**Orchid International College**

Bijay chowk, Gaushala

**Department Of Information Technology**

****

**LAB MANUAL**

**of**

**OOP with Java**

**Laboratory Works:**

* Basic Programs and using Decision Making statements
* Iterative Statements/Loop
* Array
* Using Methods
* Object Oriented Concept & Inheritance
* Exception Handling
* String Handling
* File Handling
* Generics
* Multithreading

**LAB 1: Basic Programs and using Decision Making statements**

**Introduction:**

Java is a high-level, object-oriented programming language known for its portability, performance, and security. It's widely used for developing mobile applications, web applications, enterprise software, and more. Java programs are compiled into bytecode, which can run on any system equipped with a Java Virtual Machine (JVM), making Java platform-independent.

**Objectives:**

1.Understanding Basic Java Syntax

2.Implementing Decision-Making Statements

3.Developing Problem-Solving Skills

**1. Write a program to input radius of circle and find its area.**

Source code:

package Lab1;

import java.util.\*;

public class RadiusCircle {

public static void main(String[] args) {

Scanner ob=new Scanner(System.in);

System.out.println("Enter the radius of a circle: ");

int r=ob.nextInt();

double area=3.14\*r\*r;

System.out.println("The area of circle is:"+area);

}

}

**2. Write a program to input a character and check vowel/ consonant using switch.**

Source code:

package Lab1;

import java.util.\*;

public class VowelConsonant {

public static void main(String[] args) {

Scanner ob=new Scanner(System.in);

System.out.println("Enter the character");

char ch=ob.next().charAt(0);

switch(ch){

case 'a':

case 'e':

case 'i':

case 'o':

case 'u':

System.out.println("Vowel");

break;

default:

System.out.println("Consonant");

}

}

}

**Tasks:**

a) Write a program to input 3 numbers and find the sum of their cubes. For eg: If number are 1,2,3 sum=36.

b) Write a program to input a number and find its square root.

c) Write a program to input a number and check odd/even.

d) Write a program to input 3 numbers and find the greatest one.

e) Create a menu driven program to add/subtract/multiply/divide two numbers using switch.

**LAB 2 : Iterative Statements/loop**

**Introduction:**

Iterative statements, also known as loops, are used in Java to execute a block of code repeatedly based on a condition. These loops are essential for performing repetitive tasks efficiently. Java provides several types of loops: for, while, and do-while.

**Objective:**

1.Understanding the syntax and usage of the primary iterative statements (for, while, do-while)

2.Implementation of efficient repetition

**1. Write a program to print the numbers between 100 to 200 which are exactly divisible by 5.**

Source Code:

package Lab2;

import java.util.\*;

public class Divisible {

public static void main(String[] args) {

for(int i=100;i<200;i++){

if(i%5==0){

System.out.println(i);

}

}

}

}

**2.** **Write a program to find the sum of first N natural numbers.**

Source Code:

package Lab2;

import java.util.\*;

public class NaturalNumbers {

public static void main(String[] args) {

Scanner ob=new Scanner(System.in);

System.out.println("Enter the number of natural number:");

int N =ob.nextInt();

int sum=0;

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

sum+=i;

}

System.out.println("The sum of "+N+" is:"+sum);

}

}

**3. Write a program to display the series:**

**a) \***

**\*\***

**\*\*\***

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

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

Source Code:

package Lab2;

public class Star {

public static void main(String[] args) {

int rows=5;

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

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

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

}

System.out.println();

}

}

}

**Tasks:**

1. a program to input a number and find its reverse.

b) Write a program to display the series:

i) **\*\*\*\*\***

**\*\*\***

**\***

ii)1 2 3 4 5

1 2 3 4

1 2 3

1 2

1

c) Write a program to print all prime numbers between 100 and 200.

d) Write a program to print all palindrome numbers 100 to 500.

**LAB 3: Array**

**Introduction:**

Array is a collection of homogeneous(similar) type of data. Each element of an array can be accessed using index number which always starts from ‘0’(zero). The types of array are:

a)One Dimensional Array b)Multi-Dimensional Array

**Objectives:**

1.Learn the fundamentals of arrays, including their definition, declaration, initialization, and accessing elements, to store and manage collections of data efficiently.

2. Recognize the advantages of using arrays for structured data storage, enabling efficient memory management and fast access to elements.

