PRACTICAL ASSIGNMENT

1.Write a program for checking number is positive or negative .

```
import java.util.Scanner;
public class CheckPositiveOrNegative {
    public static void main(String[] args) {
        int num;
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter a number: ");
        num = sc.nextInt();
        if (num > 0) {
            System.out.println("The number is positive.");
        } else if (num < 0) {
            System.out.println("The number is negative.");
        } else {
            System.out.println("The number is zero.");
        }
    }
}
```

2.Write a program to finding area of circle.

```
import java.util.Scanner;
class AreaOfCircle {
    public static void main(String args[]) {
        Scanner s = new Scanner(System.in);
        System.out.println("Enter the radius:");
        double r = s.nextDouble();
        double area = (22 * r * r) / 7;
        System.out.println("Area of Circle is: " + area);
```

```
    }
}
```

3.Write a program to find maximum and minimum number from given list.

```
import java.util.*;
public class MinMaxValue {
    public static void main(String[] args) {
        List<Integer> numbers = Arrays.asList(10, 5, 8, 100,20, 3, 15);
        int max = findMax(numbers);
        int min = findMin(numbers);
        System.out.println("Maximum number: " + max);
        System.out.println("Minimum number: " + min);
    }
    public static int findMax(List<Integer> numbers) {
        int max = Integer.MIN_VALUE;
        for (int num : numbers) {
            if (num > max) {
                max = num;
            }
        }
        return max;
    }
    public static int findMin(List<Integer> numbers) {
        int min = Integer.MAX_VALUE;
        for (int num : numbers) {
            if (num < min) {
                min = num;
            }
        }
        return min;
```

```
    }
}
```

4.Write a program to print following pattern.

```
*
* *
* * *
public class RightTrianglePattern {
    public static void main(String args[]) {
        int i, j, row = 3;
        for (i = 0; i < row; i++) {
            for (j = 0; j <= i; j++) {
                System.out.print("* ");
            }
            System.out.println();
        }
    }
}
```

5.Write a program to print following pattern.

```
* * *
 * *
  *
public class ReversePyramidPattern {
    public static void main(String[] args) {
        int rows = 3;
        for (int i = 0; i <= rows - 1; i++) {
            for (int j = 0; j <= i; j++) {
                System.out.print(" ");
            }
```

```
            for (int k = 0; k <= rows - 1 - i; k++) {
                System.out.print("*" + " ");
            }
            System.out.println();
        }
    }
}
```

6.Write a program to print multiplication table of given number.

```
import java.util.Scanner;
public class TableExample {
    public static void main(String args[]) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter number: ");
        int num = sc.nextInt();
        for (int i = 1; i <= 10; i++) {
            System.out.println(num + " * " + i + " = " + num * i);
        }
    }
}
```

7.Write a program to add two numbers using function overloading.

```
public class  Sum {
    public int sum(int x, int y) {
        return (x + y);
    }
    public int sum(int x, int y, int z) {
        return (x + y + z);
    }
    public double sum(double x, double y) {
```

```
        return (x + y);
    }
    public static void main(String args[]) {
        Sum s = new Sum();
        System.out.println(s.sum(10, 20));
        System.out.println(s.sum(10, 20, 30));
    }
}
```

8.Write a program to input Employee Details and display it on proper format.

```
import java.util.Scanner;
public class EmployeeDetails {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter employee name: ");
        String name = sc.nextLine();
        System.out.print("Enter employee ID: ");
        int id = sc.nextInt();
        System.out.print("Enter employee salary: ");
        double salary = sc.nextDouble();
        System.out.print("Enter employee department: ");
        String department = sc.next();
        System.out.println("\nEmployee Details:");
        System.out.println("Name: " + name);
        System.out.println("ID: " + id);
        System.out.println("Salary: $" + salary);
        System.out.println("Department: " + department);
    }
}
```

9.Write a program to design three classes that accept dimension of triangle and rectangle and calculate area of rectangle and triangle .

```
import java.util.Scanner;
public class Main {
    public static void main(String[] args) {
        Triangle triangle = new Triangle();
        triangle.inputDimensions();
        triangle.calculateAr5ea();
        triangle.displayArea();
        Rectangle rectangle = new Rectangle();
        rectangle.inputDimensions();
        rectangle.calculateArea();
        rectangle.displayArea();
    }
}
class Shape {
    double area;
    public void calculateArea() {
    }
    public void displayArea() {
        System.out.println("Area: " + area);
    }
}
class Triangle extends Shape {
    double base;
    double height;
    public void inputDimensions() {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter base of the triangle: ");
        base = scanner.nextDouble();
```

```
        System.out.print("Enter height of the triangle: ");
        height = scanner.nextDouble();
        scanner.close();
    }
    @Override
    public void calculateArea() {
        area = 0.5 * base * height;
    }
}
class Rectangle extends Shape {
    double length;
    double width;
    public void inputDimensions() {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter length of the rectangle: ");
        length = scanner.nextDouble();
        System.out.print("Enter width of the rectangle: ");
        width = scanner.nextDouble();
        scanner.close();
    }
    @Override
    public void calculateArea() {
        area = length * width;
    }
}
```

