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:*

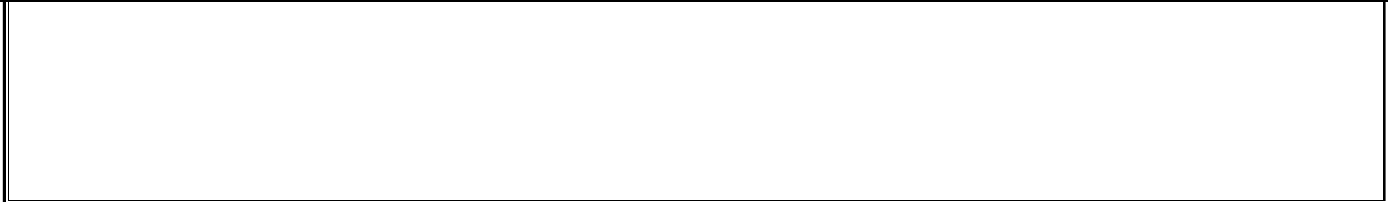
**School of Engineering and Technology Department of Computer Science & Engineering The NorthCap University Gurugram**

* **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.

make full use of labs beyond normal lab hours.

Labs are open up to 7 PM upon request. Students are encouraged to

# 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 |

### 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

### LAB REQUIREMENTS

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

### GENERAL INSTRUCTIONS

* 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.
  2. **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. **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.

### 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 |

1. **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. |

**Table of Contents**

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

# Index

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

## PRACTICAL NO. 1

|  |
| --- |
| **Student Name and Roll Number:** Arjun Bhardwaj 21csu211 |
| **Semester /Section:** 2nd Semester / CSE D |
| **Date:** 15/03/2022 |
| **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.**  CODE:  class Prac{  public static void main(String args[]){ int a=66, b=78;  int sum,sub,mul,div,mod; sum=a+b;  sub=a-b; mul=a\*b;  div=a/b; |

**FOCP II Lab Manual(CSL108) 2021-22**

|  |
| --- |
| mod=a%b; System.out.println("Addition:"+sum); System.out.println("Subtraction:"+sub); System.out.println("Multiplication:"+mul); System.out.println("Division:"+div); System.out.println("Modulus:"+mod);  }  }  OUTPUT:    **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.**  class Prac{  public static void main(String args[]){  // Arjun Bhardwaj 21csu211  int min=618960; int y,m,d,t;  // 1 day= 1440 min; 1 month = 43800; 1 year= 525600  y=min/525600; t=min%525600; m=t/43800; d=(t%43800)/1440;  System.out.println("Minutes:"+min); System.out.println("YY:MM:DD = "+y+":"+m+":"+d);  }  } |

**FOCP II Lab Manual(CSL108) 2021-22**

|  |
| --- |
| OUTPUT: |

**FOCP II Lab Manual(CSL108) 2021-22**

|  |
| --- |
| 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?   **Answer**: Java is considered dynamic because of Bytecode. The source code is written in one platform that code can be executed on any platform. It loads the class file during runtime only. Hence, anything that happens in runtime is dynamic.   1. What is Java Virtual Machine and how it is considered in the context of Java’s platform independent feature?   **Answer**: Java Virtual Machine (JVM) is a specification that provides a runtime environment in which java bytecode(. class files) can be executed.  Platform independent language means that once compiled you can execute the program on any platform (OS). Java is platform-independent. Because the Java compiler converts the source code to bytecode, which is Intermidiate Language. Bytecode can be executed on any platform (OS) using JVM( Java Virtual Machine).   1. List two Java IDE’s? List some Java keywords(unlike C, C++ keywords)?   **Answer**: NetBeans, Eclipse, etc.  import, super, finally, System and scanner, etc   1. Consider the following class: public class IdentifyMyParts { public static int x = 7;   public int y = 3;  }   * 1. What are the class variables?   **Answer**: x   * 1. What are the instance variables?   **Answer**: y |

**FOCP II Lab Manual(CSL108) 2021-22**

|  |
| --- |
| **Flipped practical**   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);  **Answer:**  a.y = 5  b.y = 6  a.x = 1  b.x = 2  IdentifyMyParts.x = 2   1. 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());  }  }  **Answer:** The code never creates a Rectangle object. With this simple program, the compiler generates an error. However, in a more realistic situation, myRect might be initialized to null in one  place, say in a constructor, and used later. In that case, the program will compile just fine, but will generate a NullPointerException at runtime. |

