|  |  |
| --- | --- |
| FOCP II | |
| Lab Manual | |
| **Department of Computer Science and Engineering**  **The NorthCap University, Gurugram** | |
|  |

|  |  |
| --- | --- |
|  | |
|  | |
|  | |
|  |

**FOCP II**

**Lab Manual**

**CSL 108**

**Dr. Neeti Kashyap**



Department of Computer Science and Engineering

NorthCap University, Gurugram- 122001, India

Session 2021-22

*Published by:*

**School of Engineering and Technology**

**Department of Computer Science & Engineering**

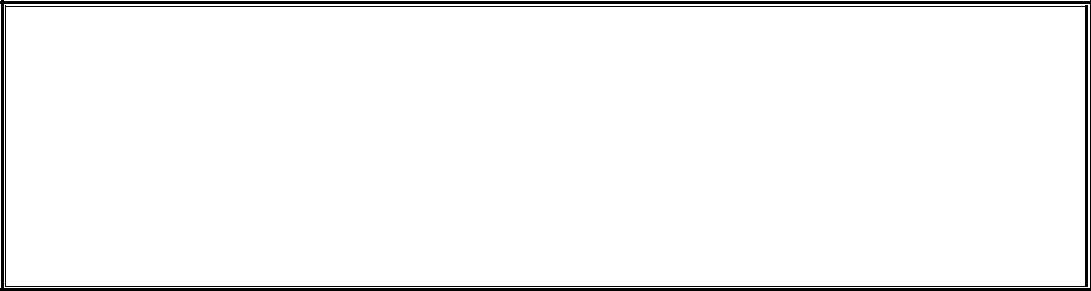
**The NorthCap University Gurugram**

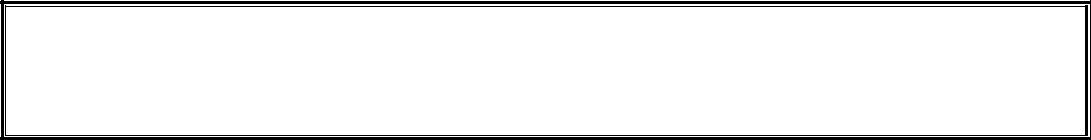
**• Laboratory Manual is for Internal Circulation only**

**© Copyright Reserved**

*No part of this Practical Record Book may be*

*reproduced, used, stored without prior permission of The NorthCap University*



Copying or facilitating copying of lab work comes under cheating and is considered as use of unfair means. Students indulging in copying or facilitating copying shall be awarded zero marks for that particular experiment. Frequent cases of copying may lead to disciplinary action. Attendance in lab classes is mandatory.

Labs are open up to 7 PM upon request. Students are encouraged to make full use of labs beyond normal lab hours.

**PREFACE**

The aim of this lab manual is to help students understand real life problem using programming skills.

This manual is required for second semester computer science and engineering students, so that they are able understand Java, one of the most in-demand programming languages. The lab manual outline is designed in such a manner that the beginners with little or no knowledge about Object Oriented programming concepts can understand the core OOP concepts including Encapsulation, Polymorphism, Inheritance etc. and their implementation in Java. The students will have extensive hands-on experience writing, compiling, testing and executing Java programs applying the above principles for developing modular reusable programs. to understand fundamental concepts of Java programming language that can be further used to design applications.

By the end of this practical, the students will gain the foundational skills a software engineer needs, to solve real-world problems, from designing algorithms to testing and debugging; and will be able to apply these concepts to build their own interactive Java applications.

Author expresses deep gratitude to Members, Governing Body-NCU for encouragement and motivation.

**Dr. Neeti Kashyap**

**The NorthCap University**

**Gurugram, India**

|  |  |  |  |
| --- | --- | --- | --- |
| **CONTENTS** | | | |
| S.N. | **Details** | Page No. | |
| 1 | **Introduction** | 6 | |
| 2 | **Lab Requirement** | 6 |
| 3 | **General Instructions** | 6 |
| 4 | **List of Experiments** | 8 |
| 5 | **Rubrics** | 9 |
| 6 | **Annexure 1 (Format of Lab Report)** | 10 |

1. **INTRODUCTION**

That ‘learning is a continuous process’ cannot be over emphasized. The theoretical knowledge

gained during lecture sessions need to be strengthened through practical experimentation.

Thus, practical makes an integral part of a learning process.

The purpose of conducting experiments can be stated as follows:

* To familiarize the students with the basic concepts, programming skill development and the take home laboratory assignments mainly implementation-oriented which have to be coded in high level language. The lab sessions will be based on exploring the concepts discussed in class.
* Observing basic structure and characteristics of Computer Systems
* Reporting and analyzing the programming concepts.
* Hands on experience on the coding and software tools

1. **LAB REQUIREMENTS**

* Mac or PC
* Java Development kit (JDK)
* Free Text editor (Sublime Recommended)
* Web Browser (Chrome and Firefox Recommended)
* GIT user Account

1. **GENERAL INSTRUCTIONS**

**3.1 General discipline in the lab**

* Students must turn up in time and contact concerned faculty for the experiment they are supposed to perform.
* Students will not be allowed to enter late in the lab.
* Students will not leave the class till the period is over.
* Students should come prepared for their experiment.
* Experimental results should be entered in the lab report format and certified/signed by concerned faculty/ lab Instructor.
* Students must get the connection of the hardware setup verified before switching on the power supply.
* Students should maintain silence while performing the experiments. If any necessity arises for discussion amongst them, they should discuss with a very low pitch without disturbing the adjacent groups.
* Violating the above code of conduct may attract disciplinary action.
* Damaging lab equipment or removing any component from the lab may invite penalties and strict disciplinary action.
  1. **Attendance**
* Attendance in the lab class is compulsory.
* Students should not attend a different lab group/section other than the one assigned at the beginning of the session.
* On account of illness or some family problems, if a student misses his/her lab classes, he/she may be assigned a different group to make up the losses in consultation with the concerned faculty / lab instructor. Or he/she may work in the lab during spare/extra hours to complete the experiment. No attendance will be granted for such case**.**

**3.3 Preparation and Performance**

* Students should come to the lab thoroughly prepared on the practicals they are assigned to perform on that day. Brief introduction to each experiment with information about self-study reference is provided on LMS.
* Students must bring the lab report during each practical class with written records of the last experiments performed complete in all respect.
* Each student is required to write a complete report of the practical he has performed and bring to lab class for evaluation in the next working lab. Sufficient space in work book is provided for independent writing of theory, observation, calculation and conclusion.
* Students should follow the Zero tolerance policy for copying / plagiarism. Zero marks will be awarded if found copied. If caught further, it will lead to disciplinary action.
* Refer **Annexure 1** for Lab Report Format
  1. **Norms to be followed for doing lab practical**
* All the students will work in a team of 2 members each.
* Each and every practical will contain a number of programming problems to be solved by students.

1. **LIST OF EXPERIMENTS**

|  |  |
| --- | --- |
| Exp. No. | List of Experiments |
| 1 | Programs on Data types |
| 2 | Programs on Control Statements |
| 3 | Programs on Arrays |
| 4 | Programs on classes and objects |
| 5 | Programs on Inheritance |
| 6 | Programs on Interface |
| 7 | Programs on Packages |
| 8 | Programs on Exception Handling |
| 9 | Programs on File Handling |

**5. RUBRICS**

|  |  |
| --- | --- |
| **Marks Distribution** | |
| **Continuous Evaluation (50 Marks)** | **End Semester Project (20 Marks)** |
| Each experiment shall be evaluated for 10 marks and at the end of the semester proportional marks shall be awarded out of 50. | Unguided project carries 20 marks. |

**6. Annexure 1**

**FOCP II**

**CSL108**

Project Report



Faculty name: Ruchika Lalit

Student name: Piyush Gambhir

Roll No.: 21CSU349

Semester: Semester-II

Group: CSE-F (CS-12)

**Department of Computer Science and Engineering**

**The NorthCap University, Gurugram- 122001, India**

**Session 2021-22**

**Table of Contents**

|  |  |  |
| --- | --- | --- |
| **S.No** |  | **Page No.** |
| **1.** | **Project Title:** |  |
| **2.** | **Description of Project:**  **Problem Statement** |  |
| **3.** | **Problem Analysis**  **3.1 Hardware Requirements**  **3.2 Software Requirements** |  |
| **4.** | **Design**  **4.1 Data/Input Output Description:**  **4.2 Algorithmic Approach / Algorithm / DFD / ER diagram/Program Steps** |  |
| **5.** | **Implementation and Testing (stage/module wise)** |  |
| **6.** | **Output (Screenshots)** |  |
| **7.** | **Conclusion and Future Scope** |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **S. S No** | **Experiment** | **Date of Experiment** | **Date of Submission** | **Marks** | **CO Covered** | **Signature** |
| **1** | Programs on Data Types |  |  |  |  |  |
| **2** | Programs on Control Statements |  |  |  |  |  |
| **3** | Programs on Arrays |  |  |  |  |  |
| **4** | Programs on Classes & Objects |  |  |  |  |  |
| **5** | Programs on Inheritance |  |  |  |  |  |
| **6** | Programs on Packages |  |  |  |  |  |
| **7** | Programs on Interfaces |  |  |  |  |  |
| **8** | Programs on Exception Handling |  |  |  |  |  |
| **9** | Programs on File handling |  |  |  |  |  |