10.Write a program which design Bank Account class as Saving and Current Account and manage information accordingly.

```
import java.util.Scanner;
public class BankAccountDemo10 {
```

```
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Welcome to the Bank!");
        System.out.println("Choose Account Type:");
        System.out.println("1. Saving Account");
        System.out.println("2. Current Account");
        int choice = sc.nextInt();
        switch (choice) {
            case 1:
                SavingAccount savingAccount = new SavingAccount();
                savingAccount.displaySavingAccount();
                break;
            case 2:
                CurrentAccount currentAccount = new CurrentAccount();
                currentAccount.displayCurrentAccount();
                break;
            default:
                System.out.println("Invalid choice!");
        }
    }
}
class BankAccount {
    String accountNumber = "12345";
    double balance = 0.0;
    public void displayBankAccount() {
        System.out.println("Account Number: " + accountNumber);
        System.out.println("Balance: " + balance);
    }
}
class SavingAccount extends BankAccount {
```

```
    double interestRate = 0.05;// 5%
    public void displaySavingAccount() {
        System.out.println("Account Number: " + accountNumber);
        System.out.println("Balance: " + balance);
        System.out.println("Account Type: Saving");
        System.out.println("Interest Rate: " + interestRate);
    }
}
class CurrentAccount extends BankAccount {
    double overdraftLimit = 10000.0;
    public void displayCurrentAccount() {
        System.out.println("Account Number: " + accountNumber);
        System.out.println("Balance: " + balance);
        System.out.println("Account Type: Saving");
        System.out.println("Interest Rate: " + overdraftLimit);
    }
}
```

11.Write a program which design a class name Fan to represent fan properties according to these properties Fan operation will be performed.

```
class Fan {
    public static final int OFF = 0;
    public static final int LOW = 1;
    public static final int MEDIUM = 2;
    public static final int HIGH = 3;
    private int speed;
    private boolean isOn;
    private double radius;
    private String color;
    public Fan(double radius, String color) {
```

```
    this.speed = OFF;
    this.isOn = false;
    this.radius = radius;
    this.color = color;
}
public void turnOn() {
    isOn = true;
}
public void turnOff() {
    isOn = false;
    speed = OFF;
}
public void setSpeed(int speed) {
    if (isOn) {
        this.speed = speed;
    } else {
        System.out.println("Fan is turned off. Cannot set speed.");
    }
}
public int getSpeed() {
    return speed;
}
public boolean isOn() {
    return isOn;
}
public double getRadius() {
    return radius;
}
public String getColor() {
    return color;
```

```
    }
    public void display() {
        String status = isOn ? "On" : "Off";
        System.out.println("Fan Status: " + status);
        if (isOn) {
            System.out.println("Speed: " + speed);
        }
        System.out.println("Color: " + color);
        System.out.println("Radius: " + radius + " inches");
    }
}
public class Main {
    public static void main(String[] args) {
        Fan myFan = new Fan(10, "White");
        myFan.turnOn();
        myFan.setSpeed(Fan.MEDIUM);
        myFan.display();
        myFan.turnOff();
        myFan.display();
    }
}
```

12.Write a program that creates two interfaces 1. Direction 2. Drive Car. And creates two classes 1. DirectionBoard 2. Car which inherits above interfaces.

```
public class Main3 {
    public static void main(String[] args) {
        DirectionBoarddirectionBoard = new DirectionBoard("North");
        directionBoard.getDirection();
        System.out.println();
        Car myCar = new Car("Toyota Camry");
```

```
        myCar.getDirection();
        myCar.start();
        myCar.accelerate();
        myCar.brake();
        myCar.stop();
    }
}
interface Direction {
    void getDirection();
}
interface DriveCar {
    void start();
    void accelerate();
    void brake();
    void stop();
}
class DirectionBoard implements Direction {
    private String direction;
    public DirectionBoard(String direction) {
        this.direction = direction;
    }
    @Override
    public void getDirection() {
        System.out.println("Direction: " + direction);
    }
}
class Car implements Direction, DriveCar {
    private String carModel;
    public Car(String carModel) {
        this.carModel = carModel;
```

```
    }
    @Override
    public void getDirection() {
        System.out.println("Car is following the GPS directions.");
    }
    @Override
    public void start() {
        System.out.println(carModel + " is starting.");
    }
    @Override
    public void accelerate() {
        System.out.println(carModel + " is accelerating.");
    }
    @Override
    public void brake() {
        System.out.println(carModel + " is braking.");
    }
    @Override
    public void stop() {
        System.out.println(carModel + " has stopped.");
    }
}
```