**FOCP II Lab Manual(CSL108) 2021-22**

## PRACTICAL NO. 2

|  |
| --- |
| **Student Name and Roll Number:** Arjun Bhardwaj 21csu211 |
| **Semester /Section:** 2nd Semester / CSE-D |
| **Date:** 22/03/2022 |
| **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.**  import java.util.\*; import java.lang.Math; class Prac{  public static void main(String args[]){  Scanner scan=new Scanner(System.in); System.out.print("Enter Coefficients a,b,c:"); double a=scan.nextDouble();  double b=scan.nextDouble(); double c=scan.nextDouble(); double d= (Math.pow(b,2.0))-4\*a\*c; if (d>=0){  double r1=(-b+Math.sqrt(d))/2\*a; double r2=(-b-Math.sqrt(d))/2\*a;  System.out.println("Roots are:"+r1+"\n"+r2);  } |

**FOCP II Lab Manual(CSL108) 2021-22**

|  |
| --- |
| else{  double r=-b/2\*a;  double i=Math.sqrt(-d)/2\*a; System.out.println("Roots are:"+r+("+")+i+"i");  System.out.println("Roots are:"+r+("-")+i+"i");  }  }  }  OUTPUT:    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.\*; class Prac{  public static void main(String args[]){  Scanner scan= new Scanner(System.in); String a= scan.next().toLowerCase();  if (a.length()==1) { |

**FOCP II Lab Manual(CSL108) 2021-22**

|  |
| --- |
| char b=a.charAt(0);  if (b=='a'|| b=='e'||b=='i'||b=='o'||b=='u') { System.out.println("Vowel");  }  else if (b>='a' && b<='z'){ System.out.println("Consonant");  }  }  else{  System.out.println("You've entered a string.");  }  }  }  OUTPUT:    3. **Write a Java program to print the following structure:**  **\***  **\*\***  **\*\*\***  **\*\*\*\***  **\*\*\*\*\***  **\*\*\*\*\*\***  class Prac{  public static void main(String args[]){  for (int i=1;i<7;i++ ) {  for(int j=1;j<=i;j++) {  System.out.print("\*");  }  System.out.print("\n");  } |

**FOCP II Lab Manual(CSL108) 2021-22**

|  |
| --- |
| }  } OUTPUT: |

**FOCP II Lab Manual(CSL108) 2021-22**

|  |
| --- |
| **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 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 if-then statement. 2. The switch statement allows for any number of possible execution paths. 3. The do-While statement is similar to the while statement but evaluates its expression at the loop. 4. **How do you write an infinite loop using the for statement?**   Answer: don’t give condition   1. **How do you write an infinite loop using the while statement?**   Answer: passing true to the condition   1. **Which looping process checks the test condition at the end of the loop?**   Answer: Do-while loop   1. **Why do we use the continue statement?**   Answer: The continue keyword is used to end the current iteration in a for loop (or a while loop), and continues to the next iteration.   1. **What is the size of a boolean variable?**   Answer: 2 Bytes   1. **Which looping process is best used when the number of iterations is known?**   Answer: for loop is the best process used when the number of iterations. |

**FOCP II Lab Manual(CSL108) 2021-22**

|  |
| --- |
| **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");   1. What output do you think the code will produce if a Number is 3? 2. Write a test program containing the previous code snippet; make a number 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? 3. Using only spaces and line breaks, reformat the code snippet to make the control flow easier to understand. 4. Use braces, { and }, to further clarify the code.   **Answer:**  a)  second string third string  b)  NestedIf second string third string  3 is greater than or equal to 0, so execution progresses to the second if statement. The second if statement's test fails because 3 is not equal to 0. Thus, the else clause executes (since it's attached to the second if statement). Thus, second string is displayed. The final println is completely outside of any if statement, so it always gets executed, and thus third string is always displayed.  C)  if (aNumber >= 0)  if (aNumber == 0)  System.out.println("first string");  else  System.out.println("second string"); System.out.println("third string");  d)  if (aNumber >= 0) {  if (aNumber == 0) {  System.out.println("first string");  } else {  System.out.println("second string");  }  }  System.out.println("third string"); |

**FOCP II Lab Manual(CSL108) 2021-22**

|  |
| --- |
| 1. What's wrong? for (int k = 2, k <= 12, k++).   **Answer:** There should be a semicolon at the end of the statement   1. 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?   **Answer:** {} are used for more than one statement in the block of a for loop.   1. What value is stored in num at the end of this looping? for (num = 1; num <= 5; num++)   **Answer:** 5 |