**Index**

**PRACTICAL NO. 1**

|  |
| --- |
| **Student Name and Roll Number:** |
| **Semester /Section:** |
| **Date:** |
| **Faculty Signature:** |

|  |
| --- |
| **Objective(s):**   * Perform variable assignment. * Use comments in code * Use operator precedence and operator associativity. * Effectively use arithmetic expressions in Java |
| **Outcome:**  Student will be familiarizing with the data types in Java. |
| **Problem Statement:**  Q1. Write a program that takes two integers (values to be given within the program) and displays the output of the following operators: addition, subtraction, multiplication, division and modulus.  Definition of Done:  DoD 1: Assign two numbers to two variables.  DoD2: Use separate variables to store the results.  Q2. Write a Java program to convert minutes into the number of years, months and days.  Definition Of Done :  DoD 1: Ask the user to input the minutes  DoD 2: Display years and months and days in the sequence yy-mm-dd. |
| **Background Study:** Data Types in Java  Primitive data types: The primitive data types include boolean, char, byte, short, int, long, float and double.  Non-primitive data types: The non-primitive data types include Classes, Interfaces, and Arrays. |
| **Question Bank:**  1. Why Java is considered dynamic?  2. What is Java Virtual Machine and how it is considered in context of Java’s platform independent feature?  3. List two Java IDE’s? List some Java keywords(unlike C, C++ keywords)?  4. Consider the following class:  public class IdentifyMyParts {  public static int x = 7;  public int y = 3;  }  a) What are the class variables?  b) What are the instance variables? |
| **Flipped practicals**  1. What is the output from the following code:  IdentifyMyParts a = new IdentifyMyParts();  IdentifyMyParts b = new IdentifyMyParts();  a.y = 5;  b.y = 6;  a.x = 1;  b.x = 2;  System.out.println("a.y = " + a.y);  System.out.println("b.y = " + b.y);  System.out.println("a.x = " + a.x);  System.out.println("b.x = " + b.x);  System.out.println("IdentifyMyParts.x = " + IdentifyMyParts.x);  **2.** What's wrong with the following program?  public class SomethingIsWrong {  public static void main(String[] args) {  Rectangle myRect;  myRect.width = 40;  myRect.height = 50;  System.out.println("myRect's area is " + myRect.area());  }  } |

**Student Work Area**

**Algorithm/Flowchart/Code/Sample Outputs**

**Problem Statement - 1**

/\*

Write a program that takes two integers (values to be given within the program) and displays the output of the following operators: addition, subtraction, multiplication, division and modulus.

Definition of Done:

DoD 1: Assign two numbers to two variables.

DoD2: Use separate variables to store the results.

\*/

import java.util.Scanner;

/\*\*

 \* practical\_1\_problem\_statement\_1

 \*/

public class practical\_1\_problem\_statement\_1 {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        System.out.print("\nEnter Number 1: ");

        int a = sc.nextInt();

        System.out.print("Enter Number 2: ");

        int b = sc.nextInt();

        int sum = a + b;

        int sub = a - b;

        int mul = a \* b;

        int div = a / b;

        int mod = a % b;

        System.out.println("\n\n");

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

        System.out.println("Subttraction     :   " + a + " - " + b + " = " + sub);

        System.out.println("Multiplication   :   " + a + " x " + b + " = " + mul);

        System.out.println("Division         :   " + a + " / " + b + " = " + div);

        System.out.println("Modulus          :   " + a + " % " + b + " = " + mod);

        System.out.println("\n\n");

    }

}

**Output:**Graphical user interface, text, email

Description automatically generated

**Problem Statement - 2**

/\*

Write a Java program to convert minutes into the number of years, months and days.

Definition Of Done :

DoD 1: Ask the user to input the minutes

DoD 2: Display years and months and days in the sequence yy-mm-dd.

\*/

import java.util.Scanner;

/\*\*

 \* practical\_1\_problem\_statement\_2

 \*/

public class practical\_1\_problem\_statement\_2 {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        System.out.print("\n\nEnter Number of Minutes: ");

        int minutes = sc.nextInt();

        int years = minutes / (60 \* 24 \* 365);

        int r1 = minutes % (60 \* 24 \* 365);

        int months = r1 / (60 \* 24 \* 30);

        int r2 = r1 % (60 \* 24 \* 30);

        int days = r2 / (60 \* 24);

        int r3 = r2 % (60 \* 24);

        int hours = r3 / (60);

        System.out.println("\nEntered Minutes in the Format hh--dd-mm-yy: " + hours + "--" + days + "--" + months + "--" + years);

    }

}

**Output:**

Graphical user interface, text, application, email

Description automatically generated

**Question Bank**

**A.1**

Java is considered dynamic because of Bytecode. The source code which is written in one platform that code can be executed in any platform.

**A.2**

Java Virtual Machine (JVM) is a specification that provides runtime environment in which java bytecode (. class files) can be executed.

**A.3**

Two IDEs of Java are:

IntelliJ Idea

NetBeans

Some Java Keywords are public, static, etc.

**A.4**

1. x
2. y

**Flipped Practical’s**

**A.1**

If x and y are static variables.

Output:

a.y = 6

b.y = 6

a.x = 2

b.x = 2

IdentifyMyParts.x = 2

**A.2**

At line 3,

Object not initiated correctly.

**Correct Code:** Rectangle myRect = new Ractangel();

**PRACTICAL NO. 2**

|  |
| --- |
| **Student Name and Roll Number:** |
| **Semester /Section:** |
| **Date:** |
| **Faculty Signature:** |

|  |
| --- |
| **Objective**  To familiarize the students with control statements in JAVA |
| **Program Outcome**  Through this practical, students will learn about the control statements |
| **Problem Statement**   1. Write a Java program to solve quadratic equations (use if, else if and else).   *Definition of Done:*  DoD 1: The program asks the values of coefficients of a quadratic equation.  DoD 2: The program should display the roots or an appropriate message.   1. Write a Java program that asks the user to provide a single character from the alphabet. Print Vowel or Consonant, depending on the user input. If the user input is not a letter (between a and z or A and Z), or is a string of length > 1, print an error message.   *Definition of Done:*  DoD 1: The program asks an input from the user.  DoD 2: A single character input is taken from the user or an error message is  generated.  DoD 3: The program should print “Vowel” if the entered character is a vowel and  “Consonant” if the entered character is a consonant.   1. Write a Java program to print following structure:   \*  \*\*  \*\*\*  \*\*\*\*  \*\*\*\*\*  \*\*\*\*\*\* |
| **Background Study:**  Java compiler executes the java code from top to bottom. The statements are executed according to the order in which they appear. However, [Java](https://www.javatpoint.com/java-tutorial) provides statements that can be used to control the flow of java code. Such statements are called control flow statements.  Java provides three types of control flow statements.   * Decision Making statements * Loop statements * Jump statements |
| **Question Bank**  1. The most basic control flow statement supported by the Java programming language is the \_\_\_ statement.  2. The \_\_\_ statement allows for any number of possible execution paths.  3. The \_\_\_ statement is similar to the while statement, but evaluates its expression at the \_\_\_ of the loop.  4. How do you write an infinite loop using the for statement?  5. How do you write an infinite loop using the while statement?  6. Which looping process checks the test condition at the end of the loop?  7. Why do we use continue statement?  8. What is the size of boolean variable?  9. Which looping process is best used when the number of iterations is known? |
| **Flipped Practicals**  1. Consider the following code snippet.  if (aNumber >= 0)  if (aNumber == 0)  System.out.println("first string");  else System.out.println("second string");  System.out.println("third string");  a) What output do you think the code will produce if aNumber is 3?  b) Write a test program containing the previous code snippet; make aNumber 3. What is the output of the program? Is it what you predicted? Explain why the output is what it is; in other words, what is the control flow for the code snippet?  c) Using only spaces and line breaks, reformat the code snippet to make the control flow easier to understand.  d) Use braces, { and }, to further clarify the code.  2. What's wrong? for (int k = 2, k <= 12, k++)  3. If there is more than one statement in the block of a for loop, what must be placed at the beginning and the ending of the loop block?  4. What value is stored in num at the end of this looping?  for (num = 1; num <= 5; num++) |

**Student Work Area**

**Algorithm/Flowchart/Code/Sample Outputs.**

**Problem Statement – 1**

/\*

Write a Java program to solve quadratic equations (use if, else if and else).

Definition of Done:

DoD 1: The program asks the values of coefficients of a quadratic equation.

DoD 2: The program should display the roots or an appropriate message.

\*/

import java.util.Scanner;

/\*\*

 \* practical\_2\_problem\_statement\_1

 \*/

public class practical\_2\_problem\_statement\_1 {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        System.out.println("\nEnter the Values of Coefficients of the Quadratic Equation: ax^2 + bx + c = 0");

        System.out.print("\nEnter tha Value of a: ");

        int a = sc.nextInt();

        System.out.print("Enter tha Value of b: ");

        int b = sc.nextInt();

        System.out.print("Enter tha Value of c: ");

        int c = sc.nextInt();

        System.out.println("\nThe Quaratic Equation : " + a + "x^2 + " + b + "x + " + c + " = 0");

        int D = b \* b - 4 \* a \* c;

        if (D > 0) {

            System.out.println("\nThe Quadratic Equation Has Real Roots.");

        } else if (D < 0) {

            System.out.println("\nThe Quadratic Equation Has Imaginary Roots.");

        } else if (D == 0) {

            System.out.println("\nThe Quadratic Equation Has Real and Equal Roots.");

        }

    }

}

**Output:**

Graphical user interface, text, application, email

Description automatically generated

**Problem Statement - 2**

/\*

Write a Java program that asks the user to provide a single character from the alphabet. Print Vowel or Consonant, depending on the user input.

If the user input is not a letter (between a and z or A and Z), or is a string of length > 1, print an error message.

