OUTPUT-A:**

****

**OUTPUT-B:**

****

**CONCLUSION:** In this practical we learnt concept of multithreading in 2ways i.e. first by extending super class Thread and another by using Runnable Interface to get the threads run and we start the Thread by start() method which implicitly calls run() method.

**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.**

**SOURCE CODE:**

import java.util.\*;

class Mythread implements Runnable {

*int* n1;

*int* t1;

*int* c = 0;

*int*[] th1 = new *int*[200];

    Mythread(*int* *n*, *int* *t*, *int*[] *th*) {

        this.n1 = *n*;

        this.th1 = *th*;

        this.t1 = *t*;

    }

    public *void* run() {

        for (*int* i = 1; i <= t1; i++) {

            c = c + th1[i];

            System.out.println(th1[i]);

        }

        try {

            Thread.sleep(500);

        } catch (InterruptedException e) {

            System.out.println("error occured:" + e);

        }

    }

}

public class Practical\_7\_2 {

    public static *void* main(String[] *args*) {

        Scanner sc = new Scanner(System.in);

        System.out.println("Enter your number :");

*int* n = sc.nextInt();

        System.out.println("Enter your threads number:");

*int* t = sc.nextInt();

*int*[] th = new *int*[200];

        th[1] = n / t;

*int* sum = 0;

        for (*int* i = 1; i < t; i++) {

            th[i] = th[1];

            sum = sum + th[i];

        }

        th[t] = n - sum;

        System.out.println("The number is: ");

        Mythread b = new Mythread(n, t, th);

        Thread a = new Thread(b);

        a.start();

        try {

            a.join();

        } catch (Exception e) {

            System.out.println("Error occured :" + e);

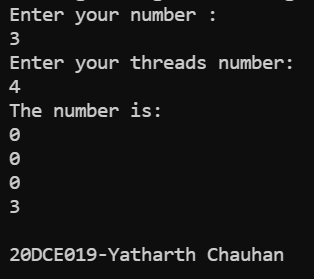
        }

        System.out.println("\n20DCE019-Yatharth Chauhan");

    }

}

**OUTPUT:**

****

**CONCLUSION:** In this practical we creating the number of threads required by user as well as dividing the numbers to each thread on basis of input by user.

**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.**

**SOURCE CODE:**

import java.util.Random;

import java.lang.Thread;

public class Practical\_7\_3 {

    public static *void* main(String[] *args*) {

        RandomNum x = new RandomNum();

        x.start();

        System.out.println("20DCE019-Yatharth Chauhan\n");

    }

}

class RandomNum extends Thread {

    Random rand = new Random();

*int* y = rand.nextInt(100);

    public *void* run() {

        if (y % 2 == 0) {

            Square s = new Square();

            s.start();

        } else {

            Cube c = new Cube();

            c.start();

        }

    }

*int* getRandom() {

        return y;

    }

}

class Square extends Thread {

    public *void* run() {

        try {

            RandomNum x = new RandomNum();

*int* y = x.getRandom();

            System.out.println("Square Of " + y + " is :" + (y \* y));

        } catch (Exception e) {

            System.out.println("An Error Has Occured");

            e.printStackTrace();

        }

    }

}

class Cube extends Thread {

    public *void* run() {

        try {

            RandomNum x = new RandomNum();

*int* y = x.getRandom();

            System.out.println("Cube Of " + y + " is :" + (y \* y \* y));

        } catch (Exception e) {

            System.out.println("An Error Has Occured");

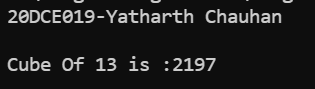
            e.printStackTrace();

        }

    }

}

**OUTPUT:**

****

**CONCLUSION:** In this program we learnt how to generate random number using random method from the math class and how to use function sleep from thread class.

**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.**

**SOURCE CODE:**

import java.util.logging.Level;

import java.util.logging.Logger;

import java.util.Scanner;

public class Practical\_7\_4 extends Thread {

    public *void* run() {

*int* x;

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

        Scanner sc = new Scanner(System.in);

        x = sc.nextInt();

        sc.close();

        x++;

        try {

            Thread.sleep(1000);

        } catch (InterruptedException ex) {

            Logger.getLogger(Practical\_7\_4.class.getName()).log(Level.SEVERE, null, ex);

        }

        System.out.println("Value of variableis : " + x);

        System.out.println("\n20DCE019-Yatharth Chauhan");

    }

    public static *void* main(String[] *args*) {

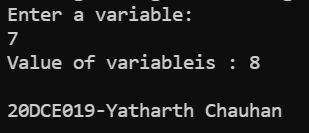
        Practical\_7\_4 t = new Practical\_7\_4();

        t.start();

    }

}

**OUTPUT:**

****

**CONCLUSION:** In this practical we can use the sleep function to stop or say pause the thread for some specific time to execute another thread in the pause time of first thread.