3.Solving complex problems

1. **Create an array to store 10 numbers and find the sum of only those numbers whose last digit is 1.**

Source Code:

package Lab3;

import java.util.\*;

public class ArraySum {

public static void main(String[] args) {

Scanner ob=new Scanner(System.in);

int s=0;

int[]a=new int[10];

System.out.println("Enter 10 numbers: ");

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

a[i]=ob.nextInt();

int r=a[i]%10;

if(r==1){

s=s+a[i];

}

}

System.out.println("sum="+s);

}

}

1. **Create an array to store 10 numbers in an array and find the greatest and smallest number.**

Source Code:

package Lab3;

import java.util.\*;

public class GreatestSmallest {

public static void main(String[] args) {

Scanner ob=new Scanner(System.in);

System.out.println("Enter the size of array");

int n=ob.nextInt();

int a[]=new int[10];

System.out.println("Enter the size of numbers: ");

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

a[i]=ob.nextInt();

}

int max=a[0];

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

if(a[i]>max){

max=a[i];

}

}

System.out.println("Greatest no="+max);

int min=a[0];

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

if(a[i]<min){

min =a[i];

}

}

System.out.println("Smallest no= "+min);

}

}

**Tasks:**

1. Create an array to store 10 numbers and sort them in ascending order.
2. Write a program to print the diagonal elements of a matrix.
3. Write a program to find sum of elements of a 3x4 matrices.

**LAB 4: Using Methods**

**Introduction:**

Methods are a fundamental concept in object-oriented programming (OOP) . They encapsulate code into reusable blocks, promoting modularity, code reusability, and readability. A method in Java is a collection of statements grouped together to perform a specific operation. Methods help to break down complex problems into smaller, manageable tasks.

**Objective:**

1.Learn the basic structure of methods in Java, including access modifiers, return types, method names, and parameters, to create well-defined and reusable code blocks.

2. Master the concept of method overloading to create multiple methods with the same name but different parameter lists, thereby increasing code flexibility and readability.

3.Promote code reusability.

1. **Create a method void sum(int ...a) that finds sum of all numbers provided as arguments.**

Source Code:

package Lab4;

public class VoidSum {

public static void main(String[] args) {

sum(3,4,5);

}

static void sum(int...a){

int s=0;

for(int m:a){

s=m+s;

}

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

}

}

1. **Create a method int max(int a[]) that returns the largest number in an array.**

Source Code:

package Lab4;

public class MaxReturn {

public static void main(String[] args) {

int b[]={4,3,8,9,6};

int s=max(b);

System.out.println("Largest number is"+s);

}

static int max(int a[]){

int s=a[0];

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

if(a[i]>s){

s=a[i];

}

}

return s;

}

}

**Tasks:**

1. Create a method void check(int n) that checks whether given number is odd/even.
2. Create a method void print(int a, int b)that prints all numbers between a and b.(exclude a and b).
3. Create a method int max(int a[]) that returns the largest number in an array.

**LAB-5 (Object Oriented Concept & Inheritance)**

**Introduction:**

Object-Oriented Programming (OOP) is a programming paradigm centered around objects, which are instances of classes. It emphasizes the concepts of encapsulation, inheritance, polymorphism, and abstraction, aiming to enhance modularity, reusability, and maintainability of code.

Inheritance is a fundamental feature of OOP that allows a new class (subclass or derived class) to inherit attributes and methods from an existing class (superclass or base class). This promotes code reuse and establishes a natural hierarchy between classes.

**Objectives:**

1.Understanding OOP principles.

2. Learn to define classes and create objects in Java, encapsulating data and behavior within well-defined structures to enhance code modularity and reuse.

3.Utilize inheritance for code reusability.

1. **Create a class Employee with the instance variables name, age, salary. Create getters and setters. Create a test class and create an object of Employee class, then set and print the employee details.**

Source Code:

package Lab5;

class Employee{

private String name;

private int age,salary;

public String getName(){

return name;

}

public void setName(String name){

this.name=name;

}

public int getAge(){

return age;

}

public void setAge(int age){

this.age=age;

}

public int getSalary(){

return salary;

}

public void setSalary(int salary){

this.salary=salary;

}

}

public class EmployeeDemo {

public static void main(String[] args) {

Employee e=new Employee();

e.setName("Aayusha");

e.setAge(19);

e.setSalary(100000);

System.out.println("Name="+e.getName());

System.out.println("Age="+e.getAge());