Definition of Done:

DoD 1: The program asks an input from the user.

DoD 2: A single character input is taken from the user or an error message is generated.

DoD 3: The program should print “Vowel” if the entered character is a vowel and

“Consonant” if the entered character is a consonant.

\*/

import java.util.Scanner;

/\*\*

 \* practical\_2\_problem\_statement\_2

 \*/

public class practical\_2\_problem\_statement\_2 {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        System.out.print("\nEnter a character: ");

        String character = sc.next();

        int ascii = character.charAt(0);

        if (character.length() == 1) {

            if (ascii >= 65 && ascii <= 90 || ascii >= 97 && ascii <= 122) {

                if (character.equalsIgnoreCase("a") ||

                        character.equalsIgnoreCase("e") ||

                        character.equalsIgnoreCase("i") ||

                        character.equalsIgnoreCase("o") ||

                        character.equalsIgnoreCase("u")) {

                    System.out.println("\nThe Entered Character is a Vowel.");

                } else {

                    System.out.println("\nThe Entered Character is a Consonant");

                }

            } else {

                System.out.println("Invalid Input! The Entered Value is not a Alapahbetic Character.");

            }

        } else {

            System.out.println("Invalid Input! You Have Entered More than One Character.");

        }

    }

}

**Output:**

Graphical user interface, text, application, email

Description automatically generated

**Problem Statement - 3**

/\*

Write a Java program to print following structure:

\*

\*\*

\*\*\*

\*\*\*\*

\*\*\*\*\*

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

\*/

import java.util.Scanner;

/\*\*

 \* practical\_2\_problem\_statement\_3

 \*/

public class practical\_2\_problem\_statement\_3 {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        // System.out.print("Enter the Number of Rows: ");

        // int n = sc.nextInt();

        int n = 6;

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

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

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

            }

            System.out.println();

        }

    }

}

**Output:**

Graphical user interface, text, application, email

Description automatically generated

**Question Bank**

**A.1**

If-else Statements

**A.2**

Switch Statement

**A.3**

do-while, bottom

**A.4**

for( ; ;){

//Execution Code

}

**A.5**

While(true){

//Execution Code

}

**A.6**

do-while loop

**A.7**

The continue keyword is used to end the current iteration in a for loop (or a while loop) and continues to the next iteration.

**A.8**

1 bit

**A.9**

for loop

**Flipped Practical’s**

**A.1**

1. Output:

second string

third string

1. Graphical user interface, text, application

   Description automatically generatedIn the above program, the value of aNumber =3. So, the else statement is executed which prints the string ‘second string’. And there is also a print statement outside the for loop which prints the string ‘third string’.
2. Program using only spaces and line breaks.Graphical user interface, text, application

   Description automatically generated
3. Text

   Description automatically generatedProgram using { and }

**A.2**

Commas are written instead of semicolon.

Correct: for(int k=2; k<=2; k++);

**A.3**

Curly Braces

**A.4**

6

|  |
| --- |
| **Student Name and Roll Number:** |
| **Semester /Section:** |
| **Date:** |
| **Faculty Signature:** |

**PRACTICAL NO. 3**

|  |
| --- |
| **Objective**  To familiarize the students with if-else loop. |
| **Program Outcome**  The students will learn the concept of looping in C. They will be able to understand the different types of statements encountered in C. |
| **Problem Statement**   1. While purchasing certain items, a discount of 10% is offered if the quantity purchased is more than 1000. If quantity and price per item are input through the keyboard, write a program to calculate the total expenses. 2. The marks obtained by a student in 5 different subjects are input through the keyboard. The student gets a division as per the following rules: Percentage above or equal to 60 - First division Percentage between 50 and 59 - Second division Percentage between 40 and 49 - Third division Percentage less than 40 - Fail Write a program to calculate the division obtained by the student. 3. Write a program to check whether a triangle is valid or not, when the three angles of the triangle are entered through the keyboard. A triangle is valid if the sum of all the three angles is equal to 180 degrees. 4. Write a program in C to read the age and display whether the candidate is eligible to vote or not.   Definition of Done:   * The program should ask the user to enter an integer. If it is floating, ask the user to enter appropriate number * The program should use if-else statement |

|  |
| --- |
| **Objective**  To familiarize the students with array in JAVA. |
| **Program Outcome**  The students will learn the concept of arrays in Java. |
| **Problem Statement**  1. Write a Java program to find the maximum and minimum value of an array.  Definition of Done:  DoD 1: The program should ask the user to enter the elements of the array.  DoD 2: The program should display the maximum and minimum elements of the array.  2. Write a Java program to find the index of an array element in an array of size 10. The program should not use any function other than main ( ) functions.  Definition of Done:  DoD 1: The program should ask the user to enter the elements of the array.  DoD 2: The program should ask the user to enter a number to search.  DoD 3: The program should display the elements of the array entered.  DoD 4: The program should display the index of the number if the item is present or display -1 of the elements is not present.    3. Write a Java Program to count even and odd numbers in an array.  Definition of Done  DoD 1: The program should ask the user to enter the elements of the array.  DoD 2: Even elements will be stored in EvenArray[] and odd elements will be stored in oddArray[].  DoD 3: Display all three arrays along with their length.  4. Write a Java program to read numbers in an integer array of size 5 and display the following (using functions for each functionality):  i) Sum of all the elements  ii) Sum of alternate elements in the array.  Definition of Done  DoD 1: The program should ask the user to enter the elements of the array.  DoD 2: The program should display a menu with the above choices and ask the user to choose one of the choices. |
| **Background Study:**  Java array is an object which contains elements of a similar data type. Additionally, The elements of an array are stored in a contiguous memory location. It is a data structure where we store similar elements. We can store only a fixed set of elements in a Java array.  Array in Java is index-based, the first element of the array is stored at the 0th index, 2nd element is stored on 1st index and so on. |
| **Question Bank:**  1. Can you pass the negative number as an array size?  2. Can you change the size of the array once you define it?  3. What is an anonymous array?  4. What is the difference between int[] a and int a[] ?  5. What are jagged arrays in java? Give example? |
| **Flipped Practicals**  1.Which of these is an incorrect array declaration? a) int arr[] = new int[5] b) int [] arr = new int[5] c) int arr[] = new int[5] d) int arr[] = int [5] new  2. What will be the output of the following program?  public class MyFirst {        public static void main(String[] args) {           MyFirst obj = new MyFirst(n);   }   static int a = 10;   static int n;   int b = 5;   int c;   public MyFirst(int m) {         System.out.println(a + ", " + b + ", " + c + ", " + n + ", " + m);     }  // Instance Block    {       b = 30;       n = 20;    }  // Static Block    static  {            a = 60;       }   } |

**Student Work Area**

**Algorithm/Flowchart/Code/Sample Outputs**

**Problem Statement – 1**

/\*

Write a Java program to find the maximum and minimum value of an array.

Definition of Done:

DoD 1:  The program should ask the user to enter the elements of the array.

DoD 2: The program should display the maximum and minimum elements of the array.

\*/

import java.util.Scanner;

/\*\*

 \* practical\_3\_problem\_statement\_1

 \*/

public class practical\_3\_problem\_statement\_1 {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        System.out.print("\nEnter the Number of Elemnts You Wish to Enter: ");

        int n = sc.nextInt();

        int[] arr = new int[n]; // Defining an Array of Size n

        for (int i = 0; i < n; i++) {

            System.out.print("Enter the Value of Element" + i + " of Array: ");

            arr[i] = sc.nextInt();

        }

        int min = arr[0];

        for (int i = 0; i < n; i++) {

            if (arr[0] > arr[i]) {

                min = arr[i];

            }

        }

        int max = arr[0];

        for (int i = 0; i < n; i++) {

            if (arr[0] < arr[i]) {

                max = arr[i];

            }

        }

        System.out.println("\nThe Maximum Element in the Array: " + max);

        System.out.println("The Minimum Element in the Array: " + min);

    }

}

**Output:**Text, email

Description automatically generated

**Problem Statement - 2**

/\*

Write a Java program to find the index of an array element in an array of size 10. The program should not use any function other than main ( ) functions.

Definition of Done:

DoD 1: The program should ask the user to enter the elements of the array.

DoD 2: The program should ask the user to enter a number to search.

DoD 3: The program should display the elements of the array entered.

DoD 4: The program should display the index of the number if the item is present or display -1 of the element is not present.

\*/

import java.util.Scanner;

/\*\*

 \* practical\_3\_problem\_statement\_2

 \*/

public class practical\_3\_problem\_statement\_2 {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        int n = 10;

        int[] arr = new int[n];

        System.out.println("\n\n");

        for (int i = 0; i < n; i++) {

            System.out.print("Enter the Value of Element " + (i + 1) + " of Array: ");

            arr[i] = sc.nextInt();

        }

        System.out.print("\nElements of the Array are: ");

        for (int i = 0; i < n; i++) {

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

        }

        System.out.print("\n\nEnter the Number You Want to Search: ");

        int num = sc.nextInt();

        for (int i = 0; i < n; i++) {

            if (arr[i] == num) {

                System.out.println("The Index of the Element Entered in the Array: " + i);

                System.out.println();

            }

        }

    }

}

**Output:**Text

Description automatically generated

**Problem Statement - 3**

/\*

Write a Java Program to count even and odd numbers in an array.

Definition of Done

DoD 1: The program should ask the user to enter the elements of the array.

DoD 2: Even elements will be stored in EvenArray[] and odd elements will be stored in oddArray[].

DoD 3: Display all three arrays along with their length.