**FOCP II Lab Manual(CSL108) 2021-22**

## PRACTICAL NO. 3

|  |
| --- |
| **Student Name and Roll Number:** Arjun Bhardwaj 21csu211 |
| **Semester /Section:** 2nd Semester / CSE-B-3 |
| **Date:** 29/03/2022 |
| **Faculty Signature:** |

|  |
| --- |
| **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.**  class Prac{  public static void main(String args[]){ int arr[]=new int[6];  Scanner scan= new Scanner(System.in); for (int i=0;i<6;i++) {  System.out.println("Enter elements:"); arr[i]=scan.nextInt();  }  int m=arr[0],n=0;  for (int j=0;j<arr.length ;j++ ) { if (arr[j]<m) {  m=arr[j];  }  }  for (int k=0;k<arr.length ;k++ ) { if (arr[k]>n) {  n=arr[k]; |

**FOCP II Lab Manual(CSL108) 2021-22**

|  |
| --- |
| }  }  System.out.println("Max:"+n); System.out.println("Min:"+m);  }  } OUTPUT:  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 the 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.** |

**FOCP II Lab Manual(CSL108) 2021-22**

|  |
| --- |
| Import class Prac{  public static void main(String args[]){  int arr[]=new int[10];  Scanner scan=new Scanner(System.in); for (int i=0;i<10;i++) {  System.out.println("Enter elements:"); arr[i]=scan.nextInt();  }  System.out.print("["); for (int k=0;k<10 ;k++ ) {  System.out.print(arr[k]+" ");  }  System.out.print("]"); System.out.print("Enter element:"); int a=scan.nextInt();  int flag=0;  for (int j=0;j<arr.length ;j++ ) { if (arr[j]==a) {  System.out.println("Your element is at:"+j); flag=1;  }  }  if (flag==0) {  System.out.println("Element not in given array.");  }  }  }  OUTPUT: |

**FOCP II Lab Manual(CSL108) 2021-22**

|  |
| --- |
|  |

**FOCP II Lab Manual(CSL108) 2021-22**

|  |
| --- |
| 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.**  class Prac{  public static void main(String args[]){  int arr[]=new int[10];  int odd[]=new int[10]; int even[]=new int[10];  Scanner scan=new Scanner(System.in); for (int i=0;i<10;i++) {  System.out.println("Enter elements:"); arr[i]=scan.nextInt();  }  System.out.println("Array:"); for (int i=0;i<10;i++) {  System.out.print(arr[i]+" ");  }  for (int j=0;j<10;j++ ) {  if (arr[j]%2==0){  even[j]=arr[j];  }  else{  odd[j]=arr[j];  }  }  System.out.print("ODD:");  for (int k=0;k<odd.length;k++ ) { System.out.print(odd[k]+" ");  }  System.out.print("EVEN:");  for (int k=0;k<even.length;k++ ) { System.out.print(even[k]+" ");  } |

**FOCP II Lab Manual(CSL108) 2021-22**

|  |
| --- |
| }  } OUTPUT:     1. **rite a Java program to read numbers in an integer array of size 5 and display the following (using functions for each functionality):**    1. **Sum of all the elements**    2. **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.**  class Prac{  class Summ{  void sum(int arr[]){  int s=0;  for (int i:arr) {  s=s+i;  }  System.out.println("Sum:"+s);  }  void sal(int arr[]){  int sa=0;  for (int i=0;i<5;i=i+2) {  sa=sa+arr[i];  }  System.out.println("Sum of alernate elements:"+sa);  }  } |

**FOCP II Lab Manual(CSL108) 2021-22**

|  |
| --- |
| public static void main(String args[]){  int arr[]=new int[5];  Scanner scan=new Scanner(System.in); System.out.println("Enter elements:"); for (int i=0;i<5;i++) {  arr[i]=scan.nextInt();  }  System.out.println("Array:"); for (int i=0;i<5;i++) {  System.out.print(arr[i]+" ");  }  Summ obj=new Summ(); obj.sum(arr);  obj.sal(arr);  }  }  OUTPUT: |

**FOCP II Lab Manual(CSL108) 2021-22**

|  |
| --- |
| **Question Bank:**   1. Can you pass the negative number as an array size?   **ANSWER:** Array dimensions cannot have a negative size.   1. Can you change the size of the array once you define it?   **ANSWER:** The size of the array is determined at the time of its creation or, initialization once it is done you cannot change the size of the array.   1. What is an anonymous array?   **ANSWER:** An array without any name   1. What is the difference between int[] a and int a[] ?   **ANSWER:** There is no difference between these two types of array declaration.   1. What are jagged arrays in java? Give example?   **ANSWER:** A jagged array is a multidimensional array where member arrays are of different sizes. |

**FOCP II Lab Manual(CSL108) 2021-22**

|  |
| --- |
| **Flipped Practicals**   1. **Which of these is an incorrect array declaration?**    1. **int arr[] = new int[5]**    2. **int [] arr = new int[5]**    3. **int arr[] = new int[5]**    4. **int arr[] = int [5] new ANSWER:** d)   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;**  **}**  **}**  **ANSWER:** 60, 30, 0, 20, 0 |

