Programming in Java LAB

**PAPER CODE : CIC-258**

Faculty Name : Mr. Anupam Kumar

Name : Amit Singhal

Enrollment No. : 11614802722

Branch : Computer Science & Engg.

Semester | Group : 4C6



MAHARAJA AGRASEN INSTITUTE OF TECHNOLOGY

PSP Area, Plot No. 1, Sector-22, Rohini, Delhi-110086

LAB Assessment Sheet

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| S.No. | **Experiment** | **M** | **A** | **R** | **K** | **S** | **Total** | **Date of** | **Date of** | **Signature** |
| **Name** | **R1** | **R2** | **R3** | **R4** | **R5** | **Marks** | **Perf.** | **Check.** |
|  |  |  |  |  |  |  |  |  |  |  |
|  | **LAB\_01** |  |  |  |  |  |  |  |  |  |
| 1. | Welcome\_Message |  |  |  |  |  |  | 01-02-24 | 08-02-24 |  |
| 2. | ASCII Code of character |  |  |  |  |  |  | 01-02-24 | 08-02-24 |  |
| 3. | Sum of 2 Integers |  |  |  |  |  |  | 01-02-24 | 08-02-24 |  |
| 4. | Swap (using Bitwise) |  |  |  |  |  |  | 01-02-24 | 08-02-24 |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  | **LAB\_02** |  |  |  |  |  |  |  |  |  |
| 5. | Character\_Order |  |  |  |  |  |  | 08-02-24 | 15-02-24 |  |
| 6. | Colour\_Code |  |  |  |  |  |  | 08-02-24 | 15-02-24 |  |
| 7. | Even\_Numbers |  |  |  |  |  |  | 08-02-24 | 15-02-24 |  |
| 8. | Floyds\_Format |  |  |  |  |  |  | 08-02-24 | 15-02-24 |  |
| 9. | Palindrome\_Check |  |  |  |  |  |  | 08-02-24 | 15-02-24 |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  | **LAB\_03** |  |  |  |  |  |  |  |  |  |
| 10. | ASCII to Char |  |  |  |  |  |  | 15-02-24 | 22-02-24 |  |
| 11. | Reverse 2D Array |  |  |  |  |  |  | 15-02-24 | 22-02-24 |  |
| 12. | Stack using LL |  |  |  |  |  |  | 15-02-24 | 22-02-24 |  |
| 13. | Queue using LL |  |  |  |  |  |  | 15-02-24 | 22-02-24 |  |
| 14. | Tokenizer |  |  |  |  |  |  | 15-02-24 | 22-02-24 |  |
| 15. | Area\_Calculator |  |  |  |  |  |  | 15-02-24 | 22-02-24 |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  | **LAB\_04** |  |  |  |  |  |  |  |  |  |
| 16. | Box\_Volume |  |  |  |  |  |  | 22-02-24 | 29-02-24 |  |
| 17. | ‘This’ use |  |  |  |  |  |  | 22-02-24 | 29-02-24 |  |
| 18. | Instance\_Counter |  |  |  |  |  |  | 22-02-24 | 29-02-24 |  |
| 19. | Cube\_Calculator |  |  |  |  |  |  | 22-02-24 | 29-02-24 |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  | **LAB\_05** |  |  |  |  |  |  |  |  |  |
| 20. | Method\_Overriding |  |  |  |  |  |  | 22-02-24 | 29-02-24 |  |
| 21. | Simple\_Inheritance |  |  |  |  |  |  | 22-02-24 | 29-02-24 |  |
| 22. | MultiLevel\_Inheritance |  |  |  |  |  |  | 22-02-24 | 29-02-24 |  |
| 23. | ‘super’ Keyword |  |  |  |  |  |  | 22-02-24 | 29-02-24 |  |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **S.No.** | **Experiment** | **M** | **A** | **R** | **K** | **S** | **Total** | **Date of** | **Date of** | **Signature** |
| **Name** | **R1** | **R2** | **R3** | **R4** | **R5** | **Marks** | **Perf.** | **Check.** |
| 24. | Dynamic Polymorphism &  Interface Overriding |  |  |  |  |  |  | 22-02-24 | 29-02-24 |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  | **LAB\_06** |  |  |  |  |  |  |  |  |  |
| 25. | Abstract Class |  |  |  |  |  |  | 29-02-24 | 07-03-24 |  |
| 26. | Interface |  |  |  |  |  |  | 29-02-24 | 07-03-24 |  |
| 27. | Packages |  |  |  |  |  |  | 29-02-24 | 07-03-24 |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  | **LAB\_07** |  |  |  |  |  |  |  |  |  |
| 28. | Exception Handling |  |  |  |  |  |  | 07-03-24 | 14-03-24 |  |
| 29. | Custom Exception |  |  |  |  |  |  | 07-03-24 | 14-03-24 |  |
| 30. | Applet (Hello World) |  |  |  |  |  |  | 07-03-24 | 14-03-24 |  |
| 31. | Applet (Analog Clock) |  |  |  |  |  |  | 07-03-24 | 14-03-24 |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  | **LAB\_08** |  |  |  |  |  |  |  |  |  |
| 32. | Multi-Threading |  |  |  |  |  |  | 14-03-24 | 21-03-24 |  |
| 33. | Threading #1 |  |  |  |  |  |  | 14-03-24 | 21-03-24 |  |
| 34. | Threading #2 |  |  |  |  |  |  | 14-03-24 | 21-03-24 |  |
| 35. | File Handling |  |  |  |  |  |  | 14-03-24 | 21-03-24 |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  | **LAB\_09** |  |  |  |  |  |  |  |  |  |
| 36. | Slang\_Censorer |  |  |  |  |  |  | 18-04-24 | 25-04-24 |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |

MAHARAJA AGRASEN INSTITUTE OF TECHNOLOGY



Computer Science & Engineering Department

VISION

“To be centre of excellence in education, research and technology transfer in the field of computer engineering and promote entrepreneurship and ethical values.”

MISSION

To foster an open, multidisciplinary and highly collaborative research environment for producing world-class engineers capable of providing innovative solutions to real-life problems and fulfil societal need.