System.out.println("Salary="+e.getSalary());

}

}

1. **Create an interface Calculation with two methods void add(int a,int b) and int multiply(int a,int b). Create a class CalculationDemo which is a subclass of Calculation. Provide a test class.**

Source Code:

package Lab5;

interface Calculation{

void add(int a, int b);

int multiply(int a, int b);

}

class CalculationDemo implements Calculation{

public void add(int a, int b){

int s=a+b;

System.out.println("Addition="+s);

}

public int multiply(int a,int b){

return a\*b;

}

}

public class Calculate {

public static void main(String[] args) {

CalculationDemo ob=new CalculationDemo();

ob.add(5,3);

int m=ob.multiply(2,6);

System.out.println("Multiply="+m);

}

}

**Task:**

1. Create a class Student with the instance variables name, age, faculty. Create getters and setters. Create a test class and create 3 student objects set the details and display the details of youngest student.
2. WAP to print the area of 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 rectangle. Length and breadth of rectangle are entered through the keyboard.
3. Create a class Person with firstName, lastName, age. Create a parameterized constructor to initialize the variables and a method void showInfo() to display person details. Create a class Student which is sub class of Person and add the field String faculty Override the method showInfo() to display faculty. Create a class Teacher which is a sub class of Person and add the field double salary. Override the method showInfo() to display salary. Provide a test class.

**LAB 6: Exception Handling**

**Introduction:**

Exception handling is a crucial aspect of Java programming that allows developers to manage and respond to unexpected or erroneous situations during program execution. An exception is an event that disrupts the normal flow of a program's instructions. Java provides robust mechanisms for identifying, managing, and recovering from such exceptions.

**Objectives:**

1.Understand the distinction between checked and unchecked exceptions in Java.

2.Learn toImplement ‘try-catch’ Blocks;

3. Utilize ‘finally’ Blocks.

**1. Write a program to input two integers, divide the first with second and print the result. Handle ArithmeticException.**

Source Code:

package Lab6;

import java.util.\*;

public class Integers {

public static void main(String[] args) {

Scanner ob=new Scanner(System.in);

System.out.println("Enter two integers: ");

int a=ob.nextInt();

int b=ob.nextInt();

try

{

int c=a/b;

System.out.println(c);

}

catch(ArithmeticException ex){

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

}

}

}

**Tasks:**

a) Write a program to handle ArrayIndexOutOfBoundsException.

b) Write a program to create your own exception class.

c) Write a program to show the use of "throws" keyword.

**LAB 7: String Handling**

**Introduction:**

String handling in Java, within the context of Object-Oriented Programming (OOP), involves the utilization of the String class and its methods to manage and manipulate textual data. In Java, strings are treated as objects of the String class, which provides various built-in methods for performing operations such as concatenation, substring extraction, comparison, and more.

**Objectives:**

1.Gain proficiency in using built-in methods of the String class to perform essential operations.

2.Students should be able to apply string handling techniques to solve practical problems in Java applications, including text manipulation, input validation, formatting, and output generation.

**1**. **WAP to input name of 5 cities and sort them in alphabetical order.**

Source Code:

package Lab7;

import java.util.\*;

public class Sorting {

public static void main(String[] args) {

Scanner ob = new Scanner(System.in);

System.out.println("Enter names of 5 cities: ");

String cities[] = new String[5];

// Read city names

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

cities[i] = ob.next();

}

// Sorting using bubble sort

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

for(int j = i + 1; j < 5; j++) {

if(cities[i].compareTo(cities[j]) > 0) {

String temp = cities[i];

cities[i] = cities[j];

cities[j] = temp;

}

}

}

// Display sorted array

System.out.println("String array after sorting:");

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

System.out.println(cities[i] + " ");

}

}

}

**Tasks:**

a) WAP to input a word and display its reverse then check whether it is palindrome or not.

b) WAP to count no. of vowels in a given string.

c) WAP to display only those names which start with "R" in an array of 5 names.

d) WAP to input name of 5 countries and display only those which end with a vowel.

**Lab 8: File Handling**

**Introduction:**

File handling refers to the process of creating, reading, writing, and managing files on a storage device. It is an essential aspect of many applications, enabling persistent data storage, retrieval, and manipulation. In Java, file handling is facilitated by classes in the java.io package, which provide a wide range of functionalities to interact with files and directories efficiently.