**FOCP II Lab Manual(CSL108) 2021-22**

## PRACTICAL NO: 4

|  |
| --- |
| **Student Name and Roll Number:** Arjun Bhardwaj 21csu211 |
| **Semester /Section:** 2nd Semester / CSE-D |
| **Date:** 12/04/2022 |
| **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.**  class Student{  int roll\_no; String name;  public static void main(String args[]){ Student obj=new Student(); obj.roll\_no=2; obj.name="John";  System.out.println(obj.name+"-"+obj.roll\_no);  }  } OUTPUT: |

**FOCP II Lab Manual(CSL108) 2021-22**

|  |
| --- |
| 1. **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.**   class Triangle{  int side1; int side2; int side3;  Triangle(){  side1=3; side2=4; side3=5;  }  }  class Student{  public static void main(String args[]){ Triangle t1=new Triangle();  System.out.println("Perimeter:"+(t1.side1+t1.side2+t1.side3)); System.out.println("Area:"+(t1.side1\*t1.side2\*t1.side3));  }  }  OUTPUT:     1. **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.**   import java.util.\*; class Area{  double length; |

**FOCP II Lab Manual(CSL108) 2021-22**

|  |
| --- |
| double breadth;  Area(double l,double b){ length=l; breadth=b;  }  double returnarea(){  return length\*breadth;  }  }  class Student{  public static void main(String args[]){ double l,b,area;  Scanner scan=new Scanner(System.in); System.out.println("Enter length,breadth:"); l=scan.nextDouble();  b=scan.nextDouble(); System.out.println("Length,Breadth="+l+","+b); Area r1=new Area(l,b);  area=r1.returnarea(); System.out.println("Area"+area);  }  }    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.**  class Complex{  void sum(int a,int ax, int b, int bx){ System.out.println("Sum:"+(a+b)+"+"+(ax+bx)+"i");  }  void sub(int a,int ax, int b, int bx){ System.out.println("Difference:"+(a-b)+"+"+(ax-bx)+"i"); |

**FOCP II Lab Manual(CSL108) 2021-22**

|  |
| --- |
| }  void pro(int a,int ax, int b, int bx){ System.out.println("Product:"+((a\*b)+(ax\*bx))+"+"+((b\*ax)+(a\*bx))+"i");  }  }  class ria{  public static void main(String args[]){ Complex c1=new Complex(); int a,ax,b,bx;  Scanner scan=new Scanner(System.in); System.out.print("Enter real 1:"); a=scan.nextInt(); System.out.println("Enter imag 1:"); ax=scan.nextInt(); System.out.print("Enter real 2:"); b=scan.nextInt(); System.out.println("Enter imag 2:"); bx=scan.nextInt();  c1.sum(a,ax,b,bx);  c1.sub(a,ax,b,bx);  c1.pro(a,ax,b,bx);  }  } OUTPUT: |

**FOCP II Lab Manual(CSL108) 2021-22**

|  |
| --- |
| 1. **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:**   * 1. **Non-parametrized and parameterized constructors are defined.**   2. **Get and set methods are defined for all the instance variables.**   3. **Distance function is defined to calculate the distance between two points.**   4. **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.util.\*; import java.lang.\*; class Practical4Q5{  public static void main(String []args){ float x1, y1, x2, y2;  TwoPoint tp = new TwoPoint(); Scanner obj = new Scanner(System.in);  System.out.println("Enter the coordinates of first point (x1, y1): "); x1 = obj.nextFloat();  y1 = obj.nextFloat();  System.out.println("Enter the coordinates of Second point (x2, y2): "); x2 = obj.nextFloat();  y2 = obj.nextFloat();  TwoPoint tp1 = new TwoPoint(x1,y1,x2,y2); tp1.getDistance();  tp1.Display();  }  }  import java.util.\*; import java.lang.\*;  class TwoPoint{  float x1, x2, y1, y2; |

**FOCP II Lab Manual(CSL108) 2021-22**

|  |
| --- |
| double distance; TwoPoint(){  }  float getx1(){ return x1;  }  void setx1(){ this.x1 = x1;  }  float gety1(){ return y1;  }  void sety1(){ this.y1 = y1;  }  float getx2(){ return x2;  }  void setx2(){ this.x2 = x2;  }  float gety2(){ return y2;  }  void sety2(){ this.y2 = y2;  }  TwoPoint(float x1, float y1, float x2, float y2){ this.x1 = x1;  this.y1 = y1; this.x2 = x2; this.y2 = y2;  }  void getDistance(){  double point1 = Math.pow((x2-x1),2); |