Lab Exercise - 1

Program - 1

//Program to accept a String as a command-line argument and print a Welcome message:

package *LAB*\_01;

public class prg\_01\_welcome {

    public static void main(String[] args) {

        if (args.length > 0) {

            String name = args[0];

            System.out.println("Welcome " + name);

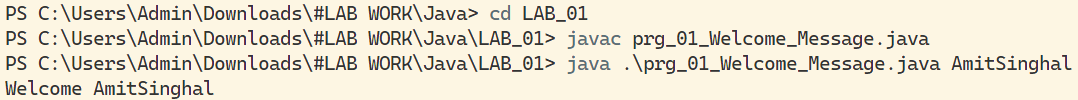
        } else {

            System.out.println("Please provide your name as a command-line argument.");

        }

    }

}



Program - 2

//Program to find ASCII code of a character:

package *LAB*\_01;

import java.util.Scanner;

public class prg\_02\_ascii {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter a character: ");

        char inputChar = scanner.next().charAt(0);

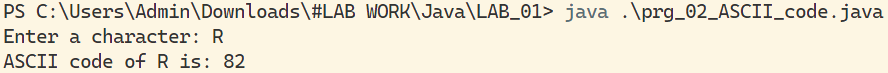
        int asciiValue = (int) inputChar;

        System.out.println("ASCII code of " + inputChar + " is: " + asciiValue);

        scanner.close();

    }

}



Program - 3

//Program to accept two integers as inputs and print their sum:

package *LAB*\_01;

import java.util.Scanner;

public class prg\_03\_sum {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter the first integer: ");

        int num1 = scanner.nextInt ();

        System.out.print("Enter the second integer: ");

        int num2 = scanner.nextInt();

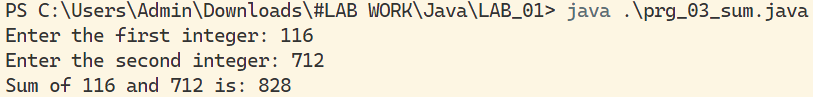
        int sum = num1 + num2;

        System.out.println("Sum of " + num1 + " and " + num2 + " is: " + sum);

        scanner.close();

    }

}



Program - 4

//Swapping two numbers using bitwise operator:

package *LAB*\_01;

public class prg\_04\_swap {

    public static void main(String[] args) {

        int a = 5;

        int b = 10;

        System.out.println("Before swapping: a = " + a + ", b = " + b);

        // Using bitwise XOR to swap values without using a temporary variable

        a = a ^ b;

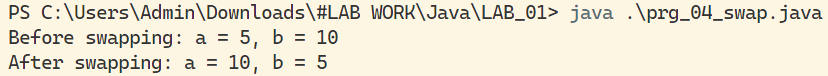
        b = a ^ b;

        a = a ^ b;

        System.out.println("After swapping: a = " + a + ", b = " + b);

    }

}



Lab Exercise - 2

//Initialize two-character variables in a program and display the characters in alphabetical order.

package *LAB*\_02;

public class prg\_05\_CharacterOrder {

    public static void main(String[] args) {

        char char1 = 'b';

        char char2 = 'a';

        System.out.println("Characters in alphabetical order:");

        if (char1 < char2) {

            System.out.println(char1 + " " + char2);

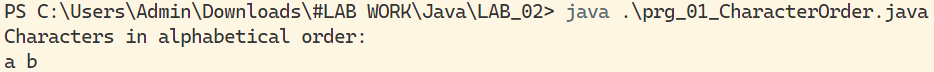
        } else {

            System.out.println(char2 + " " + char1);

        }

    }

}



Program - 5

Program - 6

//Write a program to receive a colour code from the user (an Alphabet).

package *LAB*\_02;

import java.util.Scanner;

public class prg\_06\_ColourCode {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter a colour code (Alphabet): ");

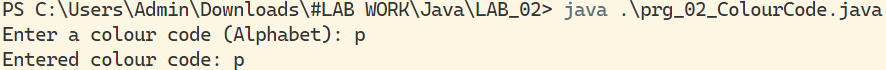
        char colorCode = scanner.next().charAt(0);

        System.out.println("Entered colour code: " + colorCode);

        scanner.close();

    }

}



Program - 7

//Write a program to print even numbers between 23 and 57.

package *LAB*\_02;

public class prg\_07\_EvenNumbers {

    public static void main(String[] args) {

        System.out.println("Even numbers between 23 and 57:");

        for (int i = 23; i <= 57; i++) {

            if (i % 2 == 0) {

                System.out.print(i);

                System.out.print(" ");

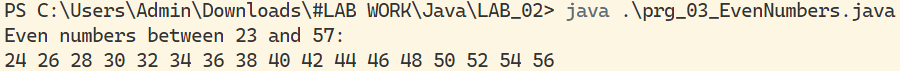
            } else

                continue;

        }

    }

}



Program - 8

//Write a program to print in Floyd's format (using for and while loop).

package *LAB*\_02;

public class prg\_08\_FloydsFormat {

    public static void main(String[] args) {

        int n = 5; // Change this value for a different number of rows

        int num = 1;

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

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

                System.out.print(num + " ");

                num++;

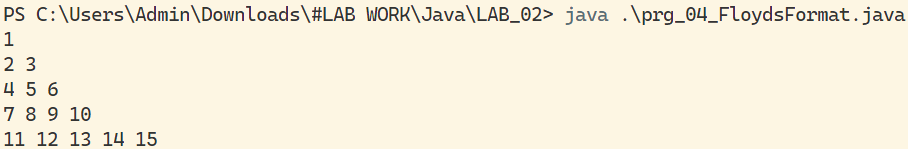
            }

            System.out.println();

        }

    }

}



Program - 9

//Write a Java program to find if the given number is palindrome or not.

package *LAB*\_02;

import java.util.Scanner;

public class prg\_09\_PalindromeCheck {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

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

        int number = scanner.nextInt();

        if (isPalindrome(number))

            System.out.println(number + " is a palindrome.");

        else

            System.out.println(number + " is not a palindrome.");

        scanner.close();