**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.**

**SOURCE CODE:**

public class Practical\_7\_5 extends Thread {

    public static *void* main(String[] *args*) {

        Practical\_7\_5 t1 = new Practical\_7\_5();

        Practical\_7\_5 t2 = new Practical\_7\_5();

        Practical\_7\_5 t3 = new Practical\_7\_5();

        System.out.println("Current Priority of Threads: ");

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

        System.out.println("Priority of Thread 1: " + t2.getPriority());

        System.out.println("Priority of Thread 1: " + t3.getPriority());

        System.out.println();

        t1.setPriority(3);

        t2.setPriority(5);

        t3.setPriority(7);

        System.out.println("Priorities of Threads After Setting New Priorities: ");

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

        System.out.println("Priority of Thread 1: " + t2.getPriority());

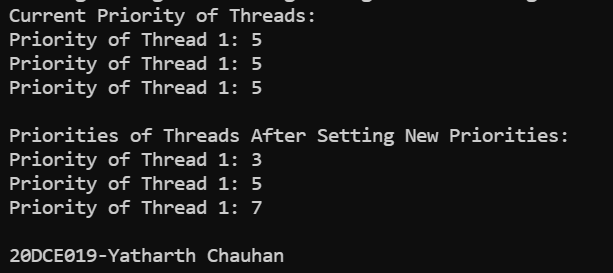
        System.out.println("Priority of Thread 1: " + t3.getPriority());

        System.out.println("\n20DCE019-Yatharth Chauhan");

    }

}

**OUTPUT:**

****

**CONCLUSION:** In this practical we can use 2 methods to get name and priority of our thread and another 2 methods to set the name and priority of the thread apart from the default name and priorities given by system.

**PRACTICAL – 7(6)**

**Aim: Write a program to solve producer-consumer problem using**

**thread synchronization.**

**SOURCE CODE:**

public class Practical\_7\_6 {

    public static *void* main(String[] *args*) {

        Producer p = new Producer();

        Consumer c = new Consumer(p);

        Thread t1 = new Thread(p);

        Thread t2 = new Thread(c);

        t2.start();

        t1.start();

    }

}

class Producer extends Thread {

    StringBuffer buffer;

    Producer() {

        buffer = new StringBuffer(4);

    }

    public *void* run() {

        synchronized (buffer) {

            try {

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

                    System.out.println("Process " + i + " is generated");

                    buffer.append(i);

                }

            } catch (Exception e) {

                System.out.println(e);

            }

            System.out.println("\nBuffer is Full");

            buffer.notify();

        }

    }

}

class Consumer extends Thread {

    Producer p;

    Consumer(Producer *temp*) {

        p = *temp*;

    }

    public *void* run() {

        synchronized (p.buffer) {

            try {

                p.buffer.wait();

            } catch (Exception e) {

                System.out.println(e);

            }

            System.out.println("\nFor Consumer side:");

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

                System.out.println(p.buffer.charAt(i) + " Process is Consumed");

            }

            System.out.println("\nBuffer is empty.");

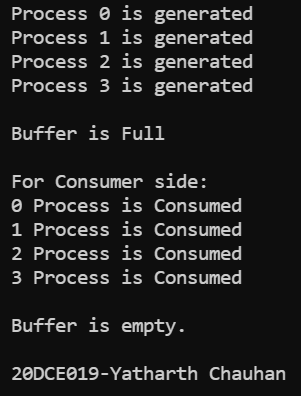
        }

        System.out.println("\n20DCE019-Yatharth Chauhan");

    }

}

**OUTPUT:**

****

**CONCLUSION:** In this practical we can solve the producer and consumer problem by using the java multithreading concept.