13.Write a program to demonstrate partial implementation of interface and extending interfaces.

```
interface Animal {
    void makeSound();
    void eat();
}
abstract class AbstractAnimal implements Animal {
    @Override
```

```
    public void makeSound() {
        System.out.println("Abstract sound");
    }
}
interface Pet extends Animal {
    void play();
}
class Dog extends AbstractAnimal implements Pet {
    @Override
    public void eat() {
        System.out.println("Dog is eating");
    }
    @Override
    public void play() {
        System.out.println("Dog is playing");
    }
}
class Cat extends AbstractAnimal implements Pet {
    @Override
    public void eat() {
        System.out.println("Cat is eating");
    }
    @Override
    public void play() {
        System.out.println("Cat is playing");
    }
}
public class Main {
    public static void main(String[] args) {
        Dog myDog = new Dog();
```

```
        Cat myCat = new Cat();
        myDog.makeSound();
        myDog.eat();
        myDog.play();
        System.out.println();
        myCat.makeSound();
        myCat.eat();
        myCat.play();
    }
}
```

14.Write a program to accept 5 command line argument and then raise the custom exception if any argument is not from the list (“BCA”,”MCA”,”BBA”,”MBA”,”OTHER”).

```
import java.util.Arrays;
public class exception7 {
    public static void main(String[] args) throws MyException {
        String[] course = { "BCA", "MCA", "BBA", "MBA", "OTHER" };
        Arrays.sort(args);
        try {
            if (Arrays.equals(args, course)) {
                System.out.println("\nYour command line arguments are valid.");
            } else {
                throw new MyException("\nSorry , Your command line arguments are not valid.");
            }
        } catch (Exception e) {
            System.out.println(e.getMessage());
        }
    }
}
class MyException extends Exception {
```

```
    public MyException(String s) {
        super(s);
    }
}
```

15.Write a program to demonstrate thow and throws keyword.

```
import java.util.Scanner;
public class demonstrate {
    public static void main(String[] args) throws MyAgeException {
        int age;
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter Your age : ");
        age = sc.nextInt();
        agechecking(age);
    }
    public static void agechecking(int age) {
        try {
            if (age < 18) {
                throw new MyAgeException("Sorry , Your age is not valid for voting.");
            } else {
                System.out.println("Your age is valid for voting.");
            }
        } catch (Exception e) {
            System.out.println(e.getMessage());
        }
    }
}
class MyAgeException extends Exception {
    public MyAgeException(String s) {
        super(s);
```

```
    }
}
```

16.Write a program to demonstrate custom exception called NagetiveNumberArgumentException.

```
class NegativeNumberArgumentException extends Exception {
    public NegativeNumberArgumentException(String message) {
        super(message);
    }
}
class MathOperation {
    public static int square(int number) throws NegativeNumberArgumentException {
        if (number < 0) {
            throw new NegativeNumberArgumentException("Negative number not allowed");
        }
        return number * number;
    }
}
public class Main {
    public static void main(String[] args) {
        try {
            int result1 = MathOperation.square(5);
            System.out.println("Square of 5: " + result1);
            int result2 = MathOperation.square(-3);
            System.out.println("Square of -3: " + result2);
        }
        catch (NegativeNumberArgumentException e) {
            System.out.println("Exception: " + e.getMessage());
        }
    }
}
```

17. Write a program to input command line argument and display those string which start with 'A' and 'a'.

```
public class CommandLineArgumentsExample {
    public static void main(String[] args) {
        System.out.println("Strings starting with 'A' or 'a':");
        for (String arg : args) {
            // Check if the string starts with 'A' or 'a'
            if (arg.startsWith("A") || arg.startsWith("a")) {
                System.out.println(arg);
            }
        }
    }
}
```