    }

    private static boolean isPalindrome(int num) {

        int originalNum = num;

        int reversedNum = 0;

        while (num > 0) {

            int digit = num % 10;

            reversedNum = reversedNum \* 10 + digit;

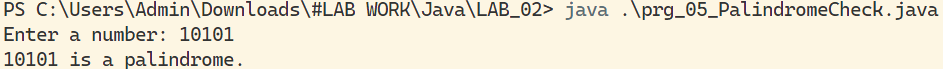
            num /= 10;

        }

        return originalNum == reversedNum;

    }

}



Lab Exercise - 3

Program - 10

//Initialize an integer array with ASCII values and print the corresponding character values in a single row.

package *LAB*\_03;

public class prg\_10\_AsciiToChar {

    public static void main(String[] args) {

        int[] asciiArray = { 65, 66, 67, 97, 98, 99 }; // Example ASCII values

        System.out.print("Corresponding characters: ");

        for (int asciiValue : asciiArray) {

            System.out.print((char) asciiValue + " ");

        }

    }

}



Program – 11

//Write a program to reverse the elements of a given 2\*2 array. Four integer numbers need to be passed as Command-Line arguments.

package *LAB*\_03;

public class prg\_11\_Reverse2DArray {

    public static void main(String[] args) {

        if (args.length != 4) {

            System.out.println("Please provide exactly 4 integers as command-line arguments.");

            return;

        }

        int[][] matrix = { { Integer.parseInt(args[0]), Integer.parseInt(args[1]) },

                { Integer.parseInt(args[2]), Integer.parseInt(args[3]) } };

        System.out.println("Original 2\*2 Array:");

        print2DArray(matrix);

        System.out.println("Reversed 2\*2 Array:");

        reverse2DArray(matrix);

        print2DArray(matrix);

  }

    private static void reverse2DArray(int[][] array) {

        int temp = array[0][0];

        array[0][0] = array[1][1];

        array[1][1] = temp;

        temp = array[0][1];

        array[0][1] = array[1][0];

        array[1][0] = temp;

    }

    private static void print2DArray(int[][] array) {

        for (int[] row : array) {

            for (int element : row) {

                System.out.print(element + " ");

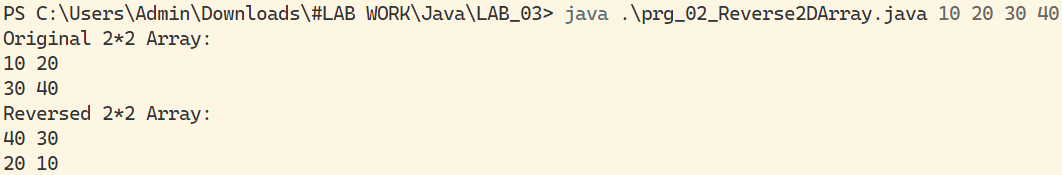
            }

            System.out.println();

        }

    }

}



Program - 12

package *LAB*\_03;

class Node {

    int data;

    Node next;

    public Node(int d) {

        data = d;

        next = null;

    }

}

public class prg\_12\_Stack\_LL {

    static class Stacks {

        public static Node Top;

        public static int Size;

        boolean IsEmpty() {

            return (Top == null);

        }

        void Push(int data) {

            Node NewNode = new Node(data);

            if (IsEmpty()) {

                Top = NewNode;

                Size++;

                return;

            }

            NewNode.next = Top;

            Top = NewNode;

            Size++;

        }

        void Pop() {

            if (IsEmpty()) {

                System.out.println("THE STACK IS EMPTY , DELETION NOT POSSIBLE");

                return;

            }

            Node Temp = Top;

            Top = Top.next;

            Temp.next = null;

            Size--;

        }

        void Peek() {

            if (IsEmpty()) {

                System.out.println("THE STACK IS EMPTY , PEEK NOT POSSIBLE");

                return;

            }

            System.out.println("THE TOP OF THE IS ::: " + Top.data);

        }

        void Size\_Stack() {

            System.out.println("THE SIZE OF THE STACK IS ::: " + Size);

        }

        void Display() {

            System.out.print("THE STACK IS ::: (Top) --> ");

            Node temp = Top;

            while (temp != null) {

                System.out.print(temp.data + " ");

                temp = temp.next;

            }

            System.out.println();

        }

    }

    public static void main(String args[]) {

        Stacks S1 = new Stacks();

        System.out.println();

        System.out.println();

        S1.Push(1);

        S1.Push(2);

        S1.Push(3);

        S1.Push(4);

        S1.Push(5);

        S1.Display();

        S1.Pop();

        S1.Display();

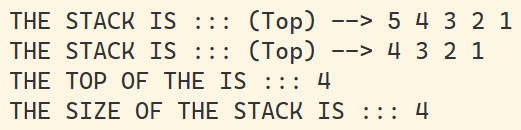
        S1.Peek();

        S1.Size\_Stack();

        System.out.println();

    }

}



Program - 13

package *LAB*\_03;

class Node {

    int data;

    Node next;

    public Node(int d) {

        data = d;

        next = null;

    }

}

public class prg\_13\_Queue\_LL {

    static class Queues {

        public static Node Front;

        public static Node Rear;

        public static int Size;

        boolean IsEmpty() {

            return (Rear == null);

        }

        void Enqueue(int data) {

            Node NewNode = new Node(data);

            if (IsEmpty()) {

                Front = Rear = NewNode;

                Size++;

                return;

            }

            Rear.next = NewNode;

            Rear = NewNode;

            Size++;

        }

        void Dequeue() {

            if (IsEmpty()) {

                System.out.println("THE STACK IS EMPTY , DELETION NOT POSSIBLE");

                return;

            }

            Node Temp = Front;

            Front = Front.next;

            Temp.next = null;

            Size--;

        }

        void Peek\_First() {

            if (IsEmpty()) {

                System.out.println("THE QUEUE IS EMPTY , PEEK NOT POSSIBLE");

                return;

            }

            System.out.println("THE FRONT OF THE QUEUE IS ::: " + Front.data);