**Objective:**

1. To enable efficient data management and storage, ensure data persistence, facilitate data exchange, and support robust application functionality.

**1.** **WAP to store a line in tu.txt file and copy it into another ku.txt file and read from another file and display it.**

Source Code:

package Lab8;

import java.io.\*;

import java.util.\*;

public class CopyFile {

public static void main(String[] args) {

Scanner ob= new Scanner(System.in);

System.out.println("Enter a line of text");

String s=ob.nextLine();

byte b[]=s.getBytes();

try{

FileOutputSystem fout=new FileOutputSystem("tu.txt");

fout.write(b);

fout.close();

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

FileOutputStream fout1=new FileOutputStream("ku.txt");

int ch;

while((ch=fin.read())!=-1){

fout1.write(ch);

}

fout1.close();

fin.close();

FileInputStream fin1=new FileInputStream("ku.txt");

int ch1;

while((ch1=fin1.read())!=-1){

System.out.print((char)ch1);

}

fin1.close();

}

catch(Exception ex){

System.out.println(ex);

}

}

}

**Tasks:**

a) WAP to input a line of text and store it in a file. Also read the content from file to display its contents.

b) WAP to input name, address, salary of 5 employees, store them in a file. And read from the file to display the contents.

c) WAP to input name, address, post, salary of 5 employees, store them in a file. And read from the file to display the contents of only those employees whose salary is more than 50000.

d) WAP to input name, address, post, salary of 5 employees, store them in a file. And read from the file to display the contents of only those employees whose address is "Kathmandu".

e) WAP to input name, address, post, salary of 5 employees, store them in a file "emp.txt". Also copy the content into another file "employ.txt".

f) Create a class Student with name, faculty, age. Create 3 student objects initialize them using parameterized constructor, and store them in a file "student.txt". Also read from the file to display the details.

**Lab 9: Generics**

**Introduction:**

It is a style of computer programming in which programs are written in terms of “type to-be-specified-later”. In java, generic programming was introduced in J2SE5.

**Objective:**

1. To enhance type safety, increase code reusability, and improve readability and maintainability of code.

**1. Create a generic method that finds the average of numbers.**

Source Code:

package Lab9;

public class GenericAverage {

public static <T extends Number>void avg(T array[]){

double sum=0.0;

for (T v:array){

sum=+v.doubleValue();

}

System.out.println("Average="+(sum/array.length));

}

public static void main(String[] args) {

Integer a[]={5,6,8,3,9};

avg(a);

}

}

Tasks:

a) WAP to create a generic class with a single type parameter.

b) WAP to create a generic class with two type parameters.

c) Create a generic method to find sum of two numbers. Use bounded parameter.

d) Create a generic method to swap any two elements of an array.

e) Create a generic method to find the maximum value in an array. Use bounded parameter.

**Lab 10: Multithreading**

**Introduction:**

Multithreading is a programming technique that allows multiple threads to run concurrently within a single process, enabling more efficient use of system resources. It is widely used to perform multiple tasks simultaneously, improving the performance and responsiveness of applications, particularly in scenarios involving I/O operations, complex calculations, or real-time processing.

**Objective:**

1.By leveraging multithreading, students can enhance the throughput and efficiency of their software, making better use of multi-core processors.

1. **Create a thread which displays the numbers 1 to 10 in the interval of 1 second.**

Source Code:

package Lab10;

public class Multithreading implements Runnable {

@Override

public void run() {

try {

for (int i = 1; i <= 10; i++) {

System.out.println(i);

Thread.sleep(1000); // Sleep for 1 second

}

} catch (InterruptedException e) {

System.err.println("Thread was interrupted");

}

}

public static void main(String[] args) {

// Create a Runnable instance

Multithreading numberPrinter = new Multithreading();

// Create a Thread instance with the Runnable

Thread thread = new Thread(numberPrinter);

// Start the thread

thread.start();

}

}

**Tasks:**

a) Create two threads, the first thread should display the factors of 20 in the interval of 1 second and second thread should display the factors of 25 in the interval of 1 and half second.

b) Create two threads, the first thread should display the name of any 5 birds in the interval of 1 second and second thread should display the name of any 5 animals in the interval of 1 and half second.

c) Write a program to illustrate the concept of synchronization.

d) Create two threads, one thread to display 1 to 50 and another thread to display 100 to 50. Assign maximum priority to the first thread and minimum priority to the second thread.