\*/

import java.util.Scanner;

/\*\*

 \* practical\_3\_problem\_statement\_3

 \*/

public class practical\_3\_problem\_statement\_3 {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        System.out.print("\nEnter the Number of Elements You Want to Enter: ");

        int n = sc.nextInt();

        int[] arr = new int[n];

        System.out.println("\n");

        for (int i = 0; i < n; i++) {

            System.out.print("Enter the Value of Element " + (i + 1) + " of Array: ");

            arr[i] = sc.nextInt();

        }

        System.out.print("\nElements of the Array are: ");

        for (int i = 0; i < n; i++) {

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

        }

        System.out.print("\nLength of the Array is: " + arr.length);

        int count\_even = 0;

        int count\_odd = 0;

        for (int i = 0; i < n; i++) {

            if (arr[i] % 2 == 0) {

                count\_even++;

            } else {

                count\_odd++;

            }

        }

        int[] EvenArray = new int[count\_even];

        int[] OddArray = new int[count\_odd];

        int counter\_even = 0;

        int counter\_odd = 0;

        for (int i = 0; i < n; i++) {

            if (arr[i] % 2 == 0) {

                EvenArray[counter\_even] = arr[i];

                counter\_even++;

            } else {

                OddArray[counter\_odd] = arr[i];

                counter\_odd++;

            }

        }

        System.out.print("\n\nElements of the Even Array are: ");

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

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

        }

        System.out.print("\nLength of the Array is: " + EvenArray.length);

        System.out.print("\n\nElements of the Odd Array are: ");

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

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

        }

        System.out.print("\nLength of the Array is: " + OddArray.length);

    }

}

**Output:**

Text, letter

Description automatically generated

**Problem Statement – 4**

/\*

Write a Java program to read numbers in an integer array of size 5 and display the following (using functions for each functionality):

i) Sum of all the elements

ii) Sum of alternate elements in the array.

Definition of Done:

DoD 1: The program should ask the user to enter the elements of the array.

DoD 2: The program should display a menu with the above choices and ask the user to choose one of the choices.

\*/

import java.util.Scanner;

/\*\*

 \* practical\_3\_problem\_statement\_4

 \*/

public class practical\_3\_problem\_statement\_4 {

    static int Sum\_Array\_Elements(int arr[]) {

        int sum = 0;

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

            sum += arr[i];

        }

        return sum;

    }

    static int Sum\_Array\_Alternate\_Elements(int arr[]) {

        int sum = 0;

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

            if (i % 2 == 0) {

                sum += arr[i];

            }

        }

        return sum;

    }

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        int n = 5;

        int[] arr = new int[n];

        System.out.println("\n\n");

        for (int i = 0; i < n; i++) {

            System.out.print("Enter the Value of Element " + (i + 1) + " of Array: ");

            arr[i] = sc.nextInt();

        }

        System.out.print("\nElements of the Array are: ");

        for (int i = 0; i < n; i++) {

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

        }

        System.out.println("\n");

        while (true) {

            System.out.println("\n");

            System.out.println("""

                    Enter 1 To Print Sum of All Elements of Array.

                    Enter 2 To Print Sum of Alternate Elements of Array.

                    Enter 3 To Exit the Program.

                    """);

            System.out.print("Enter Your Choice: ");

            int user\_choice = sc.nextInt();

            if (user\_choice == 1) {

                System.out.println("The Sum of the Elements of The Array are: " + Sum\_Array\_Elements(arr));

            }

            else if(user\_choice == 2){

                System.out.println("The Sum of the Alternate Elemnts of the Array are: " + Sum\_Array\_Alternate\_Elements(arr));

            }

            else if(user\_choice == 3){

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

                break;

            }

            else{

                System.out.println("Invalid Input!");

                System.out.println("Try Again");

            }

        }

    }

}

**Output:**

Text

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

**Question Bank**

**A.1**

No, we cannot pass negative number as array size.

**A.2**

No, once the size of the array is defined it cannot be changed.

**A.3**

Anonymous array in java is an array with no name.

**A.4**

There is no difference between int[] a and int a[], both are used for the declaration of an arry.

**A.5**

Jagged array is a multidimensional array where member arrays are of different size.

Example:

1 2

3 6 4

2

**Flipped Practical’s**

**A.1**

d) int arr[] = int [5] new

**A.2**

60, 30, 0, 20, 0

**PRACTICAL NO: 4**

|  |
| --- |
| **Student Name and Roll Number:** |
| **Semester /Section:** |
| **Date:** |
| **Faculty Signature:** |

|  |
| --- |
| **Objective**  To familiarize the students with classes and objects. |
| **Program Outcome**  The students will learn the concept of classes and objects. |
| **Problem Statement**  1.Create a class named 'Student' with String variable 'name' and integer variable 'roll\_no'. Assign the value of roll\_no as '2' and that of name as "John" by creating an object of the class Student.  2. Write a program to print the area and perimeter of a triangle having sides of 3, 4 and 5 units by creating a class named 'Triangle' without any parameter in its constructor.  3. 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 the rectangle are entered through the keyboard.  4. 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 the user.  5. Write a program to calculate the distance between two points (x1, y1) and (x2, y2). All numbers and return values should be of type double.  Definition of Done:  DoD 1: Two java files to be defined. One for class definitions and another for the application  DoD 2: A class point is defined with two float variables for x1 and x2 and the following functionality:  i. Non-parametrized and parameterized constructors are defined.  ii. Get and set methods are defined for all the instance variables.  iii. Distance function is defined to calculate the distance between two points.  iv. Display function is defined with width of 7 and precision of 2.  Write this program with a static method definition for calculating the distance between two points. |
| **Background Study**  In object-oriented programming technique, we design a program using objects and classes. An object in Java is the physical as well as a logical entity, whereas, a class in Java is a logical entity only.  An object is an instance of a class. A class is a template or blueprint from which objects are created. So, an object is the instance(result) of a class. |
| **Question Bank**  1. What is the difference between class and object?  2. What is constructor chaining?  3. What is No-arg constructor?  4. What happens if you keep return type for a constructor?  5. What is the use of private constructor?  6. Can we use this() in a method?  7. Can we define a method with same name of class? |
| **Flipped Practicals**  **1.** What will be the output of the following Java program?  **class** A  {  **int** i;  **int** j;  A()  {  i = 1;  j = 2;  }  }  **class** Output  {  **public** **static** **void** main(String args[])  {  A obj1 = **new** A();  A obj2 = **new** A();  System.out.print(obj1.equals(obj2));  }  }  a) false b) true c) 1 d) Compilation Error |

**Student Work Area**

**Algorithm/Flowchart/Code/Sample Outputs**

**Problem Statement – 1**

/\*

Create a class named 'Student' with String variable 'name' and integer variable 'roll\_no'. Assign the value of roll\_no as '2' and that of name as "John" by creating an object of the class Student.

\*/

/\*\*

 \* practical\_4\_problem\_statement\_1

 \*/

class Student {

    String name;

    int roll\_no;

}

public class practical\_4\_problem\_statement\_1 {

    public static void main(String[] args) {

        Student john = new Student();

        john.name = "John";

        john.roll\_no = 2;

        System.out.println("\n\nName of the Student: " + john.name);

        System.out.println("Roll No of the Student: " + john.roll\_no);

    }

}

**Output:**

Graphical user interface, text, application, email

Description automatically generated

**Problem Statement - 2**

/\*

Write a program to print the area and perimeter of a triangle having sides of 3, 4 and 5 units by creating a class named 'Triangle' without any parameter in its constructor.

\*/

/\*\*

 \* practical\_4\_problem\_statement\_2

 \*/

import java.math.\*;

class Triangle {

    double a;

    double b;

    double c;

    public double area() {

        double s = ((a + b + c) / 2);

        return s \* Math.pow((s \* (s - a) \* (s - b) \* (s - c)), (1 / 2));

    }

    public double perimeter(){

        return a+b+c;

    }

}

public class practical\_4\_problem\_statement\_2 {

    public static void main(String[] args) {

        Triangle triangle = new Triangle();

        triangle.a = 5;

        triangle.b = 4;

        triangle.c = 3;

        System.out.println("Area of the Triangle = " + triangle.area());

        System.out.println("Perimeter of the Triangle = " + triangle.perimeter());

    }

}

**Output:**

Text

Description automatically generated

**Problem Statement – 3**

/\*

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 the rectangle are entered through the keyboard.

\*/

/\*\*

 \* practical\_4\_problem\_statement\_3

 \*/

import java.util.Scanner;

class Area {

    double length;

    double breadth;

    public Area(double l, double b) {

        length = l;

        breadth = b;

    }

    public double returnArea() {

        return length \* breadth;

    }

}

public class practical\_4\_problem\_statement\_3 {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        System.out.print("Enter Length of the Rectangle: ");

        double length = sc.nextDouble();

        System.out.print("Enter Breadth of the Rectangle: ");

        double breadth = sc.nextDouble();

        Area rectangle = new Area(length, breadth);

        System.out.println("\nArea of the Rectangle = " + rectangle.returnArea());

    }

}

**Output:**

Text, application

Description automatically generated

**Problem Statement – 4**

/\*

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 the user.