        }

        void Peek\_Last() {

            if (IsEmpty()) {

                System.out.println("THE QUEUE IS EMPTY , PEEK NOT POSSIBLE");

                return;

            }

            System.out.println("THE REAR OF THE QUEUE IS ::: " + Rear.data);

        }

        void Size\_Queue() {

            System.out.println("THE SIZE OF THE QUEUE IS ::: " + Size);

        }

        void Display() {

            System.out.print("THE QUEUE IS ::: (Front) --> ");

            Node temp = Front;

            while (temp != null) {

                System.out.print(temp.data + " ");

                temp = temp.next;

            }

            System.out.println("<-- (Rear)");

        }

    }

    public static void main(String args[]) {

        Queues q = new Queues();

        System.out.println();

        System.out.println();

        q.Enqueue(1);

        q.Enqueue(2);

        q.Enqueue(3);

        q.Enqueue(4);

        q.Enqueue(5);

        q.Display();

        q.Dequeue();

        q.Display();

        q.Peek\_First();

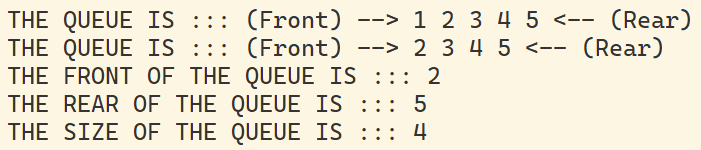
        q.Peek\_Last();

        q.Size\_Queue();

        System.out.println();

    }

}



Program - 14

//Write a Java program to produce the tokens from the given long string.

package *LAB*\_03;

import java.util.StringTokenizer;

public class prg\_14\_Tokenizer {

    public static void main(String[] args) {

        String longString = "This is a long string with multiple words.";

        StringTokenizer tokenizer = new StringTokenizer(longString);

        System.out.println("Tokens:");

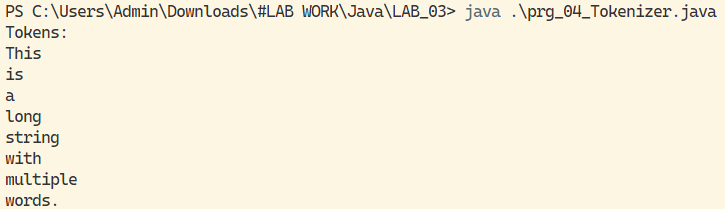
        while (tokenizer.hasMoreTokens()) {

            System.out.println(tokenizer.nextToken());

        }

    }

}



Program – 15

//Using the concept of method overloading, write methods for calculating the area of a triangle, circle, and rectangle.

package *LAB*\_03;

public class prg\_15\_AreaCalculator {

    public static void main(String[] args) {

        // Example usage

        System.out.println("Area of Triangle: " + calculateArea(5.324, 8.765));

        System.out.println("Area of Circle: " + calculateArea(3.5));

        System.out.println("Area of Rectangle: " + calculateArea(4, 6));

    }

    // Area of Triangle

    private static double calculateArea(double base, double height) {

        return 0.5 \* base \* height;

    }

    // Area of Circle

    private static double calculateArea(double radius) {

        return Math.PI \* radius \* radius;

    }

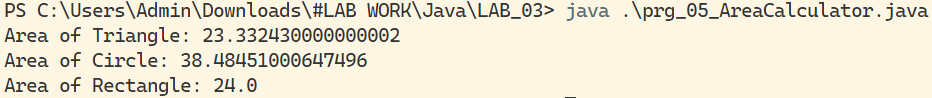
    // Area of Rectangle

    private static double calculateArea(int length, int width) {

        return length \* width;

    }

}



Lab Exercise - 4

Program - 16

//Create a class Box with a parameterized constructor and a method to calculate the volume:

package *LAB*\_04;

public class prg\_16\_Box {

    private double width;

    private double height;

    private double depth;

    // Parameterized constructor

    public prg\_16\_Box(double width, double height, double depth) {

        this.width = width;

        this.height = height;

        this.depth = depth;

    }

    // Method to calculate volume

    public double calculateVolume() {

        return width \* height \* depth;

    }

    public static void main(String[] args) {

        // Create an object of the Box class

        prg\_16\_Box myBox = new prg\_16\_Box(3.0, 4.0, 5.0);

        // Test the functionalities

        System.out.println("Volume of the box: " + myBox.calculateVolume());

    }

}



Program - 17

//Write a program to display the use of this keyword:

package *LAB*\_04;

public class prg\_17\_This {

    private int value;

    // Parameterized constructor using this keyword

    public prg\_17\_This(int value) {

        this.value = value;

    }

    // Method using this keyword

    public void displayValue() {

        System.out.println("Value: " + this.value);

    }

    public static void main(String[] args) {

        // Create an object of the class

        prg\_17\_This obj = new prg\_17\_This(42);

        // Call the method to display the value

        obj.displayValue();

    }

}



Program - 18

//Write a program to count the number of instances created for the class:

package *LAB*\_04;

public class prg\_18\_InstanceCounter {

    private static int instanceCount = 0;

    // Constructor increments the instance count

    public prg\_18\_InstanceCounter() {

        instanceCount++;

    }

    // Static method to get the instance count

    public static int getInstanceCount() {

        return instanceCount;

    }

    public static void main(String[] args) {

        // Create instances of the class

        prg\_18\_InstanceCounter obj1 = new prg\_18\_InstanceCounter();

        prg\_18\_InstanceCounter obj2 = new prg\_18\_InstanceCounter();

        // Get and display the instance count

        System.out.println("Number of instances created: " + prg\_18\_InstanceCounter.getInstanceCount());

    }

}



Program - 19

//Java program to get the cube of a given number using a static method:

package *LAB*\_04;

public class prg\_19\_CubeCalculator {

    // Static method to calculate the cube

    public static double calculateCube(double number) {

        return Math.pow(number, 3);

    }

    public static void main(String[] args) {

        // Test the static method

        double result = prg\_19\_CubeCalculator.calculateCube(4.0);