**PRACTICAL – 8(1)**

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

****

**SOURCE CODE:**

import java.util.\*;

public class Practical\_8\_1 {

    public static *void* main(String[] *args*) {

        Integer arr[] = new Integer[] { 1, 2, 3, 4 };

        MyStack s = new MyStack(arr);

        System.out.println("Current top= " + s.peek());

        System.out.println("Pushing 7,8,9 in stack....");

        s.push(7);

        s.push(8);

        s.push(9);

        System.out.println("Elements in the stack are: ");

        while (!s.isEmpty()) {

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

        }

        System.out.println("\n20DCE019-Yatharth Chauhan");

    }

}

class MyStack {

    ArrayList<Object> list;

    MyStack(Object *elements*[]) {

        list = new ArrayList<Object>();

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

            list.add(*elements*[i]);

        }

    }

*boolean* isEmpty() {

        return (list.size() == 0);

    }

    Object peek() {

        return list.get(list.size() - 1);

    }

    Object pop() {

        Object ob = list.get(list.size() - 1);

        list.remove(list.size() - 1);

        return ob;

    }

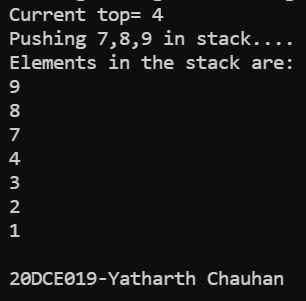
*void* push(Object *o*) {

        list.add(*o*);

    }

}

**OUTPUT:**

****

**CONCLUSION:** In this practical we learnt about different types of methods or functions of Array list

**PRACTICAL – 8(2)**

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

**SOURCE CODE:**

import java.util.\*;

public class Practical\_8\_2 {

    public static *void* main(String[] *args*) {

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

        list.add(10);

        list.add(300);

        list.add(45);

        list.add(2);

        list.add(5);

        System.out.println("BEFORE SORTING:-" + list);

        Collections.sort(list);

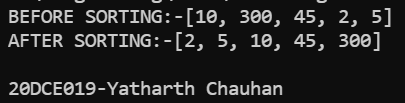
        System.out.println("AFTER SORTING:-" + list);

        System.out.println("\n20DCE019-Yatharth Chauhan");

    }

}

**OUTPUT:**

****

**CONCLUSION:** In this practical we leant about sorting of Array List.

**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.**

**SOURCE CODE:**

import java.util.\*;

public class Practical\_8\_3 {

    public static *void* main(String[] *args*) {

        Scanner sc = new Scanner(System.in);

        System.out.println("Enter the text: ");

        String s = sc.nextLine();

        s = s.toLowerCase();

        StringTokenizer st = new StringTokenizer(s, ".,;/()! ");

        TreeSet<String> words = new TreeSet<String>();

        sc.close();

        while (st.hasMoreTokens()) {

            words.add(st.nextToken());

        }

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

        Iterator<String> i = words.iterator();

        while (i.hasNext()) {

            String word = i.next();

*int* count = 0;

            st = new StringTokenizer(s, ".,;/()! ");

            while (st.hasMoreTokens()) {

                String swe = st.nextToken();

                if (word.equals(swe))

                    count++;

            }

            map.put(word, count);

        }

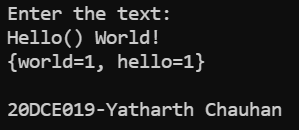
        System.out.println(map);

        System.out.println("\n20DCE019-Yatharth Chauhan");

    }

}

**OUTPUT:**

****

**CONCLUSION:** In this practical we learnt about Map and Set classes.

**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.**

**SOURCE CODE:**

import java.util.\*;

import java.io.\*;

class Practical\_8\_4 {

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

        Scanner sc = new Scanner(System.in);

        System.out.println("Enter the file name you want to scan: ");

        String f = sc.nextLine();

        File file = new File(f);

        FileReader br = new FileReader(file);

        BufferedReader fr = new BufferedReader(br);

        sc.close();

        String keywords[] = new String[] { "abstract", "assert ", "boolean", "break", "byte", "case", "catch", "char",

                "class", "continue", "default", "do", "double", "else", "enum ", "extends", "final", "finally", "float",

                "for", "if", "implements", "import", "instanceof", "int", "interface", "long", "native", "new",

                "package", "private", "protected", "public", "return", "short", "static", "strictfp", "super", "switch",

                "synchronized", "this", "throw", "throws", "transient", "try", "void", "volatile", "while" };

        HashSet<String> set = new HashSet<String>();

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

            set.add(keywords[i]);

        }

        String st;

*int* count = 0;

        while ((st = fr.readLine()) != null) {

            StringTokenizer str = new StringTokenizer(st, " +-/\*%<>;:=&|!~()");

            while (str.hasMoreTokens()) {

                String swre = str.nextToken();

                if (set.contains(swre)) {

                    count++;

                }

            }

        }

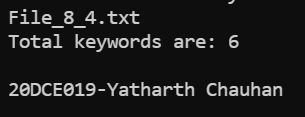
        System.out.println("Total keywords are: " + count);

        System.out.println("\n20DCE019-Yatharth Chauhan");

    }

}

**OUTPUT:**

****

**CONCLUSION:** in this Practical We Learned How to use Hash set to store keywords & find it in a file.