18.Write java application which accept a string and display the string in reverse order by interchanging its odd positioned characters with even positioned characters.

```
import java.util.Scanner;
public class ReverseStringInterchange {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a string: ");
        String inputString = scanner.nextLine();
        String reversedString = reverseAndInterchange(inputString);
        System.out.println("Reversed string with interchanged characters: " + reversedString);
    }
    private static String reverseAndInterchange(String str) {
        char[] charArray = str.toCharArray();
        int start = 0;
        int end = charArray.length - 1;
        while (start < end) {
            char temp = charArray[start];
```

```
            charArray[start] = charArray[end];
            charArray[end] = temp;
            start++;
            end--;
        }
        for (int i = 0; i < charArray.length - 1; i += 2) {
            char temp = charArray[i];
            charArray[i] = charArray[i + 1];
            charArray[i + 1] = temp;
        }
        return new String(charArray);
    }
}
```

19.Write a program to input two strings search similar characters from both string and replace it with '*'.

```
import java.util.Scanner;
public class ReplaceSimilarCharacters {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the first string: ");
        String str1 = scanner.nextLine();
        System.out.print("Enter the second string: ");
        String str2 = scanner.nextLine();
        String result = replaceSimilarCharacters(str1, str2);
        System.out.println("Strings after replacing similar characters with '*':");
        System.out.println("String 1: " + str1);
        System.out.println("String 2: " + str2);
        System.out.println("Result: " + result);
    }
```

```
    private static String replaceSimilarCharacters(String str1, String str2) {
        StringBuilder result = new StringBuilder();
        for (int i = 0; i < str1.length(); i++) {
            char char1 = str1.charAt(i);
            if (str2.indexOf(char1) != -1) {
                result.append('*');
            } else {
                result.append(char1);
            }
        }
        return result.toString();
    }
}
```

20.Write a program to perform bubble sort on given inputted String.

```
import java.util.Scanner;
public class BubbleSortString {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the number of strings: ");
        int n = scanner.nextInt();
        scanner.nextLine();
        String[] arr = new String[n];

        System.out.println("Enter the strings:");
        for (int i = 0; i < n; i++) {
            arr[i] = scanner.nextLine();
        }
        bubbleSort(arr);
        System.out.println("Sorted Strings:");
```

```
        for (String str : arr) {
            System.out.println(str);
        }
    }
    private static void bubbleSort(String[] arr) {
        int n = arr.length;
        for (int i = 0; i < n - 1; i++) {
            for (int j = 0; j < n - i - 1; j++) {
                if (arr[j].compareTo(arr[j + 1]) > 0) {
                    String temp = arr[j];
                    arr[j] = arr[j + 1];
                    arr[j + 1] = temp;   }
            }
        }
    }
}
```

21.Write an application that executes three threads from one thread class. One thread displays "JAVA" every 1 second. another display "PAPER" every 2 seconds and last one display "COLLEGE" every 3 seconds. Create the thread by using Runnable Interface .

```
public class MultiThreadRunnableExample {
    public static void main(String[] args) {
        RunnableThreadjavaThread = new RunnableThread("JAVA", 1000);
        RunnableThreadpaperThread = new RunnableThread("PAPER", 2000);
        RunnableThreadcollegeThread = new RunnableThread("COLLEGE", 3000);
        Thread thread1 = new Thread(javaThread);
        Thread thread2 = new Thread(paperThread);
        Thread thread3 = new Thread(collegeThread);
        thread1.start();
        thread2.start();
        thread3.start();
    }
```

```
}
class RunnableThread implements Runnable {
    private String message;
    private int interval;
    public RunnableThread(String message, int interval) {
        this.message = message;
        this.interval = interval;
    }
    @Override
    public void run() {
        try {
            while (true) {
                System.out.println(message);
                Thread.sleep(interval);
            }
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}
```

22.Write a program to implement stack using thread.

```
import java.util.Stack;
class SharedStack {
    private Stack<Integer> stack = new Stack<>();
    synchronized void push(int item) {
        stack.push(item);
        System.out.println("Pushed: " + item);
        notify();
    }
```

```
    synchronized int pop() {
        while (stack.isEmpty()) {
            try {
                wait();
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
        int item = stack.pop();
        System.out.println("Popped: " + item);
        return item;
    }
}
class PushThread extends Thread {
    private SharedStack sharedStack;
    private int value;
    PushThread(SharedStack sharedStack, int value) {
        this.sharedStack = sharedStack;
        this.value = value;
    }
    public void run() {
        sharedStack.push(value);
    }
}
class PopThread extends Thread {
    private SharedStack sharedStack;
    PopThread(SharedStack sharedStack) {
        this.sharedStack = sharedStack;
    }
    public void run() {
```

```
            sharedStack.pop();
        }
    }
    public class StackWithThreads {
        public static void main(String[] args) {
            SharedStack stack = new SharedStack();
            for (int i = 0; i < 5; i++) {
                new PushThread(stack, i).start();
            }
            for (int i = 0; i < 5; i++) {
                new PopThread(stack).start();
            }
        }
    }
```