\*/

/\*\*

 \* practical\_4\_problem\_statement\_4

 \*/

import java.util.Scanner;

class Complex {

    int real;

    int imaginary;

    public Complex() {

    }

    public Complex(int r, int i) {

        real = r;

        imaginary = i;

    }

    public void PrintComplexNumber(int real, int imaginary) {

        System.out.print(real + " + " + imaginary + "i");

    }

    public void AddComplex(Complex c1, Complex c2) {

        Complex temp\_complex = new Complex();

        temp\_complex.real = c1.real + c2.real;

        temp\_complex.imaginary = c1.imaginary + c2.imaginary;

        System.out.print("\nSum of Complex Numbers: ");

        PrintComplexNumber(temp\_complex.real, temp\_complex.imaginary);

    }

    public void SubtractComplex(Complex c1, Complex c2) {

        Complex temp\_complex = new Complex();

        temp\_complex.real = c1.real - c2.real;

        temp\_complex.imaginary = c1.imaginary - c2.imaginary;

        System.out.print("\nDifference of Complex Numbers: ");

        PrintComplexNumber(temp\_complex.real, temp\_complex.imaginary);

    }

    public void MultiplyComplex(Complex c1, Complex c2) {

        Complex temp\_complex = new Complex();

        temp\_complex.real = (c1.real \* c2.real - c1.imaginary \* c2.imaginary);

        temp\_complex.imaginary = (c1.real \* c2.imaginary + c1.imaginary \* c2.real);

        System.out.print("\nProduct of Complex Numbers: ");

        PrintComplexNumber(temp\_complex.real, temp\_complex.imaginary);

    }

}

public class practical\_4\_problem\_statement\_4 {

    public static void main(String[] args) {

        Complex comp = new Complex();

        Scanner sc = new Scanner(System.in);

        System.out.println("\n\nEnter the Values For First Complex Number.");

        System.out.print("\nEnter Real Part of Complex Number: ");

        int rl1 = sc.nextInt();

        System.out.print("Enter Imaginary Part of Complex Number: ");

        int img1 = sc.nextInt();

        Complex c1 = new Complex(rl1, img1);

        System.out.print("\nComplex Number: ");

        comp.PrintComplexNumber(rl1, img1);

        System.out.println("\n\nEnter the Values For Second Complex Number.");

        System.out.print("\nEnter Real Part of Complex Number: ");

        int rl2 = sc.nextInt();

        System.out.print("Enter Imaginary Part of Complex Number: ");

        int img2 = sc.nextInt();

        Complex c2 = new Complex(rl2, img2);

        System.out.print("\nComplex Number: ");

        comp.PrintComplexNumber(rl2, img2);

        while (true) {

            System.out.println("\n");

            System.out.println("""

                    1. Enter 1 to Add Complex Numbers.

                    2. Enter 2 to Subtract Complex Numbers.

                    3. Enter 3 to Multiply Complex Numbers.

                    4. Enter 4 to Exit the Program.

                    """);

            System.out.print("\nEnter Your Choice: ");

            int user\_input = sc.nextInt();

            if (user\_input == 1) {

                comp.AddComplex(c1, c2);

            } else if (user\_input == 2) {

                comp.SubtractComplex(c1, c2);

            } else if (user\_input == 3) {

                comp.MultiplyComplex(c1, c2);

            } else if (user\_input == 4) {

                break;

            } else {

                System.out.println("Invalid Input!");

            }

        }

    }

}

**Output:**

Text

Description automatically generated

Text

Description automatically generated

**Problem Statement – 5**

/\*

Write a program to calculate the distance between two points (x1, y1) and (x2, y2). All numbers and return values should be of type double.

Definition of Done:

DoD 1: Two java files to be defined. One for class definitions and another for the application

DoD 2: A class point is defined with two float variables for x1 and x2 and the following functionality:

i. Non-parametrized and parameterized constructors are defined.

ii. Get and set methods are defined for all the instance variables.

iii. Distance function is defined to calculate the distance between two points.

iv. Display function is defined with width of 7 and precision of 2.

Write this program with a static method definition for calculating the distance between two points.

\*/

import java.lang.Math;

/\*\*

 \* practical\_4\_problem\_statement\_5

 \*/

class point {

    float x;

    float y;

    point() {

    }

    point(float temp\_x, float temp\_y) {

        this.x = temp\_x;

        this.y = temp\_y;

    }

    float getx() {

        return this.x;

    }

    float gety() {

        return y;

    }

    void setx(float temp\_x) {

        this.x = temp\_x;

    }

    void sety(float temp\_y) {

        this.y = temp\_y;

    }

    double Distance(point p) {

        double distance = Math.sqrt(Math.pow((this.x - p.x), 2) + Math.pow((this.y - p.y), 2));

        System.out.printf("\n\nThe Distance Between Two Points is: %2.7f", distance);

        return distance;

    }

}

public class practical\_4\_problem\_statement\_5 {

    public static void main(String[] args) {

        point p1 = new point(3, 5);

        point p2 = new point(2, 3);

        System.out.println("x1: " + p1.getx());

        System.out.print("y1: " + p1.gety());

        System.out.print("\n\n");

        System.out.println("x2: " + p2.getx());

        System.out.print("y2: " + p2.gety());

        p1.setx(2);

        p1.sety(9);

        p2.sety(4);

        p2.sety(6);

        System.out.print("\n\n");

        System.out.println("x1: " + p1.getx());

        System.out.print("y1: " + p1.gety());

        System.out.print("\n\n");

        System.out.println("x2: " + p2.getx());

        System.out.print("y2: " + p2.gety());

        p1.Distance(p2);

    }

}

**Output:**Graphical user interface, text, email

Description automatically generated

**Question Bank**

**A.1**

Class is the blueprint of an object. It is used to declare and create objects. Whereas object is an instance of a class.

**A.2**

Constructor Chaining is the process of calling one constructor of a class from another constructor of the same class or another class using the current object of the class.

**A.3**

A constructor that has no parameter is No-arg constructor and is also known as the default constructor.

**A.4**

If we add a return type to a constructor, then it will become a method of the class.

**A.5**

The private constructor in Java is used to create a singleton class. A singleton class is a class in Java that limits the number of objects of the declared class to one. A private constructor in Java ensures that only one object is created at a time.

**A.6**

Yes, we can use this() keyword in a method.

**A.7**

Yes, it is allowed to define a method with the same name as that of a class.

**Flipped Practical’s**

**A.1**

A) false

**PRACTICAL NO: 5**

|  |
| --- |
| **Student Name and Roll Number:** |
| **Semester /Section:** |
| **Date:** |
| **Faculty Signature:** |

|  |
| --- |
| **Objective**  To familiarize the students with the concept of Inheritance. |
| **Program Outcome**  The students will learn the concept of inheritance. |
| **Program Statement**   1. 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.     2. Create three classes:  Class Vehicle:  Vehicle Class will contain a display() function, which will say "This is a Vehicle".  Class Car:  Car Class will derive the Vehicle Class and overwrite its display() function. it will say "This is a Car".  Class Bike:  Bike Class will derive the Vehicle Class and overwrite its display() function. it will say "This is a Bike".    Write an application that reads an Integer N, which will denote the number of tyres in the vehicle. You have to create an object of the appropriate class according to the value of N and use it display() function.  If N = 2, Create a Bike Object.  If N = 4, Create a Car Object.  Create a Vehicle Object, otherwise.  Definition of Done:  DoD 1: Each class definition is stored in its own .java file.  DoD 2: Switch statement is used for identifying the appropriate class for which the object is to be invoked.    3. Define a class Box with the following instance variables: width, height and depth, all of type float. Create a new class BoxWeigth that extends Box to include weight as an instance variable. Write an application that tests the functionalities of both these classes.  Definition of Done:  DoD 1: Three java files to be defined. One for each class definition: Box, BoxWeight and BoxWeightDemo.  DoD 2: Box and BoxWeight should have three types of constructors defined: clone of an object, all dimensions specified as arguments, no argument.  DoD 3: Super is used to call base class constructors in derived class  DoD 4: Get and set functions defined as applicable in Box and BoxWeight classes.  DoD 5: Function to display volume in Box class and weight in BoxWeigth class |
| **Background Study**  Inheritance can be defined as the process where one class acquires the properties (methods and fields) of another. With the use of inheritance, the information is made manageable in a hierarchical order.  The class which inherits the properties of other is known as subclass (derived class, child class) and the class whose properties are inherited is known as superclass (base class, parent class). extends Keyword **extends** is the keyword used to inherit the properties of a class. Following is the syntax of extends keyword.  **Syntax**  class Super {  .....  .....  }  class Sub extends Super {  .....  .....  } |
| **Question Bank**  1. What is the use of super keyword?  2. What is multi-level inheritance?  3. What is the usage of inheritance? |
| **Flipped Questions**  Q1. What is the output of the following?  class A  {      {          System.out.println(1);      }  }    class B extends A  {      {          System.out.println(2);      }  }    class C extends B  {      {          System.out.println(3);      }  }    public class MainClass  {      public static void main(String[] args)      {          C c = new C();      }  }  Q2. What is the output of the following?  class A  {      public A()      {          System.out.println("Class A Constructor");      }  }    class B extends A  {      public B()      {          System.out.println("Class B Constructor");      }  }    class C extends B  {      public C()      {          System.out.println("Class C Constructor");      }  }    public class MainClass  {      public static void main(String[] args)      {          C c = new C();      }  } |

**Student Work Area**

**Algorithm/Flowchart/Code/Sample Outputs**

**Problem Statement – 1**

/\*

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.

\*/

/\*\*

 \* practical\_5\_problem\_statement\_1

 \*/

class Shape {

    void PrintMessage\_Shape() {

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

    }

}

class Rectangle extends Shape {

    void PrintMessage\_Rectangle() {

        System.out.println("This is Rectangular Shape");

    }

}

class Circle extends Shape {

    void PrintMessage\_Circle() {

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

    }

}

class Square extends Rectangle {

    void PrintMessage\_Square() {

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

    }

}

public class practical\_5\_problem\_statement\_1 {

    public static void main(String[] args) {

        Square s = new Square();

        s.PrintMessage\_Shape();

        s.PrintMessage\_Rectangle();

    }

}

**Output:**

Text, email

Description automatically generated

**Problem Statement - 2**

/\*

Create three classes:

Class Vehicle: Vehicle Class will contain a display() function, which will say "This is a Vehicle".