**FOCP II Lab Manual(CSL108) 2021-22**

|  |
| --- |
| double point2 = Math.pow((y2-y1),2); distance = Math.pow((point1 + point2),0.5);  }  void Display(){  System.out.println("Distance between two points = " + (String.format("%.2f",distance)));  }  }  OUTPUT: |

|  |
| --- |
| **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?**   **ANSWER:** Class is a blueprint or template from which objects are created. An object is a real-world entity such as a pen, laptop, mobile, bed, keyboard, mouse, chair, etc. Class is a group of similar objects. An object is a physical entity.   1. **What is constructor chaining?**   **ANSWER:** Constructor chaining is the process of calling one constructor from another constructor with respect to the current object.   1. **What is a No-argument constructor?**   **ANSWER:** A constructor that does not accept any arguments.   1. **What happens if you keep return type for a constructor?**   **ANSWER:** If we add a return type to a constructor, then it will become a method of the class.   1. **What is the use of private constructor?**   **ANSWER:** 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.   1. **Can we use this() in a method?**   **ANSWER:** This() can be used to invoke the current class constructor. this can be passed as an argument in the method call. This can be used to return the current class instance from the method.   1. **Can we define a method with same name of class?**   **ANSWER:** Yes, It is allowed to define a method with the same name as that of a 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));  }  }   * 1. false   2. true   3. 1   4. Compilation Error   **ANSWER:** b) False |

## PRACTICAL NO: 5

|  |
| --- |
| **Student Name and Roll Number:** Arjun Bhardwaj 21csu211 |
| **Semester /Section:** 2nd Semester / CSE-D |
| **Date:** 26/04/2022 |
| **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.**  import java.util.\*; class Practical5Q1{  public static void main(String []args){ Square on = new Square(); on.display();  on.display1();  }  }  class Shape{  void display(){  System.out.println("This is This is Shape");  }  } |

**FOCP II Lab Manual(CSL108) 2021-22**

|  |
| --- |
| class Rectangle extends Shape{ void display1(){  System.out.println("This is Rectangular Shape");  }  }  class Square extends Rectangle{ void display2(){  System.out.println("Square is a Rectangle");  }  }  class Circle extends Shape{ void display3(){  System.out.println("This is Circular Shape");  }  }  OUTPUT:    **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 tires 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.** |

**FOCP II Lab Manual(CSL108) 2021-22**

|  |
| --- |
| **Definition of Done:**  **DoD 1: Each class definition is stored in its own .java file.**  **DoD 2: A switch statement is used for identifying the appropriate class for which the object is to be invoked**.  import java.util.\*; class Practical5Q2{  public static void main(String []args){  int n;  Vehicle vechile = new Vehicle(); Car car = new Car();  Bike bike = new Bike();  Scanner obj = new Scanner(System.in); System.out.println("Enter the number of tyres in the Vehicle :"); n = obj.nextInt();  switch(n){  case 2: bike.display(); break;  case 4: car.display(); break;  default: vechile.display(); break;  }  }  }  class Vehicle{ void display(){  System.out.println("This is a Vehicle");  }  }  class Car extends Vehicle{ void display(){  System.out.println("This is a Car"); |

**FOCP II Lab Manual(CSL108) 2021-22**

|  |
| --- |
| }  }  class Bike extends Vehicle{ void display(){  System.out.println("This is a Bike");  }  }  OUTPUT: |

**FOCP II Lab Manual(CSL108) 2021-22**

|  |
| --- |
| 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.**  import java.util.\*; class Boxx{  double height; double length; double width; Boxx(Boxx b){  this.height=b.height; this.length=b.length; this.width=b.width;  }  Boxx(double height,double length,double width){ this.height=height;  this.length=length; this.width=width;  }  Boxx(){  height=55.5; length=70.0; width=30.5;  }  void get(){  System.out.println("Length,Width,Height:"+length+width+height);  }  void set(){  Scanner s=new Scanner(System.in); System.out.println("Enter length:"); length=s.nextDouble(); |

**FOCP II Lab Manual(CSL108) 2021-22**

|  |
| --- |
| System.out.println("Enter width:"); width=s.nextDouble(); System.out.println("Enter height:"); height=s.nextDouble();  }  void vol(){  System.out.println("Volume:"+(length\*width\*height));  }  }  class Boxweight extends Boxx{ double weight; Boxweight(Boxweight bw){  this.weight=bw.weight;  }  Boxweight(double weight){ this.weight=weight;  }  Boxweight(){  super(); weight=30.0;  }  void getw(){  System.out.println("Weight:"+weight);  }  void setw(){  Scanner s=new Scanner(System.in); System.out.println("Enter weight:"); weight=s.nextDouble();  }  void disweight(){  System.out.println("Weight:"+weight);  }  }  class BoxweightDemo{  public static void main(String args[]){ Boxweight bw=new Boxweight();  bw.vol(); |