23.Write a java code which accepts name a of 10 students. Sort the names of students in ascending order. Display the names of students using thread class at interval of one seconds.

```
import java.util.Arrays;
import java.util.Scanner;
class DisplayThread extends Thread {
    private String[] names;
    DisplayThread(String[] names) {
        this.names = names;
    }
    @Override
    public void run() {
        try {
            for (String name : names) {
                System.out.println(name);
                Thread.sleep(1000); // Sleep for one second
```

```
            }
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}
public class StudentNameSorter {
    public static void main(String[] args) {
        String[] students = new String[10];
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter the names of 10 students:");
        for (int i = 0; i < 10; i++) {
            System.out.print("Student " + (i + 1) + ": ");
            students[i] = scanner.nextLine();
        }
        scanner.close();
        Arrays.sort(students);
        DisplayThread displayThread = new DisplayThread(students);
        displayThread.start();
    }
}
```

24.Write a program to demonstrate thread priorities.

```
public class priority {
    public static void main(String[] args) {
        Thread highPriorityThread = new Thread(new WorkerThread(), "HighPriorityThread");
        Thread normalPriorityThread = new Thread(new WorkerThread(), "NormalPriorityThread");
        Thread lowPriorityThread = new Thread(new WorkerThread(), "LowPriorityThread");
        highPriorityThread.setPriority(Thread.MAX_PRIORITY);
        normalPriorityThread.setPriority(Thread.NORM_PRIORITY);
```

```
        lowPriorityThread.setPriority(Thread.MIN_PRIORITY);

        highPriorityThread.start();
        normalPriorityThread.start();
        lowPriorityThread.start();
    }
}
class WorkerThread implements Runnable {
    public void run() {
        for (int i = 1; i <= 5; i++) {
            System.out.println(Thread.currentThread().getName() + " is executing iteration " + i);
            try {
                Thread.sleep(100);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}
```

25.Write a program to demonstrate thread synchronization.

```
public class ThreadSynchronizationExample {
    public static void main(String[] args) {
        Counter counter = new Counter();
        Thread incrementThread1 = new Thread(new IncrementOperation(counter), "Thread1");
        Thread incrementThread2 = new Thread(new IncrementOperation(counter), "Thread2");
        incrementThread1.start();
        incrementThread2.start();
    }
}
```

```
class Counter {
    private int count = 0;

    public synchronized void increment() {
        for (int i = 0; i < 5; i++) {
            System.out.println(Thread.currentThread().getName() + " - Increment: " + (++count));
            try {
                Thread.sleep(500); // Sleep for 500 milliseconds
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}
class IncrementOperation implements Runnable {
    private Counter counter;
    public IncrementOperation(Counter counter) {
        this.counter = counter;
    }
    @Override
    public void run() {
        counter.increment();
    }
}
```

26.Write a java application which displays a Hexagon containing a circle within the pentagon where the circumference of the circle touches to the edges of the pentagon. Provide separate colors to both the objects. Display your name in center position of the circle.

```
import javax.swing.*;
import java.awt.*;
import java.awt.geom.Ellipse2D;
```

```
import java.awt.geom.Path2D;
public class HexagonWithCircle extends JFrame {
    public HexagonWithCircle() {
        setTitle("Hexagon with Circle");
        setSize(400, 400);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null);
        setBackground(Color.WHITE);
        HexagonPanelhexagonPanel = new HexagonPanel();
        add(hexagonPanel);
        setVisible(true);
    }
    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> new HexagonWithCircle());
    }
}
class HexagonPanel extends JPanel {
    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);
        Graphics2D g2d = (Graphics2D) g;
        g2d.setColor(Color.BLUE); // Set color for the hexagon
        drawHexagon(g2d);
        g2d.setColor(Color.RED); // Set color for the circle
        drawCircle(g2d);
        g2d.setColor(Color.BLACK); // Set color for the text
        drawName(g2d);
    }
    private void drawHexagon(Graphics2D g2d) {
        int centerX = getWidth() / 2;
```

```
        int centerY = getHeight() / 2;
        int sideLength = 100;
        int[] xPoints = new int[6];
        int[] yPoints = new int[6];
        for (int i = 0; i < 6; i++) {
            double angle = 2 * Math.PI / 6 * i;
            xPoints[i] = (int) (centerX + sideLength * Math.cos(angle));
            yPoints[i] = (int) (centerY + sideLength * Math.sin(angle));
        }
        g2d.drawPolygon(xPoints, yPoints, 6);
    }
    private void drawCircle(Graphics2D g2d) {
        int centerX = getWidth() / 2;
        int centerY = getHeight() / 2;
        int circleRadius = 50;
        Ellipse2D circle = new Ellipse2D.Double(centerX - circleRadius, centerY - circleRadius,
                2 * circleRadius, 2 * circleRadius);
        g2d.draw(circle);
    }
    private void drawName(Graphics2D g2d) {
        int centerX = getWidth() / 2;
        int centerY = getHeight() / 2;
        g2d.setFont(new Font("Arial", Font.BOLD, 14));
        String name = "Your Name";
        int textWidth = g2d.getFontMetrics().stringWidth(name);
        int textHeight = g2d.getFontMetrics().getHeight();
        g2d.drawString(name, centerX - textWidth / 2, centerY + textHeight / 2);
    }
}
```