        System.out.println("Cube of the given number: " + result);

    }

}



Lab Exercise - 5

Program – 20

//Implement Method Overriding:

package *LAB*\_05;

class Animal {

    void sound() {

        System.out.println("Animal makes a sound");

    }

}

class Dog extends Animal {

    void sound() {

        System.out.println("Dog barks");

    }

}

public class prg\_20\_MethodOverriding {

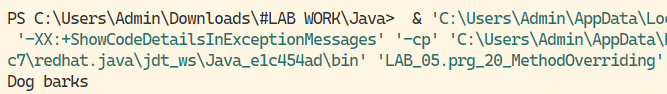
    public static void main(String[] args) {

        Animal animal = new Dog();

        animal.sound(); // Calls the overridden method in Dog class

    }

}



Program – 21

//Illustrate Simple Inheritance:

package *LAB*\_05;

class Parent {

    void display() {

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

    }

}

class Child extends Parent {

    void show() {

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

    }

}

public class prg\_21\_SimpleInheritance {

    public static void main(String[] args) {

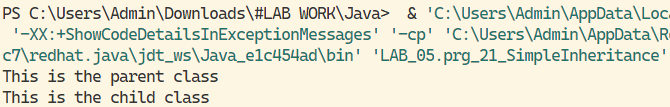
        Child childObj = new Child();

        childObj.display(); // Accessing method from the parent class

        childObj.show(); // Accessing method from the child class

    }

}



Program – 22

//Illustrate Multilevel Inheritance:

package *LAB*\_05;

class Vehicle {

    void start() {

        System.out.println("Vehicle started");

    }

}

class Car extends Vehicle {

    void accelerate() {

        System.out.println("Car is accelerating");

    }

}

class SportsCar extends Car {

    void turboCharge() {

        System.out.println("Sports car turbocharged");

    }

}

public class prg\_22\_MultilevelInheritance {

    public static void main(String[] args) {

        SportsCar sportsCarObj = new SportsCar();

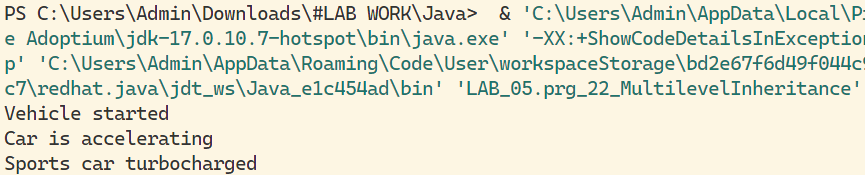
        sportsCarObj.start(); // Accessing method from the Vehicle class

        sportsCarObj.accelerate(); // Accessing method from the Car class

        sportsCarObj.turboCharge(); // Accessing method from the SportsCar class

    }

}



Program – 23

// Illustrate all Uses of super Keyword:

package *LAB*\_05;

class Base {

    int x = 10;

    void display() {

        System.out.println("This is the Base class");

    }

}

class Derived extends Base {

    int x = 20;

    void show() {

        int x = 30;

        System.out.println("Local variable x: " + x);

        System.out.println("Derived class variable x: " + this.x);

        System.out.println("Base class variable x: " + super.x);

        super.display(); // Calls the method from the Base class using super

    }

}

public class prg\_23\_SuperKeyword {

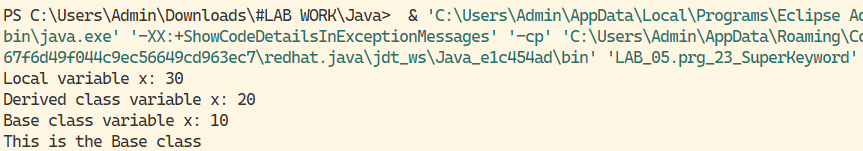
    public static void main(String[] args) {

        Derived derivedObj = new Derived();

        derivedObj.show();

    }

}



Program – 24

//Show Dynamic Polymorphism and Interface Overriding:

package *LAB*\_05;

interface Shape {

    void draw();

}

class Circle implements Shape {

    @Override

    public void draw() {

        System.out.println("Drawing Circle");

    }

}

class Rectangle implements Shape {

    @Override

    public void draw() {

        System.out.println("Drawing Rectangle");

    }

}

public class prg\_24\_DynamicPolymorphism {

    public static void main(String[] args) {

        Shape circle = new Circle();

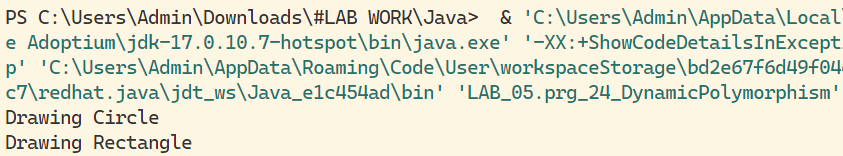
        Shape rectangle = new Rectangle();

        circle.draw(); // Calls Circle's implementation of draw()

        rectangle.draw(); // Calls Rectangle's implementation of draw()

    }

}



Lab Exercise - 6

Program – 25

//WAP to create an abstract class "Shape" where "Rectangle" & "Triangle" inherit the "Shape" class.

package *LAB*\_06;

abstract class Shape {

    int Base, Height;

    public abstract void Show();

    public abstract void Area();

}

class Rectangle extends Shape {

    public Rectangle(int b, int h) {

        Base = b;

        Height = h;

    }

    @Override

    public void Show() {

        System.out.println("The Rectangle Has Base = " + Base + " & Height = " + Height);

    }

    public void Area() {

        System.out.println("The Area Of Rectangle is -> " + (Base \* Height));

    }

}

class Triangle extends Shape {

    public Triangle(int b, int h) {

        Base = b;

        Height = h;

    }

    @Override

    public void Show() {

        System.out.println("The Triangle Has Base = " + Base + " & Height = " + Height);

    }

    public void Area() {

        System.out.println("The Area Of Triangle is -> " + (0.5 \* Base \* Height));

    }

}

public class prg\_25\_AbstractClass {

    public static void main(String args[]) {

        System.out.println();