**FOCP II Lab Manual(CSL108) 2021-22**

|  |
| --- |
| bw.disweight();  }  }  Output:    **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 {  .....  .....  } |

**FOCP II Lab Manual(CSL108) 2021-22**

|  |
| --- |
| **Question Bank**   1. **What is the use of the super keyword?**   **ANSWER:** The super keyword refers to super class (parent) object. It is used to call superclass methods, and to access the superclass constructor.   1. **What is multi-level inheritance?**   **ANSWER:** The multi-level inheritance includes the involvement of at least two or more than two classes. One class inherits the features from a parent class and the newly created sub- class becomes the base class for another new class.   1. **What is the usage of inheritance?**   **ANSWER:** Inheritance is used to use the existing features of class. It is used to achieve runtime polymorphism. |
| **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();  }  } |



|  |
| --- |
| **ANSWER:** 1  2  3  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();  }  ANSWER: Class A Constructor  Class B Constructor Class C Constructor |

### PRACTICAL NO. 6

|  |
| --- |
| **Student Name and Roll Number: Arjun Bhardwaj 21csu211** |
| **Semester /Section: 2nd Sem/ D** |
| **Date:** |
| **Faculty Signature:** |

|  |
| --- |
| **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. a) Write a program in java to check if a class can extends another class and/ can implement one and more than one interface.   1. Write a program in java to check if an interface can extend other interface. 2. Write a program in java to check if an interface can also extend multiple interfaces.   CODE:  interface Aims{} interface Manipal{}  interface Artemis extends Manipal,Aims{} class Hospital{}  class Apollo extends Hospital implements Aims,Manipal{}  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. |

|  |
| --- |
| CODE:  import java.util.\*; interface GeoObj{  public double getper(); public double getarea();  }  interface Resizeable{  public void resize(int percent);  }  class Circle implements GeoObj{ double radius=1.0;  Circle(double radius){ |

|  |
| --- |
| this.radius=radius;  }  public double getper(){ return (2\*3.14\*radius);  }  public double getarea(){  return (3.14\*radius\*radius);  }  }  class ResizeableCircle extends Circle implements Resizeable{ int percent;  ResizeableCircle(double radius){ super(radius);  }  public void resize(int percent){  System.out.println("Old Radius:"+radius); radius=radius+(radius\*percent)/100; |

|  |
| --- |
| System.out.println("New size:"+radius);  }  }  class Sixcircle{  public static void main(String[] args) { Scanner sc=new Scanner(System.in); double r=sc.nextDouble();  int p=sc.nextInt();  ResizeableCircle r1=new ResizeableCircle(r); r1.resize(p);  }  } OUTPUT:    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.  CODE:  import java.util.\*; import java.lang.Math; abstract class Shape{  abstract void rectar(int l, int b); abstract void cirar(int r); abstract void squar(int s);  }  class Area extends Shape{ void rectar(int l, int b){  System.out.println("Area of Rectangle:"+(l\*b));  }  void cirar(int r){  System.out.println("Area of Circle:"+(3.14\*(Math.pow(r,2))));  }  void squar(int s){  System.out.println("Area of Square:"+s\*s);  }  }  class Six{  public static void main(String[] args) { Shape s=new Area(); s.rectar(5,7);  s.squar(5);  s.cirar(2);  }  }  OUTPUT: |

|  |
| --- |
|  |
| **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. There can be only abstract methods in the Java interface, not method body. It is used to achieve abstraction and multiple 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); |

|  |
| --- |
| }  }   1. 0 2. 2 3. **4** 4. None of the mentioned 5. Which of the following package stores all the standard java classes?    1. lang    2. java    3. **util**    4. 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?   **If we make an interface final, we cannot implement its methods which defies the very purpose of the interfaces. Therefore, we cannot make an interface final in Java.**   1. Can an abstract class implement an interface?   **Yes an abstract class implement an interface.**   1. Can you declare an interface method static?   **Yes we can declare an interface method static.**   1. What is the difference between abstract class and interface?   **Abstract class can have both abstract and non-abstract methods whereas interface can only have abstract methods. Abstract class cannot have multiple inheritance whereas interface can have it.**   1. When can an object reference can be cast to an interface reference ?   **If we implement an interface and provide body to its methods from a class and hold object of the that class using the reference variable of the interface then the object reference can be casted to an interface reference.** |