Class Car: Car Class will derive the Vehicle Class and overwrite its display() function. it will say "This is a Car".

Class Bike:Bike Class will derive the Vehicle Class and overwrite its display() function. it will say "This is a Bike".

Write an application that reads an Integer N, which will denote the number of tyres in the vehicle. You have to create an object of the appropriate class according to the value of N and use it display( function.

If N = 2, Create a Bike Object.

If N = 4, Create a Car Object.

Create a Vehicle Object, otherwise.

Definition of Done:

DoD 1: Each class definition is stored in its own .java file.

DoD 2: Switch statement is used for identifying the appropriate class for which the object is

to be invoked.

\*/

/\*\*

 \* practical\_5\_problem\_statement\_2

 \*/

import java.util.Scanner;

class Vehicle {

    void Display() {

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

    }

}

class Car extends Vehicle {

    void Display() {

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

    }

}

class Bike extends Vehicle {

    void Display() {

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

    }

}

public class practical\_5\_problem\_statement\_2 {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

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

        int N = sc.nextInt();

        switch (N) {

            case 2:

                Bike B1 = new Bike();

                B1.Display();

                break;

            case 4:

                Car C1 = new Car();

                C1.Display();

                break;

            default:

                System.out.println("Invalid Input!");

                break;

        }

    }

}

**Output:**

Text

Description automatically generated

**Problem Statement – 3**

/\*

Define a class Box with the following instance variables: width height and depth, all of type float. Create a new class BoxWeigth that extends Box to include weight as an instance variable. Write an application that tests the functionalities of both these classes.

Definition of Done:

DoD 1: Three java files to be defined. One for each class definition: Box, BoxWeight and BoxWeightDemo.

DoD 2: Box and BoxWeight should have three types of constructors defined: clone of an object, all dimensions specified as arguments, no argument.

DoD 3: Super is used to call base class constructors in derived class

DoD 4: Get and set functions defined as applicable in Box and BoxWeight classes.

DoD 5: Function to display volume in Box class and weight in BoxWeigth class

\*/

/\*\*

 \* practical\_5\_problem\_statement\_3

 \*/

class Box {

    float length;

    float width;

    float depth;

    public Box() {

    }

    public Box(float l, float w, float d) {

        this.length = l;

        this.width = w;

        this.depth = d;

    }

    void setlength(float len) {

        this.length = len;

    }

    void setwidth(float wid) {

        this.length = wid;

    }

    void setdep(float dep) {

        this.depth = dep;

    }

    float getlength() {

        return this.length;

    }

    float getwodth() {

        return this.width;

    }

    float getdepth() {

        return this.depth;

    }

    float volume() {

        float volume = this.length \* this.width \* this.depth;

        return volume;

    }

    class BoxWeight extends Box {

        float weight;

        void setweight(float weight) {

            this.weight = weight;

        }

        float getlength() {

            return this.weight;

        }

        float weight() {

            return weight;

        }

    }

public class practical\_5\_problem\_statement\_3 {

    public static void main(String[] args) {

    }

}

**Output:**

Text, letter

Description automatically generated

**Question Bank**

**A.1**

The super keyword refers to superclass (parent) objects. It is used to call superclass methods, and to access the superclass constructor.

**A.2**

Multilevel Inheritance occurs when a class extends a class that extends another class. For example, class C extends class B, and class B extends class A. This is referred to as multilevel Inheritance.

**A.3**

Inheritance is used to use the existing features of class.

**Flipped Practical’s**

**A.1**

1

2

3

**A.2**

Class A Constructor

Class B Constructor

Class C Constructor

|  |
| --- |
| **Student Name and Roll Number:** |
| **Semester /Section:** |
| **Date:** |
| **Faculty Signature:** |

**PRACTICAL NO. 6**

|  |
| --- |
| **Objective**  To familiarize the students with the concept of Interfaces in Java. |
| **Program Outcome**  The students will be able to understand where and how interfaces are implemented. |
| **Problem Statement**  1. 1. a) Write a program in java to check if a class can extends another class and/ can implement one and more than one interface.  b) Write a program in java to check if an interface can extend other interface.  c) Write a program in java to check if an interface can also extend multiple interfaces.  2. Define the interface / class hierarchy as detailed in the following class diagram  **Definition of Done:**  DOD 1: The class definitions are defined as per the class diagram.  DOD 2: Each class definition is stored in its own .java file.  DOD 3: Base class constructors are invoked using super keyword  DOD 4: Function overriding is applied wherever applicable.      3. We have to calculate the area of a rectangle, a square and a circle. Create an abstract class 'Shape' with three abstract methods namely 'RectangleArea' taking two parameters, 'SquareArea' and 'CircleArea' taking one parameter each. The parameters of 'RectangleArea' are its length and breadth, that of 'SquareArea' is its side and that of 'CircleArea' is its radius. Now create another class 'Area' containing all the three methods 'RectangleArea', 'SquareArea' and 'CircleArea' for printing the area of rectangle, square and circle respectively. Create an object of class 'Area' and call all the three methods. |
| **Background Study**  An **interface in Java** is a blueprint of a class. It has static constants and abstract methods.  The interface in Java is *a mechanism to achieve*[abstraction](https://www.javatpoint.com/abstract-class-in-java). There can be only abstract methods in the Java interface, not method body. It is used to achieve abstraction and multiple [inheritance in Java](https://www.javatpoint.com/inheritance-in-java).  In other words, you can say that interfaces can have abstract methods and variables. It cannot have a method body. |
| **Flipped Practicals**  1. What is the output of this program?  **interface** calculate {  **void** cal(**int** item);  }  **class** display **implements** calculate {  **int** x;  **public** **void** cal(**int** item) {  x = item \* item;  }  }  **class** interfaces {  **public** **static** **void** main(String args[]) {  display arr = **new** display;  arr.x = 0;  arr.cal(2);  System.out.print(arr.x);  }  }  a) 0 b) 2 c) 4 d) None of the mentioned  2. Which of the following package stores all the standard java classes? a) lang b) java c) util d) java.packages  3. Determine output of the following code.  interface A { }  class C { }  class D extends C { }  class B extends D implements A { }  public class Test extends Thread{  public static void main(String[] args){  B b = new B();  if (b instanceof A)  System.out.println("b is an instance of A");  if (b instanceof C)  System.out.println("b is an instance of C");  }  }   1. b is an instance of A. 2. b is an instance of C. 3. b is an instance of A followed by b is an instance of C. |
| **Question Bank**   1. Can an interface be final? 2. Can an abstract class implement an interface? 3. Can you declare an interface method static? 4. What is the difference between abstract class and interface? |

**Student Work Area**

**Algorithm/Flowchart/Code/Sample Outputs**

**Problem Statement – 1**

/\*

a) Write a program in java to check if a class can extends another class and can implement one and more than one interface.

b) Write a program in java to check if an interface can extend other interface.

c) Write a program in java to check if an interface can also extend multiple interfaces.

\*/

/\*\*

 \* practical\_6\_problem\_statement\_1

 \*/

interface interface1 {

    void display1();

}

interface interface2 extends interface1 {

    void display2();

}

interface interface3 extends interface1, interface2 {

    void display3();

}

class class1 {

    void display() {

        System.out.println("This is class1.");

    }

}

class class2 extends class1 implements interface1, interface2 {

    @Override

    public void display() {

        System.out.println("This is class2.");

    }

    public void display1() {

        System.out.println("This is interface1 display function.");

    }

    public void display2() {

        System.out.println("This is interface2 display function.");

    }

}

public class practical\_6\_problem\_statement\_1 {

    public static void main(String[] args) {

        class2 c = new class2();

        c.display();

        c.display1();

        c.display1();

    }

}

**Output:**

Text, letter, email

Description automatically generated

**Problem Statement – 2**

**Circle Class File**

interface GeometricalObject {

    double getPerimeter();

    double getArea();

}

class Circle implements GeometricalObject {

    double radius = 1.0;

    public Circle() {

    }

    public Circle(double r) {

        this.radius = r;

    }

    public double getPerimeter() {

        return 2 \* Math.PI \* this.radius;

    }

    public double getArea() {

        return Math.PI \* this.radius \* this.radius;

    }

}

**ResizableCircle Java File**

interface Resizeable {

    void resize(int percent);

}

class ResizableCircle extends Circle implements Resizeable {

    public ResizableCircle() {

        super();

    }

    public void resize(int percent) {

        double p = percent;

        double new\_radius = this.radius + this.radius \* p / 100;

        this.radius = new\_radius;

    }

}

**Main Java File**

/\*

Define the interface / class hierarchy as detailed in the following class diagram

Definition of Done:

DOD 1: The class definitions are defined as per the class diagram.

DOD 2: Each class definition is stored in its own .java file.

DOD 3: Base class constructors are invoked using super keyword

DOD 4: Function overriding is applied wherever applicable.