27.Write a javacode (application or applet) which display current Date and Time Use thread class to execute the code.

```
import java.util.Date;
public class DateTimeDisplayApp {
    public static void main(String[] args) {
        DateTimeDisplayThreaddateTimeThread = new DateTimeDisplayThread();
        Thread thread = new Thread(dateTimeThread);
        thread.start();
    }
}
class DateTimeDisplayThread implements Runnable {
    public void run() {
        try {
            while (true) {
                Date currentDate = new Date();
                System.out.println("Current Date and Time: " + currentDate);
                Thread.sleep(1000);
            }
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}
```

28.Write an applet for moving banner.

```
import java.applet.Applet;
import java.awt.*;
import java.util.Timer;
import java.util.TimerTask;
public class MovingBannerApplet extends Applet {
```

```
    private String bannerText = "Welcome to Java Applet!";
    private int xCoordinate = 0;
    private int bannerSpeed = 5; // Adjust the banner speed
    public void init() {
        Timer timer = new Timer();
        timer.scheduleAtFixedRate(new BannerTask(), 0, 100); // Adjust the delay for the banner speed
    }
    public void paint(Graphics g) {
        g.clearRect(0, 0, getWidth(), getHeight());
        g.setColor(Color.BLUE);
        g.setFont(new Font("Arial", Font.BOLD, 20));
        g.drawString(bannerText, xCoordinate, getHeight() / 2);
    }
    class BannerTask extends TimerTask {
        public void run() {
            xCoordinate += bannerSpeed;
            if (xCoordinate > getWidth()) {
                xCoordinate = -g.getFontMetrics().stringWidth(bannerText); // Reset to the left side
            }
            repaint();
        }
    }
}
```

29. Write an applet for bouncing Ball using thread.

```
import java.applet.Applet;
import java.awt.*;
public class BouncingBallApplet extends Applet implements Runnable {
    private int x, y;
    private int xSpeed = 2;
```

```
private int ySpeed = 2;
private int ballSize = 20;
private Thread animationThread;
public void init() {
   x = getWidth() / 2;
   y = getHeight() / 2;
   animationThread = new Thread(this);
   animationThread.start();
}
public void run() {
   while (true) {
      moveBall();
      repaint();
      try {
         Thread.sleep(20);
      } catch (InterruptedException e) {
         e.printStackTrace();
      }
   }
}
private void moveBall() {
   x += xSpeed;
   y += ySpeed;
   if (x < 0 || x + ballSize > getWidth()) {
      xSpeed = -xSpeed;
   }
   if (y < 0 || y + ballSize > getHeight()) {
      ySpeed = -ySpeed;
   }
}
```

```
    public void paint(Graphics g) {
        g.clearRect(0, 0, getWidth(), getHeight());
        g.setColor(Color.RED);
        g.fillOval(x, y, ballSize, ballSize);
    }
    public void stop() {
        if (animationThread != null) {
            animationThread.interrupt();
            animationThread = null;
        }
    }
}
```

30. Write a program to demonstrate communication between two applet using ActionListner.

Applet 1 Contain login page

Applet 2 Diaply Welcome message for user who is logged in Applet1

// Applet 1 (Login Applet - LoginApplet.java):

```
import java.applet.Applet;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class LoginApplet extends Applet implements ActionListener {
    private TextFieldusernameTextField, passwordTextField;
    private Button loginButton;

    public void init() {
        setLayout(new GridLayout(3, 2));

        add(new Label("Username:"));
```

```
        usernameTextField = new TextField();
        add(usernameTextField);

        add(new Label("Password:"));
        passwordTextField = new TextField();
        passwordTextField.setEchoChar('*');
        add(passwordTextField);

        loginButton = new Button("Login");
        add(loginButton);
        loginButton.addActionListener(this);
    }
    public void actionPerformed(ActionEvent e) {
        String username = usernameTextField.getText();
        WelcomeAppletwelcomeApplet = new WelcomeApplet(username);
        welcomeApplet.init();
        add(welcomeApplet);
    }
}
```