**Experiment No: 7**

|  |
| --- |
| **Student Name and Roll Number: Arjun Bhardwaj 21csu211** |
| **Semester /Section: 2nd Sem/D** |
| **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.  **CODE:**  package exercises.java; public class PackageDemo{  public void display(){  System.out.println("PackageDemo executed.");  }  }  package exercises.java;  public class PackageDemoDriver{  public static void main(String[] args) { PackageDemo p=new PackageDemo(); p.display();  }  } |

|  |
| --- |
| OUTPUT:    2. Write a java program outside the above defined package that imports the package PackageDemo and calls its display() function.  CODE:  package exercises.java; public class PackageDemo{  public void display(){  System.out.println("PackageDemo executed.");  }  }  import exercises.java.PackageDemo; class Seven{  public static void main(String[] args) { PackageDemo p2=new PackageDemo(); p2.display();  }  } OUTPUT: |

|  |
| --- |
|  |
| **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;   1. 0 2. 1 3. 2 4. 0 1 2 |
| **Question Bank**   1. What are packages ? what is use of packages ?   **Java package is a group of similar types of classes, interfaces and sub-*packages*. Packages are used for:**   * + **Preventing naming conflicts. For example there can be two classes with name Employee in two packages.**   + **Making searching/locating and usage of classes, interfaces, enumerations and annotations easier**   + **Providing controlled access: protected and default have package level access control. A protected member is accessible by classes in the same package and its subclasses. A default member (without any access specifier) is accessible by classes in the same package only.**   + **Packages can be considered as data encapsulation (or data-hiding).**  1. What is difference between importing "java.applet.Applet" and "java.applet.\* "?   **In the former, we’ve only imported the Applet class from applet sub package while in the latter one we’ve imported all the classes from the applet sub-package.**   1. What do you understand by package access specifier?   **Access modifiers (or access specifiers) are keywords in object-oriented languages that set the accessibility of classes, methods, and other members. Access modifiers are a specific part of programming language syntax used to facilitate the encapsulation of components**   1. By default,all program import the java.lang package. True/False ?   **True.** |

|  |
| --- |
| 1. Java compiler stores the .class files in the path specified in CLASSPATH environmental variable. True/False ?   **True.**   1. User-defined package can also be imported just like the standard packages True/False ?   **True.**   1. A \_**package** is used to separate the hierarchy of the class while declaring an Import statement. 2. All standard classes of Java are included within a package called .   **Java** |

**PRACTICAL NO.8**

|  |
| --- |
| **Student Name and Roll Number: Arjun Bhardwaj 21csu211** |
| **Semester /Section: 2nd Sem / D** |
| **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  **CODE:**  class CheckargException extends Exception{ CheckargException(String s){ |

|  |
| --- |
| super(s);}  }  class Check{  public static void main(String[] args) { int arr[]=new int[5];  int sum=0; try{  for (int i=0;i<args.length;i++) { arr[i]=Integer.parseInt(args[i]);} if (args.length<5){  throw new CheckargException("Less than 5 arguments.");} else{  for (int j=0;j<args.length;j++ ) { sum=sum+Integer.parseInt(args[j]);}  }  System.out.println("Sum:"+sum);} catch(CheckargException e){ System.out.println(e);}  }  } |

|  |
| --- |
| OUTPUT:    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.  import java.io.\*; class Excep{  public static void main(String[] args) { try{  throw new IOException("Exception occurred.");} catch(IOException e){  System.out.println(e) System.out.println("Exception handled");}  finally{System.out.println("Yess...I was here!!"); }  }  } OUTPUT: |

|  |
| --- |
|  |
| **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. |
| **Question Bank**   1. When does Exceptions in Java arises in code sequence?    1. **Run Time**    2. Compilation Time    3. Can Occur Any Time    4. None of the mentioned 2. Which of these keywords must be used to monitor for exceptions?    1. **try**    2. finally    3. throw    4. catch 3. Which of these keywords must be used to handle the exception thrown by try block in some rational manner?    1. try    2. finally |

1. throw
2. **catch**
3. Which of these keywords is used to manually throw an exception?
   1. try
   2. finally
   3. **throw**
   4. catch

**Flipped Practicals**

1. What will be the output of the following Java program?

1. Hello
2. **World**
3. HelloWorld
4. Hello World

2. What will be the output of the following Java program?

**class** exception\_handling

{

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

{

**try**

{

**class** exception\_handling

{

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

{

**try**

{

System.out.print("Hello" + " " + 1 / 0);

}

**catch**(ArithmeticException e)

{

System.out.print("World");