\*/

/\*\*

 \* practical\_6\_problem\_statement\_2

 \*/

public class practical\_6\_problem\_statement\_2 {

    public static void main(String[] args) {

        ResizableCircle rc = new ResizableCircle();

        rc.radius = 5;

        System.out.print("\nArea: " + rc.getArea());

        System.out.print("\nPerimeter: " + rc.getPerimeter());

        System.out.print("\n\nAfter Resizing Circle");

        System.out.println(rc.radius);

        System.out.println();

        System.out.print("\nArea: " + rc.getArea());

        System.out.print("\nPerimeter: " + rc.getPerimeter());

    }

}

**Output:**

Graphical user interface, text, application, email

Description automatically generated

**Problem Statement - 3**

/\*

We have to calculate the area of a rectangle, a square and a circle. Create an abstract class 'Shape' with three abstract methods namely 'RectangleArea' taking two parameters, 'SquareArea' and 'CircleArea' taking one parameter each. The parameters of 'RectangleArea' are its length and breadth, that of 'SquareArea' is its side and that of 'CircleArea' is its radius. Now create another class 'Area' containing all the three methods 'RectangleArea', 'SquareArea' and 'CircleArea' for printing the area of rectangle, square and circle respectively. Create an object of class 'Area' and call all the three methods.

\*/

/\*\*

 \* practical\_6\_problem\_statement\_3

 \*/

abstract class Shape {

    abstract double RectangleArea(double l, double b);

    abstract double SquareArea(double s);

    abstract double CircleArea(double r);

}

class Area extends Shape {

    double RectangleArea(double length, double breadth) {

        return length \* breadth;

    }

    double SquareArea(double side) {

        return side \* side;

    }

    double CircleArea(double radius) {

        return Math.PI \* radius \* radius;

    }

}

public class practical\_6\_problem\_statement\_3 {

    public static void main(String[] args) {

        Area a = new Area();

        System.out.println("Area of Rectangle: " + a.RectangleArea(5, 7));

        System.out.println("Area of Square: " + a.SquareArea(6));

        System.out.println("Area of Circle: " + a.CircleArea(8));

    }

}

**Output:**

Graphical user interface, text, application, letter, email

Description automatically generated

**Question Bank**

**A.1**

If you make an interface final, you cannot implement its methods which defies the very purpose of the interfaces. Therefore, you cannot make an interface final in Java.

**A.2**

Java Abstract class can implement interfaces without even providing the implementation of interface methods.

**A.3**

No, we cannot declare interface methods as static because static methods can not be overridden.

**A.4**

An abstract class allows you to create functionality that subclasses can implement or override. An interface only allows you to define functionality, not implement it.

**Flipped Practical’s**

**A.1**

4

**A.2**

(b) lang

**A.3**

(c) b is an instance of A followed by b is an instance of C

**Experiment No: 7**

|  |
| --- |
| **Student Name and Roll Number:** |
| **Semester /Section:** |
| **Date:** |
| **Faculty Signature:** |

|  |
| --- |
| **Objective**  To familiarize the students with package in Java. |
| **Program Outcome**  The students will learn the concept of package in Java. They will be able to understand |
| **Problem Statement**  **1.** Create a Java package called exercises. Inside the exercises package, create another package (subpackage) called java. Create a Java class called PackageDemo inside the java package. Insert a display() method inside the PackageDemo class. Inside the method, insert this statement:System.out.println("PackageDemo executed");Write a PackageDemoDriver class within the same package to run the display () method of PackageDemo class.  2. Write a java program outside the above defined package that imports the package PackageDemo and calls its display() function. |
| **Background Study**  A **java package** is a group of similar types of classes, interfaces and sub-packages.  Package in java can be categorized in two form, built-in package and user-defined package.  There are many built-in packages such as java, lang, awt, javax, swing, net, io, util, sql etc. |
| **Flipped Practicals**  1. What will be the output of the following Java program?  **package** pkg;  **class** display  {  **int** x;  **void** show()  {  **if** (x > 1)  System.out.print(x + " ");  }  }  **class** packages  {  **public** **static** **void** main(String args[])  {  display[] arr=**new** display[3];  **for**(**int** i=0;i<3;i++)  arr[i]=**new** display();  arr[0].x = 0;  arr[1].x = 1;  arr[2].x = 2;  **for** (**int** i = 0; i < 3; ++i)  arr[i].show();  }  }  Note : packages.class file is in directory pkg; a) 0 b) 1 c) 2 d) 0 1 2 |
| **Qu Question Bank**  1. What are packages? what is use of packages ?  2. What is difference between importing "java.applet.Applet" and "java.applet.\* "?  3.What do you understand by package access specifier?  **4.** By default,all program import the java.lang package.     True/False ?  **5.**Java compiler stores the .class files in the path specified in CLASSPATH environmental    variable.    True/False ?  **6.**User-defined package can also be imported just like the standard packages     True/False ?  **7.**A \_\_\_\_\_\_\_ is used to separate the hierarchy of the class while declaring an Import statement.   **8.**All standard classes of Java are included within a package called \_\_\_\_\_. |

**Student Work Area**

**Algorithm/Flowchart/Code/Sample Outputs**

**Problem Statement – 1**

**PackageDemo Class Java File**

package exercise.java;

public class PackageDemo {

    public void display() {

        System.out.println("Package Demo Executed");

    }

}

**Package DemoDriver Class Java File**

/\*

Create a Java package called exercises. Inside the exercises package, create another package (subpackage) called java. Create a Java class called PackageDemo inside the java package. Insert a display() method inside the PackageDemo class.

Inside the method, insert this statement:System.out.println("PackageDemo executed");Write a PackageDemoDriver class within the same package to run the display () method of PackageDemo class.

\*/

/\*\*

 \* practical\_7\_problem\_statement\_2

 \*/

import exercise.java.PackageDemo;

public class PackageDemoDriver {

    public static void main(String[] args) {

        PackageDemo PDD = new PackageDemo();

        PDD.display();

    }

}

**Output:**

Text, letter

Description automatically generated

**Problem Statement – 2**

/\*

Write a java program outside the above defined package that imports the package PackageDemo and calls its display() function.

\*/

import exercise.java.PackageDemo;

/\*\*

 \* practical\_7\_problem\_statement\_2

 \*/

public class practical\_7\_problem\_statement\_2 {

    public static void main(String[] args) {

        PackageDemo PD = new PackageDemo();

        PD.display();

    }

}

**Output:**

Text, letter

Description automatically generated

**Question Bank**

**A.1**

A package is a container of a group of related classes where some of the classes are accessible are exposed and others are kept for internal purpose. Packages are used to avoid name conflicts, and to write a better maintainable code.

**A.2**

The difference between importing "java.applet.Applet" and "java.applet.\* is that “java.applet.Applet” is the syntax to import Applet class from the applet subpackage whereas “java.applet.\*” is the syntax to import all the classes from the subpackage applet from the package java.

**A.3**

Java provides entities called “Access Modifiers or access specifiers” that help us to restrict the scope or visibility of a package, class, constructor, methods, variables, or other data members. These access modifiers are also called “Visibility Specifiers”.

**A.4**

True

**A.5**

False

**A.6**

True

**A.7**

Package

**A.8**

lang

**Flipped Practical’s**

**A.1**

2

**A.2**

(b) lang

**A.3**

(c) b is an instance of A followed by b is an instance of C

**PRACTICAL NO.8**

|  |
| --- |
| **Student Name and Roll Number:** |
| **Semester /Section:** |
| **Date:** |
| **Faculty Signature:** |

|  |
| --- |
| **Objective**  To familiarize the students with exception in java. |
| **Program Outcome**  The students will learn the concept of exceptions in Java. |
| **Problem Statement**     1. Write a program that takes 5 integer command line arguments. Create a user defined Exception named CheckArgumentException to check the number of arguments passed through command line. If the number of arguments is less than five, throw the CheckArgumentException, else print the addition of all five numbers.   **Definition of Done:**  DOD 1:Create a user-defined exception by the name CheckArgumentException  DOD 2:Ask the user to enter the number of arguments  DOD 3:Use for loop to enter the arguments  DOD 4: Calculate the sum of the values entered   1. Create a class with a main() method that *throws* an object of class **Exception** inside a *try* block. Give the constructor for **Exception** a String argument. Catch the exception inside a *catch* clause and print the String argument. Add a *finally* clause and print a message to prove you were there. |
| **Background Study**  An exception (or exceptional event) is a problem that arises during the execution of a program. When an **Exception** occurs the normal flow of the program is disrupted and the program/Application terminates abnormally, which is not recommended, therefore, these exceptions are to be handled.  An exception can occur for many different reasons. Following are some scenarios where an exception occurs.   * A user has entered an invalid data. * A file that needs to be opened cannot be found. * A network connection has been lost in the middle of communications or the JVM has run out of memory. |
| **QuestionQuestion Bank**  1. When does Exceptions in Java arises in code sequence? a) Run Time b) Compilation Time c) Can Occur Any Time d) None of the mentioned  3. Which of these keywords must be used to monitor for exceptions? a) try b) finally c) throw d) catch  4. Which of these keywords must be used to handle the exception thrown by try block in some rational manner? a) try b) finally c) throw d) catch  5. Which of these keywords is used to manually throw an exception? a) try b) finally c) throw d) catch |
| **Flip Flipped Practicals**    1. What will be the output of the following Java program?  **class** exception\_handling  {  **public** **static** **void** main(String args[])  {  **try**  {  System.out.print("Hello" + " " + 1 / 0);  }  **catch**(ArithmeticException e)  {  System.out.print("World");  }  }  }  a) Hello b) World c) HelloWorld d) Hello World  2. What will be the output of the following Java program?  **class** exception\_handling  {  **public** **static** **void** main(String args[])  {  **try**  {  **int** a, b;  b = 0;  a = 5 / b;  System.out.print("A");  }  **catch**(ArithmeticException e)  {  System.out.print("B");  }  }  }  a) A b) B c) Compilation Error d) Runtime Error |

**Student Work Area**

**Algorithm/Flowchart/Code/Sample Outputs**

**Problem Statement – 1**

/\*

Write a program that takes 5 integer command line arguments. Create a user defined Exception named CheckArgumentException to check the number of arguments  passed through command line. If the number of arguments is less than five, throw the CheckArgumentException, else print the addition of all five numbers.