// Applet 2 (Welcome Applet - WelcomeApplet.java):

```
import java.applet.Applet;
import java.awt.*;
public class WelcomeApplet extends Applet {
    private String username;
    public WelcomeApplet(String username) {
        this.username = username;
    }
    public void init() {
        setLayout(new FlowLayout());
        Label welcomeLabel = new Label("Welcome, " + username + "!");
```

```
        add(welcomeLabel);
    }
}
```

//HTML File to embed both applets (index.html):

```
<html>
<body>
<applet code="LoginApplet.class" width="300" height="150">
</applet>
</body>
</html>
```

31.Write a program to demonstrate Applet HTML <PARAM>Tag.

```
import java.applet.Applet;
import java.awt.*;
public class ParamTagApplet extends Applet {
    private String message;
    public void init() {
        message = getParameter("message");
        if (message == null || message.isEmpty()) {
            message = "Hello, World!";
        }
    }
    public void paint(Graphics g) {
        g.drawString(message, 20, 20);
    }
}
```

32.Write an applet program that gets number of rectangles from the user using(<PARAM Tag)and draw the rectangle in different position.

```
import java.applet.Applet;
```

```
import java.awt.*;
public class RectanglesApplet extends Applet {
   private int numberOfRectangles;
   public void init() {
      String numberOfRectanglesParam = getParameter("numberOfRectangles");
      try {
         numberOfRectangles = Integer.parseInt(numberOfRectanglesParam);
      } catch (NumberFormatException e) {
         numberOfRectangles = 0;
      }
   }
   public void paint(Graphics g) {
      for (int i = 1; i <= numberOfRectangles; i++) {
         int x = (int) (Math.random() * getWidth());
         int y = (int) (Math.random() * getHeight());
         int width = (int) (Math.random() * 100);
         int height = (int) (Math.random() * 100);
         g.drawRect(x, y, width, height);
      }
   }
}
```

33. Write a program add/sub/mul/div of 2 numbers in Applet. Accept value in 2 textboxes & create button name “+”,”-“,”/”,”*”, when you click them, it will give result in third box.

```
import java.applet.Applet;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
public class CalculatorApplet extends Applet implements ActionListener {
   private TextField num1TextField, num2TextField, resultTextField;
```

```
public void init() {
    num1TextField = new TextField(10);
    num2TextField = new TextField(10);
    resultTextField = new TextField(10);
    resultTextField.setEditable(false);
    Button addButton = new Button("+");
    Button subtractButton = new Button("-");
    Button multiplyButton = new Button("*");
    Button divideButton = new Button("/");
    add(new Label("Number 1:"));
    add(num1TextField);
    add(new Label("Number 2:"));
    add(num2TextField);
    add(new Label("Result:"));
    add(resultTextField);
    add(addButton);
    add(subtractButton);
    add(multiplyButton);
    add(divideButton);
    addButton.addActionListener(this);
    subtractButton.addActionListener(this);
    multiplyButton.addActionListener(this);
    divideButton.addActionListener(this);
}
public void actionPerformed(ActionEvent e) {
    try {
        double num1 = Double.parseDouble(num1TextField.getText());
        double num2 = Double.parseDouble(num2TextField.getText());
        if (e.getActionCommand().equals("+")) {
            resultTextField.setText(String.valueOf(num1 + num2));
```

```
        } else if (e.getActionCommand().equals("-")) {
            resultTextField.setText(String.valueOf(num1 - num2));
        } else if (e.getActionCommand().equals("*")) {
            resultTextField.setText(String.valueOf(num1 * num2));
        } else if (e.getActionCommand().equals("/")) {
            if (num2 != 0) {
                resultTextField.setText(String.valueOf(num1 / num2));
            } else {
                resultTextField.setText("Cannot divide by zero");
            }
        }
    } catch (NumberFormatException ex) {
        resultTextField.setText("Invalid input");
    }
  }
}
```

34.Write an applet to run analog clock with current date & time in following format.