}

}

}

|  |
| --- |
| **int** a, b; b = 0;  a = 5 / b; System.out.print("A");  }  **catch**(ArithmeticException e)  {  System.out.print("B");  }  }  }   1. A 2. **B** 3. Compilation Error 4. Runtime Error |

**PRACTICAL NO. 9**

|  |
| --- |
| **Student Name and Roll Number: Arjun Bhardwaj 21csu211** |
| **Semester /Section: 2nd Sem / D** |
| **Date:** |
| **Faculty Signature:** |

|  |
| --- |
| **Objective**  To familiarize the students with File handling. |
| **Program Outcome**  The students will learn the concept of files in Java. |
| **Problem Statement**   1. Write a program to check if the file exist is a file or directory. 2. Write a program to change the file permissions. 3. Write a program to perform simple read and write operation into file.   **CODE:**  import java.io.\*; import java.util.\*; class Fileh{  public static void main(String[] args) {  File f1=new File("series.txt"); |

|  |
| --- |
| try{ if(f1.createNewFile()){  System.out.println("File "+f1.getName()+"created");  }  else {  System.out.println("File exists");  }  }  catch(Exception e){ e.printStackTrace();  }  try{  FileWriter fw=new FileWriter("series.txt"); BufferedWriter bw=new BufferedWriter(fw); Scanner sc =new Scanner(System.in); System.out.println("Enter string:"); while(sc.hasNext()){  bw.write(sc.nextLine()); bw.newLine();  } |

|  |
| --- |
| bw.close();  }  catch(Exception e){ e.printStackTrace();  }  try{  FileReader fr=new FileReader("series.txt"); BufferedReader br=new BufferedReader(fr); String sr;  while((sr=br.readLine())!=null){ System.out.println(sr);  }  br.close();  }  catch(Exception e){ System.out.println("Exception");  }  }  } |

|  |
| --- |
| OUTPUT:      3. 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.  **CODE:**  import java.util.\*; import java.io.\*; class Nine{  public static void main(String[] args) { File f=new File("report.txt");  try{ if(f.createNewFile()){  System.out.println("New File "+f.getName()+"created.");  }  else{  System.out.println("File exists."); |

|  |
| --- |
| }  }  catch(Exception e){ e.printStackTrace();  }  try{  FileWriter fw=new FileWriter("report.txt"); BufferedWriter bw=new BufferedWriter(fw); Scanner sc=new Scanner(System.in);  System.out.println("Enter Serial no.,Name,CGPA,Grade:"); while(sc.hasNext()){  bw.write(sc.nextLine()); bw.newLine();  }  bw.close();  }  catch(Exception e){ e.printStackTrace();  }  try{ |

|  |
| --- |
| FileReader fr=new FileReader("report.txt"); BufferedReader br=new BufferedReader(fr); String s;  while((s=br.readLine())!=null){ System.out.println(s);  }  br.close();  }  catch(Exception e){ System.out.println("Exception.");  }  }  }  **OUTPUT:** |

**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**

**Question Bank**

1. Which of these exception is thrown in cases when the file specified for writing is not found?
   1. IOException
   2. FileException
   3. **FileNotFoundException**
   4. FileInputException
2. Which of these methods are used to read in from file?
   1. get()
   2. **read()**
   3. scan()
   4. readFileInput()

import java.io.File; // Import the File class

File myObj = new File("filename.txt"); // Specify the filename

1. Which of these values is returned by read() method is end of file (EOF) is encountered?
   1. 0
   2. 1
   3. **-1**
   4. Null

**Flipped Practicals**

1. What will be the output of the following Java program?

Note: inputoutput.java is stored in the disk.

1. true
2. false
3. **prints number of bytes in file**
4. prints number of characters in the file
5. 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. {

1. String obj = "abc";
2. **byte** b[] = obj.getBytes();
3. ByteArrayInputStream obj1 = **new** ByteArrayInputStream(b);
4. **for** (**int** i = 0; i < 2; ++ i)

10. {

1. **int** c;
2. **while**((c = obj1.read()) != -1)
3. {
4. **import** java.io.\*;
5. **class** filesinputoutput
6. {
7. **public static void** main(String args[])
8. {
9. InputStream obj = **new** FileInputStream("inputoutput.java");
10. System.out.print(obj.available());

8. }

9. }

|  |
| --- |
| 1. **if**(i == 0) 2. { 3. System.out.print(Character.toUpperCase((**char**)c)); 4. obj2.write(1);   18. }  19. }  20. System.out.print(obj2);  21. }  22. }  23. }   1. AaBaCa 2. ABCaaa 3. AaaBaaCaa 4. **AaBaaCaaa** |