Definition of Done:

DOD 1:Create a user-defined exception by the name CheckArgumentException

DOD 2:Ask the user to enter the number of arguments

DOD 3:Use for loop to enter the arguments

DOD 4:Calculate the sum of the values entered

\*/

/\*\*

 \* practical\_8\_problem\_statement\_1

 \*/

class CheckArgumentException extends Exception {

}

public class practical\_8\_problem\_statement\_1 {

    public static void main(String[] args) {

        int sum = 0;

        if (args.length < 5) {

            try {

                throw new CheckArgumentException();

            } catch (CheckArgumentException e) {

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

            }

        } else {

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

                sum = sum + Integer.parseInt(args[i]);

            }

            System.out.println("Sum is: " + sum);

        }

    }

}

**Output:**

Text, letter

Description automatically generated

**Problem Statement – 2**

/\*

Create a class with a main() method that throws an object of class Exception inside a try block. Give the constructor for Exception a String argument. Catch the exception inside a catch clause and print the String argument. Add a finally clause and print a message to prove you were there.

\*/

/\*\*

 \* practical\_8\_problem\_statement\_2

 \*/

public class practical\_8\_problem\_statement\_2 {

    public static void main(String[] args) {

        try {

            String argument = "Practical 8 Problem Statement 2";

            throw new Exception(argument);

        } catch (Exception e) {

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

        } finally {

            System.out.println("Final Block");

        }

    }

}

**Output:**

Text, letter

Description automatically generated

**Question Bank**

**A.1**

(a) Run Time

**A.2**

(a) try

**A.3**

(d) catch

**A.4**

(c)throw

**Flipped Practical’s**

**A.1**

(b) World

**A.2**

(b) B

|  |
| --- |
| **Student Name and Roll Number:** |
| **Semester /Section:** |
| **Date:** |
| **Faculty Signature:** |

**PRACTICAL NO. 9**

|  |
| --- |
| **Objective**  To familiarize the students with File handling. |
| **Program Outcome**  The students will learn the concept of files in Java. |
| **Problem Statement**  1. 1. Write aprogram to check if the file exist is a file or directory.  2. W 2. Write aprogram to change the file permissions.  3.W 3. Write a program to perform simple read and write operation into file.  4. Write a program that writes an Serial No.(int), First Name (String), CGPA (float) and Grade(char) into a text file using bufferedWriter and displays the contents using bufferedReader. |
| **B**  **Background Study**  File handling is an important part of any application. Java has several methods for creating, reading, updating, and deleting files. Java File Handling The File class from the java.io package, allows us to work with files.  To use the File class, create an object of the class, and specify the filename or directory name: Example import java.io.File; // Import the File class  File myObj = new File("filename.txt"); // Specify the filename |
| **QuestiQ Question Bank**  1. Which of these exception is thrown in cases when the file specified for writing is not found? a) IOException b) FileException c) FileNotFoundException d) FileInputException  2. Which of these methods are used to read in from file? a) get() b) read() c) scan() d) readFileInput()  3. Which of these values is returned by read() method is end of file (EOF) is encountered? a) 0 b) 1 c) -1 d) Null |
| **Flipped Practicals**  1. What will be the output of the following Java program?   1. **import** java.io.\*; 2. **class** filesinputoutput 3. { 4. **public** **static** **void** main(String args[]) 5. { 6. InputStream obj = **new** FileInputStream("inputoutput.java"); 7. System.out.print(obj.available()); 8. } 9. }   Note: inputoutput.java is stored in the disk. a) true b) false c) prints number of bytes in file d) prints number of characters in the file  2. What will be the output of the following Java program?   1. **import** java.io.\*; 2. **public** **class** filesinputoutput 3. { 4. **public** **static** **void** main(String[] args) 5. { 6. String obj = "abc"; 7. **byte** b[] = obj.getBytes(); 8. ByteArrayInputStream obj1 = **new** ByteArrayInputStream(b); 9. **for** (**int** i = 0; i < 2; ++ i) 10. { 11. **int** c; 12. **while**((c = obj1.read()) != -1) 13. { 14. **if**(i == 0) 15. { 16. System.out.print(Character.toUpperCase((**char**)c)); 17. obj2.write(1); 18. } 19. } 20. System.out.print(obj2); 21. } 22. } 23. }   a) AaBaCa b) ABCaaa c) AaaBaaCaa d) AaBaaCaaa |

**Student Work Area**

**Algorithm/Flowchart/Code/Sample Outputs**

**Problem Statement – 1**

/\*

Write a program to check if the file exist is a file or directory.

\*/

import java.io.\*;

/\*\*

 \* practical\_9\_problem\_statement\_1

 \*/

public class practical\_9\_problem\_statement\_1 {

    public static void main(String[] args) {

        try {

            File f = new File("practical\_9\_problem\_statement\_1.txt");

            f.createNewFile();

            if (f.exists()) {

                System.out.println("Is a File.");

            }

            else if (f.isDirectory()) {

                System.out.println("Is a Direcotry.");

            }

        }

        catch (Exception e) {

            System.out.println("File Does Not Exists.");

        }

    }

}

**Output:**

Text

Description automatically generated

**Problem Statement – 2**

/\*

Write a program to change the file permissions.

\*/

import java.io.\*;

/\*\*

 \* practical\_9\_problem\_statement\_2

 \*/

public class practical\_9\_problem\_statement\_2 {

    public static void main(String[] args) {

        File file = new File("practical\_9\_problem\_statement\_2.txt");

        if (file.exists()) {

            file.setExecutable(true);

            file.setReadable(true);

            file.setWritable(false);

            System.out.println("File Permissions Changed.");

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

            System.out.println("Executable: " + file.canExecute());

            System.out.println("Readable: " + file.canRead());

            System.out.println("Writable: " + file.canWrite());

        }

        else {

            System.out.println("File Not Found.");

        }

    }

}

**Output:**

Graphical user interface, text, application, email

Description automatically generated

**Problem Statement – 3**

/\*

Write a program to perform simple read and write operation into file.

\*/

import java.io.\*;

/\*\*

 \* practical\_9\_problem\_statement\_3

 \*/

public class practical\_9\_problem\_statement\_3 {

    public static void writeFile(String fileName) {

        try {

            FileWriter fileWriter = new FileWriter(fileName);

            BufferedWriter bufferWriter = new BufferedWriter(fileWriter);

            bufferWriter.write("This is sample file for Practical 9 Problem Statement 3.");

            bufferWriter.close();

            fileWriter.close();

            System.out.println("\nText Succesfully Written in File");

        } catch (Exception e) {

            System.out.println(e);

        }

    }

    public static void readFile(String fileName) {

        try {

            FileReader fileReader = new FileReader(fileName);

            BufferedReader bufferReader = new BufferedReader(fileReader);

            System.out.println("\nText in the File: ");

            String st;

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

                System.out.println(st);

            }

            bufferReader.close();

            fileReader.close();

            System.out.println("\nText Succesfully Read From File");

        } catch (Exception e) {

            System.out.println(e);

        }

    }

    public static void main(String[] args) {

        writeFile("practical\_9\_problem\_statement\_3.txt");

        readFile("practical\_9\_problem\_statement\_3.txt");

    }

}

**Output**Graphical user interface, text, application, email

Description automatically generated

**Problem Statement – 4**

/\*

Write a program that writes an Serial No.(int), First Name (String), CGPA (float) and Grade(char) into a text file using bufferedWriter and displays the contents using bufferedReader.

\*/

import java.io.\*;

import java.util.Scanner;

/\*\*

 \* practical\_9\_problem\_statement\_4

 \*/

public class practical\_9\_problem\_statement\_4 {

    public static void writeFile(String fileName) {

        Scanner sc = new Scanner(System.in);

        try {

            FileWriter fileWriter = new FileWriter(fileName);

            BufferedWriter bufferWriter = new BufferedWriter(fileWriter);

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

            String serialNumber = sc.nextLine();

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

            String name = sc.nextLine();

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

            String cgpa = sc.nextLine();

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

            char grade = sc.next().charAt(0);

            bufferWriter.write(serialNumber);

            bufferWriter.write("\t");

            bufferWriter.write(name);

            bufferWriter.write("\t");

            bufferWriter.write(cgpa);

            bufferWriter.write("\t");

            bufferWriter.write(grade);

            bufferWriter.close();

            fileWriter.close();

            System.out.println("\nText Succesfully Written in File");

        } catch (Exception e) {

            System.out.println(e);

        }

    }

    public static void readFile(String fileName) {

        try {

            FileReader fileReader = new FileReader(fileName);

            BufferedReader bufferReader = new BufferedReader(fileReader);

            System.out.println("Content in the File: ");

            String st;

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

                System.out.println(st);

            }

            bufferReader.close();

            fileReader.close();

        } catch (Exception e) {

            System.out.println(e);

        }

    }

    public static void main(String[] args) {

        writeFile("practical\_9\_problem\_statement\_4.txt");

        readFile("practical\_9\_problem\_statement\_4.txt");

    }

}

**Output:**

Graphical user interface, text, application, email

Description automatically generated

**Question Bank**

**A.1**

(c) File Not Found Exception

**A.2**

(b) Read

**A.3**

(c) -1

**Flipped Practical’s**

**A.1**

(b) World

**A.2**

(b) B