        Shape s = new Rectangle(10, 20);

        s.Show();

        s.Area();

        System.out.println();

        s = new Triangle(20, 20);

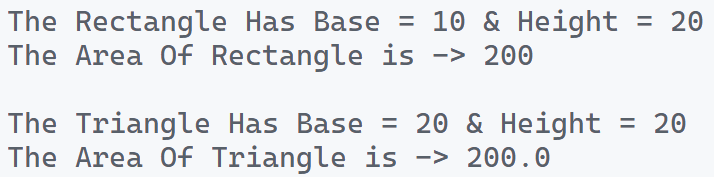
        s.Show();

        s.Area();

        System.out.println();

    }

}



Program – 26

//WAP that creates an Interface & implements it

package *LAB*\_06;

// Define the interface

interface MyInterface {

    // Abstract method declaration

    void myMethod();

}

// Implement the interface in a class

class MyClass implements MyInterface {

    // Implementing the abstract method from the interface

    public void myMethod() {

        System.out.println("Implementing myMethod() in MyClass");

    }

}

public class prg\_26\_Interface {

    public static void main(String[] args) {

        // Create an object of the implementing class

        MyClass obj = new MyClass();

        // Call the method implemented from the interface

        obj.myMethod();

    }

}



Program – 27

/\*

Write an interface "playable" with a method void "play()", let this Interface be placed in a package called "music".

Write a class "Veena" which implement the "playable" interface, let this class be placed in a package called "music.string".

Write a class "saxophone" which implement the "playable" interface, let this class be placed in a package called "music.wind".

Write another class "test" in package "live". Then ->

(i) create an instance of "Veena" and call the "play()" method

(ii) create an instance of "saxophone" and Call the "play()" method

(iii) place the above instances in a variable of type "playable" and then call "play())"

\*/

playable.java

package *LAB*\_06.prg\_27\_*P*ackage*I*nterface.live.music;

public interface playable {

    void play();

}

veena.java

package *LAB*\_06.prg\_27\_*P*ackage*I*nterface.live.music.string;

import LAB\_06.prg\_27\_PackageInterface.live.music.playable;

public class veena implements playable {

    String name;

    public veena(String n) {

        this.name = n;

    }

    @Override

    public void play() {

        System.out.println(name + "can play veena well!");

    }

    public static void main(String[] args) {

        playable p = new veena("Amit ");

        p.play();

    }

}

saxophone.java

package *LAB*\_06.prg\_27\_*P*ackage*I*nterface.live.music.wind;

import LAB\_06.prg\_27\_PackageInterface.live.music.playable;

public class saxophone implements playable {

    String name;

    public saxophone(String n) {

        this.name = n;

    }

    @Override

    public void play() {

        System.out.println(name + "can play saxophone well!");

    }

    public static void main(String[] args) {

        playable p = new saxophone("Amit ");

        p.play();

    }

}

test.java

package *LAB*\_06.prg\_27\_*P*ackage*I*nterface.live;

import LAB\_06.prg\_27\_PackageInterface.live.music.playable;

import LAB\_06.prg\_27\_PackageInterface.live.music.string.veena;

import LAB\_06.prg\_27\_PackageInterface.live.music.wind.saxophone;

public class test {

    public static void main(String[] args) {

        System.out.print("Instance of Veena Class : ");

        veena v = new veena("Amit ");

        v.play();

        System.out.print("Instance of Saxophone Class : ");

        saxophone s = new saxophone("Amit ");

        s.play();

        System.out.println("Instance of Playable Interface ->");

        playable p = new veena("Amit ");

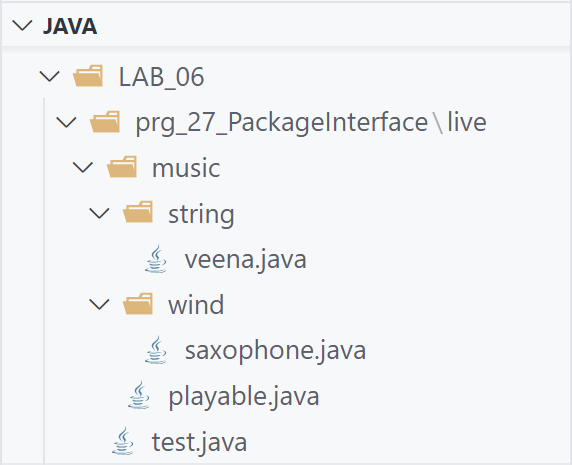
        p.play();

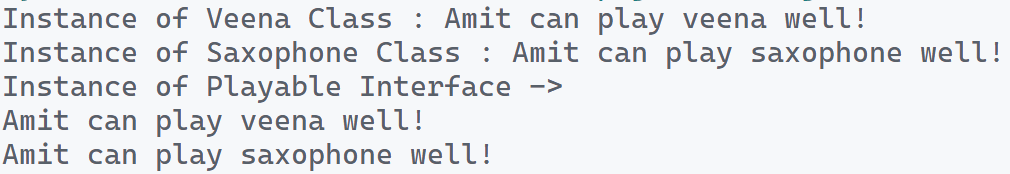
        p = new saxophone("Amit ");

        p.play();

    }

}





Lab Exercise - 7

Program – 28

//WAP to accept name & age of a person from the user. Ensure that entered age is between 15 & 60. Display proper error message & the program must execute gracefully after displaying the error message in case the argument pass is not proper.

package *LAB*\_07;

import java.util.\*;

class MyException extends Exception {

    public MyException(String message) {

        super(message);

    }

}

public class prg\_28\_Exception {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        System.out.println();

        String name;

        int Age;

        try {

            System.out.print("Enter your Name -> ");

            name = sc.nextLine();

            System.out.print("Enter your Age -> ");

            Age = sc.nextInt();

            System.out.println();

            if (Age < 15) {

                throw new MyException(name + " Age is " + Age + " Which Is Less Than 15 Years");

            } else if (Age > 60) {

                throw new MyException(name + " Age is " + Age + " Which Is More Than 60 Years");

            } else {

                System.out.println(name + " Age is ::: " + Age);