Tue Jan 01 2013 12.22.39

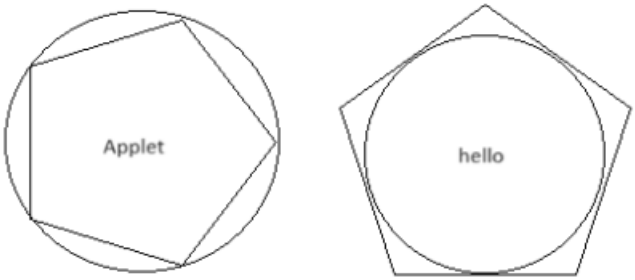
```
import java.applet.*;
import java.awt.*;
import java.util.*;
import java.text.*;
public class AnalogClock extends Applet implements Runnable {
```

```
private Thread clockThread;
public void init() {
    setBackground(Color.white);
}
public void start() {
    if (clockThread == null) {
        clockThread = new Thread(this);
        clockThread.start();
    }
}
public void run() {
    while (true) {
        repaint();
        try {
            Thread.sleep(1000); // Update every second
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}
public void paint(Graphics g) {
    Calendar calendar = Calendar.getInstance();
    Date now = calendar.getTime();
    SimpleDateFormat dateFormat = new SimpleDateFormat("E MMM dd yyyy HH.mm.ss");
    String formattedDateTime = dateFormat.format(now);
    int centerX = getWidth() / 2;
    int centerY = getHeight() / 2;
    int radius = Math.min(centerX, centerY) - 20;
    g.setColor(Color.black);
    g.drawOval(centerX - radius, centerY - radius, 2 * radius, 2 * radius);
```

```
        drawHand(g, centerX, centerY, calendar.get(Calendar.SECOND) * 6, radius - 10); // Second hand
        drawHand(g, centerX, centerY, calendar.get(Calendar.MINUTE) * 6, radius - 20); // Minute hand
        drawHand(g, centerX, centerY, (calendar.get(Calendar.HOUR) % 12 +
calendar.get(Calendar.MINUTE) / 60.0) * 30,
                radius - 40); // Hour hand
        g.setColor(Color.blue);
        g.drawString(formattedDateTime, 10, getHeight() - 10);
    }
    private void drawHand(Graphics g, int x, int y, double angle, int length) {
        double radianAngle = Math.toRadians(angle);
        int endX = x + (int) (length * Math.sin(radianAngle));
        int endY = y - (int) (length * Math.cos(radianAngle));
        g.drawLine(x, y, endX, endY);
    }
}
```

```
<!DOCTYPE html>
<html>
<head>
    <title>Analog Clock Applet</title>
</head>
<body>
    <applet code="AnalogClock.class" width="400" height="400">
    </applet>
</body>
</html>
```

35.Write an applet program to print name in following pictures & fill different colors in picture




```
import java.awt.*;
import java.applet.*;
public class applet extends Applet {
    int x1[] = { 115, 215, 300, 215, 115 };
    int y1[] = { 150, 100, 200, 297, 250 };
    int n1 = 5;
    int x2[] = { 500, 600, 550, 450, 405 };
    int y2[] = { 70, 150, 250, 250, 150 };
    int n2 = 5;

    public void paint(Graphics g) {
        g.setColor(Color.red);
        g.fillOval(100, 100, 200, 200);
        g.setColor(Color.yellow);
        g.fillPolygon(x1, y1, n1);
        g.setColor(Color.red);
        g.fillPolygon(x2, y2, n2);
        g.setColor(Color.black);
        g.drawString("Applet", 180, 200);
        g.setColor(Color.yellow);
        g.fillOval(425, 93, 155, 155);
        g.setColor(Color.black);
        g.drawString("Hello", 485, 180);
```

```
    }
}
```

36.Write a package that shows all combinations of the access control modifiers.

```
package accessmodifiersdemo;
public class PublicClass {
    public void display() {
        System.out.println("PublicClass: Public Method");
    }
}
class DefaultClass {
    void display() {
        System.out.println("DefaultClass: Default (Package-Private) Method");
    }
}
class BaseClass {
    protected void display() {
        System.out.println("BaseClass: Protected Method");
    }
}
class SubClass extends BaseClass {
    void invokeProtectedMethod() {
        display(); // Can access protected method from superclass
    }
}
class PrivateClass {
    private void display() {
        System.out.println("PrivateClass: Private Method");
    }
```

```
    void invokePrivateMethod() {
        display();
    }
}
public interface PublicInterface {
    void display();
}
class InterfaceImplementation implements PublicInterface {
    public void display() {
        System.out.println("InterfaceImplementation: Public Method from Interface");
    }
}
```