            }

        } catch (Exception e) {

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

        } finally {

            System.out.println("Finally Program is Finished...");

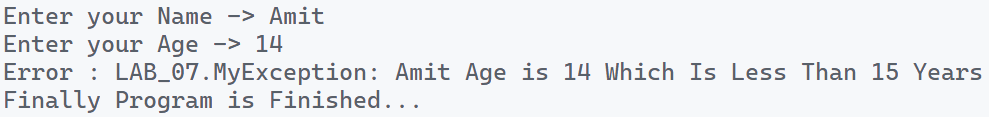
            System.out.println();

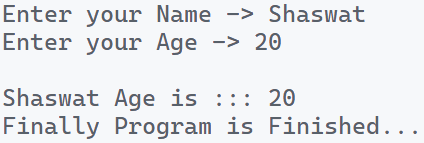
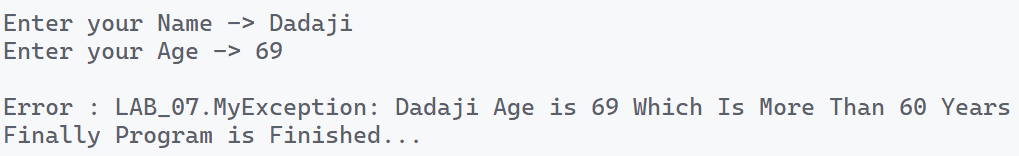
            sc.close();

        }

    }

}





Program – 29

//WAP to create a customized exception & also make use of all the 5 exception keywords.

package *LAB*\_07;

class MyException extends Exception {

    public MyException(String message) {

        super(message);

    }

}

public class prg\_29\_CustomException {

    public static void CheckException(String Name, int Age) {

        try {

            Check(Name, Age);

        } catch (MyException e) {

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

        } finally {

            System.out.println("Finally Program Executed...");

            System.out.println();

        }

    }

    public static void Check(String Name, int Age) throws MyException {

        if (Age < 18) {

            throw new MyException(Name + " Is Not Eligible For Voting");

        } else {

            System.out.println(Name + " Is Eligible For Voting");

        }

    }

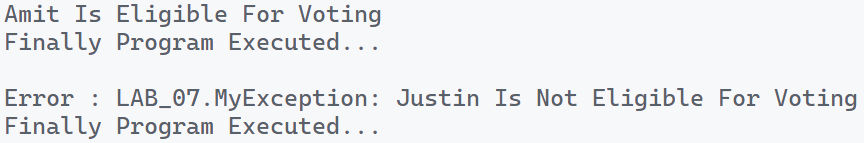
    public static void main(String args[]) {

        CheckException("Amit", 21);

        CheckException("Justin", 17);

    }

}



Program – 30

//Write an applet prg that displays "Hello World" with background color "black", text color "blue" and your name in the status window.

.java file

package *LAB*\_07;

import java.applet.\*;

import java.awt.\*;

public class prg\_30\_Applet1 extends Applet {

    public void paint(Graphics g) {

        g.setColor(Color.blue);

        g.drawString("Hello World", 50, 50);

    }

}

.html file

<!DOCTYPE html>

<html lang="en">

  <head>

    <meta charset="UTF-8" />

    <meta name="viewport" content="width=device-width, initial-scale=1.0" />

    <title>Hello World</title>

  </head>

  <body>

    <*applet* code="prg\_30\_Applet1.class" width="300" height="300"></*applet*>

  </body>

</html>

Program – 31

//Develop an Analog Clock using applet.

package *LAB*\_07;

import java.applet.\*;

import java.awt.\*;

import java.util.\*;

public class prg\_31\_Applet2 extends Applet {

    @Override

    public void init() {

        this.setSize(new Dimension(800, 400));

        setBackground(new Color(50, 50, 50));

        new Thread() {

            @Override

            public void run() {

                while (true) {

                    repaint();

                    delayAnimation();

                }

            }

        }.start();

    }

    private void delayAnimation() {

        try {

            Thread.sleep(1000);

        } catch (InterruptedException e) {

            e.printStackTrace();

        }

    }

    @Override

    public void paint(Graphics g) {

        // Get the system time

        Calendar time = Calendar.getInstance();

        int hour = time.get(Calendar.HOUR\_OF\_DAY);

        int minute = time.get(Calendar.MINUTE);

        int second = time.get(Calendar.SECOND);

        // 12 hour format

        if (hour > 12) {

            hour -= 12;

        }

        // Draw clock body center at (400, 200)

        g.setColor(Color.white);

        g.fillOval(300, 100, 200, 200);

        // Labeling

        g.setColor(Color.black);

        g.drawString("12", 390, 120);

        g.drawString("9", 310, 200);

        g.drawString("6", 400, 290);

        g.drawString("3", 480, 200);

        // Declaring variables to be used

        double angle;

        int x, y;

        // Second hand's angle in Radian

        angle = Math.toRadians((15 - second) \* 6);

        // Position of the second hand with length 100 unit

        x = (int) (Math.cos(angle) \* 100);

        y = (int) (Math.sin(angle) \* 100);

        // Red color second hand

        g.setColor(Color.red);

        g.drawLine(400, 200, 400 + x, 200 - y);

        // Minute hand's angle in Radian

        angle = Math.toRadians((15 - minute) \* 6);

        // Position of the minute hand

        // with length 80 unit

        x = (int) (Math.cos(angle) \* 80);

        y = (int) (Math.sin(angle) \* 80);

        // blue color Minute hand

        g.setColor(Color.blue);

        g.drawLine(400, 200, 400 + x, 200 - y);

        // Hour hand's angle in Radian

        angle = Math.toRadians((15 - (hour \* 5)) \* 6);

        // Position of the hour hand

        // with length 50 unit

        x = (int) (Math.cos(angle) \* 50);

        y = (int) (Math.sin(angle) \* 50);

        // Black color hour hand

        g.setColor(Color.black);

        g.drawLine(400, 200, 400 + x, 200 - y);

    }

}



Lab Exercise - 8

Program – 32

//WAP to show Multi-Threading

package *LAB*\_08;

class Base {

    int num;

    boolean ValueSet = false;

    public synchronized void Put(int n) {

        while (ValueSet == true) {

            try {

                wait();

            } catch (Exception e) {

            }

        }

        System.out.println("Put -->  num : " + n);

        this.num = n;

        ValueSet = true;

        notify();

    }

    public synchronized void Get() {

        while (ValueSet == false) {

            try {

                wait();

            } catch (Exception e) {

            }

        }

        System.out.println("Get -->  num : " + num);

        ValueSet = false;

        notify();

    }

}

class Producer implements Runnable {

    Base Obj;

    public Producer(Base b) {

        this.Obj = b;

        Thread T1 = new Thread(this, "Producer");

        T1.start();

    }

    @Override

    public void run() {

        int i = 0;

        while (i <= 5) {

            Obj.Put(i++);

            try {

                Thread.sleep(1000);

            } catch (Exception e) {

                e.getStackTrace();

            }

        }

    }

}

class Consumer implements Runnable {

    Base Obj;

    public Consumer(Base b) {

        this.Obj = b;

        Thread T2 = new Thread(this, "Consumer");

        T2.start();

    }

    @Override

    public void run() {

        int i = 0;

        while (i <= 5) {

            Obj.Get();

            try {

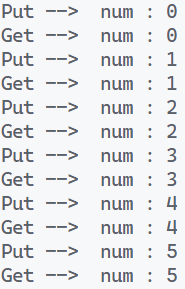
                Thread.sleep(2000);

            } catch (Exception e) {

                e.getStackTrace();

            }

        }

    }

}

public class prg\_32\_MultiThread {

    public static void main(String args[]) {

        Base Obj = new Base();

        Producer P = new Producer(Obj);

        Consumer C = new Consumer(Obj);

    }

}

Program – 33

//WAP that executes to threads. One thread displays "An" after every 1000ms & the other displays "B" after every 3000ms. Create the threads by executing the thread class.

package *LAB*\_08;

class First extends Thread {

    public void run() {

        for (int i = 0; i < 9; i++) {

            System.out.print("An    ");

            try {

                Thread.sleep(1000);

            } catch (Exception e) {

            }

        }

    }

}

class Second extends Thread {

    public void run() {

        for (int i = 0; i < 3; i++) {

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

            try {

                Thread.sleep(3000);

            } catch (Exception e) {

            }

        }

    }

}

public class prg\_33\_Thread1 {

    public static void main(String args[]) {

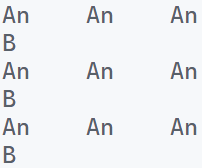
        System.out.println();

        First F = new First();

        Second S = new Second();

        F.start();

        try {

            Thread.sleep(3000);

        } catch (Exception e) {

        }

        S.start();

    }

}

Program – 34

//WAP & Create a class "salesperson" as a thread that will display a salesperson name. Create a class "days" as other thread that has array of 7 days. Call the instance of "salesperson" in "days" and start both the threads. Suspend the sales person on Sunday & resume on Wednesday. We can only use suspend and resume methods from the thread only.

package *LAB*\_08;

class SalesPerson extends Thread {

    public String name;

    public SalesPerson(String n) {

        this.name = n;

    }

    public void run() {

        System.out.println("Sales\_Person " + name + " reporting for duty.");

        System.out.println();

    }

}

class Days extends Thread {

    String[] days = { "Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday" };

    SalesPerson salesPerson;

    public Days(SalesPerson salesPerson) {

        this.salesPerson = salesPerson;

    }

    public void run() {

        for (String day : days) {

            if (day.equals("Sunday")) {

                System.out.println("Sales\_Person " + salesPerson.name + " suspended shop on : " + day);

                // salesPerson.suspend();

            } else if (day.equals("Monday") || day.equals("Tuesday")) {

                continue;

            } else if (day.equals("Wednesday")) {

                System.out.println("Sales\_Person " + salesPerson.name + " resumed shop on : " + day);

            } else {

                System.out.println("Sales\_Person " + salesPerson.name + " continued shop on : " + day);

            }

        }

    }

}

public class prg\_34\_Thread2 {

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

        System.out.println();

        SalesPerson SP = new SalesPerson("Amit");

        Days DP = new Days(SP);

        SP.start();

        DP.start();

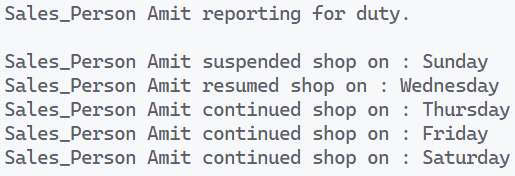
        SP.join();

        DP.join();

        System.out.println();

    }

}



Program – 35

//WAP that read & write in the file.

package *LAB*\_08;

import java.io.\*;

import java.util.\*;

public class prg\_35\_FileHandling {

    public static void main(String args[]) {

        try {

            File myFile = new File("D:\\Admin\\B.Tech\\LAB-WORK\\Java\\LAB\_09\\prg\_35\_FileHandling.txt");

            myFile.createNewFile();

            FileWriter WriteFile = new FileWriter("D:\\Admin\\B.Tech\\LAB-WORK\\Java\\LAB\_09\\prg\_35\_FileHandling.txt");

            WriteFile.write("Hello, I Am Amit Singhal\nI Am 18 Years Old\n");

            WriteFile.write("Java is my Favourite Subject");

            WriteFile.close();

            Scanner sc = new Scanner(myFile);

            while (sc.hasNextLine()) {

                String line = sc.nextLine();

                System.out.println(line);

            }

            sc.close();

            myFile.delete();

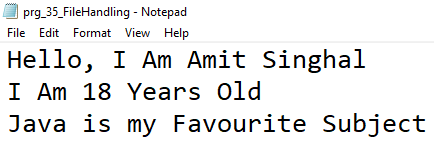
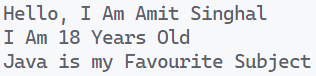
        } catch (Exception e) {

            e.printStackTrace();

        }

    }

}

*  TXT FILE