Department of Computer Application

National Institute of Technology Kurukshetra, Kurukshetra -136119

Object Oriented Programming in JAVA

MCA-136

(2024-27)

Submitted by:

***Name**: Shrushranto Rajbongshi*

***Roll no**: 524110026*

***Semester**: 2nd*

***Section:** Group 2*

Submitted to:

***Name**: Dr. Sarika Jain*

***Date**: 2-05-2025*

DECLARATION

I, Shrushranto Rajbongshi, bearing Roll No. 524110026, hereby declare that the **Java Programming Lab Manual** submitted by me is the result of my own independent and original work. This manual reflects not only the learning objectives of the lab but also my personal effort, analytical reasoning, and problem-solving skills developed throughout the course.

Each program in this manual has been written, compiled, and executed by me using standard Java development tools. The implementations demonstrate my understanding of core Java concepts including data types, control structures, object-oriented programming, exception handling, multithreading, GUI development, and file handling.

While I have referred to classroom lectures, official documentation, and prescribed textbooks for conceptual clarity, all source codes, program structures, and logic flow have been developed independently. I have refrained from copying or replicating solutions and instead approached each task with original logic and coding practices.

This lab manual is submitted in full compliance with the academic regulations of the **National Institute of Technology, Kurukshetra**, and stands as an honest representation of my efforts and learning in the Java Programming Laboratory.

Date: 30-04-2025
Signature: *Shrushranto Rajbongshi*

Contents

DECLARATION 2

Week 1 (To do some basic programs in JAVA) 5

1: Write a Java program to display the message “This is my first java class”. Execute this program through the command line using javac and java. 5

2: Write a Java program to display the message “This is just a test”. Pass these 5 strings as arguments in the main function while executing the program through cmd. 5

3: Write a Java Program to make a frequency count of words in a given text entered by the user. 6

4: Write a Java program that prompts the user for an integer and then prints out all prime numbers up to that integer. (use Scanner class to read input). 7

5: Write a Java program to multiply two given matrices. 8

6: Write a Java program to find the Fibonacci series using recursive and non-recursive functions. 9

Week 2 (To study Data types, Scope, and a lifetime of variables, operators, expressions, and control statements) 10

1: Write a Java program to identify the symbols of expression given by the user is the operator or of which datatype. 10

2: Write a Java program to make a class A in that class declare function add perform addition through main. 11

3: Write a Java program to print prime numbers from 1 to 50 using *while* & *do-while* loops. 11

4: Write a Java program to print Diamond the following structure using *for* loop. 12

5: Write a Java program to find the factorial of given number using command line. 13

6: Write a Java program to find the volume of the cuboid using *this* pointer 14

7: Write a Java program to swap the value of the speed of bikes by making class *bike* using call by value and call by reference 15

8: Write a Java program to find whether the number is palindrome or not using recursion. 16

9: Write a Java program to print all the substrings of the given string “CODING”. 17

Week 3 (To study the Classes and Objects in OOPJ) 18

1: Write a Java program to create a class Rectangle with data members length and breadth. Create a method area () that finds the area of the rectangle. Use the constructor(s) to assign value to data members. Use “this” in a parameterized constructor. 18

2: Print the sum, difference, and product of two complex numbers by creating a class named 'Complex' with separate methods for each operation whose real and imaginary parts are entered by the user. 19

3: Write a Java program to create a class Box with data members length, breadth, and height. Create multiple constructors to assign values to objects in different ways. Use the overridden “equals” method to compare objects, if found equal then display the objects using overridden to string method. 20

Week 4 (To study inheritance in Java) 21

1: Write a Java program to identify the accessibility of a variable by means of different access specifiers within and outside the package. 21

2: Write an employee class Marketer to accompany the other employees. Marketers make $50,000 ($10,000 more than general employees), and they have an additional method named advertise that prints "Act now, while supplies last!" Use the super keyword to interact with the Employee superclass as appropriate. 22

3: Create a base class Person and two derived classes as Student and Teacher with their constructors and methods. Assume the student to be in the same package as that of Person and Teacher class to be in a different package. 23

4: Describe abstract class called Shape which has three subclasses say Triangle, Rectangle, Circle. 25

5: Assume that a bank maintains two kinds of accounts for its customers. 26

Week 5 (To study Interfaces and Exception Handling in Java) 28

1: Write a Java program to compute the area of circle and rectangle using Interfaces. 28

2: Write a Java program that uses interface for the implementation of fixed-size and dynamic-size stacks (for dynamic stack size should be redefined as per the number of elements). 29

3: Write a Java program for exception handling with StringIndexOutOfBound exception. 31

4: An array is declared with 5 elements. Then the code tries to access the 6th element of the array which throws an exception. Write the program for this. 32

5: Write a suitable program for the following conditions: 32

Week 6 (To study about Multi-Threading in Java) 36

1: Suppose there are 5 workers who carry out work at the same time. Develop a program that shows this scenario with the concept of multithreading. 36

2: Write a Java program that creates three threads. The first thread displays “Good Morning” every second, the second thread displays “Hello” every two seconds, and the third thread displays “Welcome” every three seconds. 37

3: Implement a class that checks whether a given number is prime using both the Thread class and Runnable interface. 38

4: Write a Java program that correctly implements the producer-consumer problem using the concept of inter-thread communication. 39

Week 7 (To study Strings in Java) 41

1: Write a Java program to find whether a given string is palindrome or not. 41

2: Write a method that will remove given character from the String. 42

3: Write a Java program for sorting a given list of names. 42

4: Write a Java program that computes your initials from your full name and displays them. 43

5: An anagram is a word or a phrase made by transposing the letters of another word or phrase; 44

Week 8 (To study Event Handling & AWT in Java) 45

1: Write a Java program that handles all mouse events and shows the event name at the centre of the window when a mouse event is fired (Use Adapter classes). 45

2: Write a Java program for handling Key events. 46

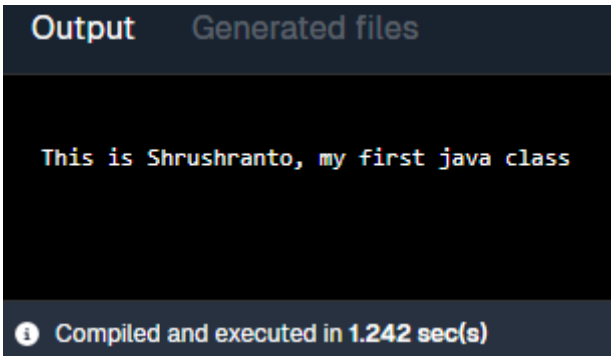
3: Write a java program that simulates a traffic light. 47

..... 49

Week 1 (To do some basic programs in JAVA)

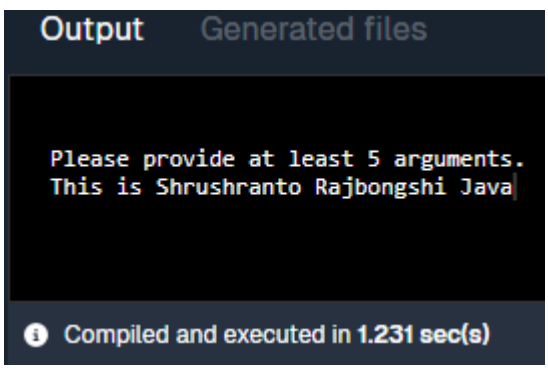
1: Write a Java program to display the message “This is my first java class”. Execute this program through the command line using javac and java.

```
public class FirstJavaClass {
    public static void main(String[] args) {
        System.out.println("This is Shrushranto, my first java class");
    }
}
```



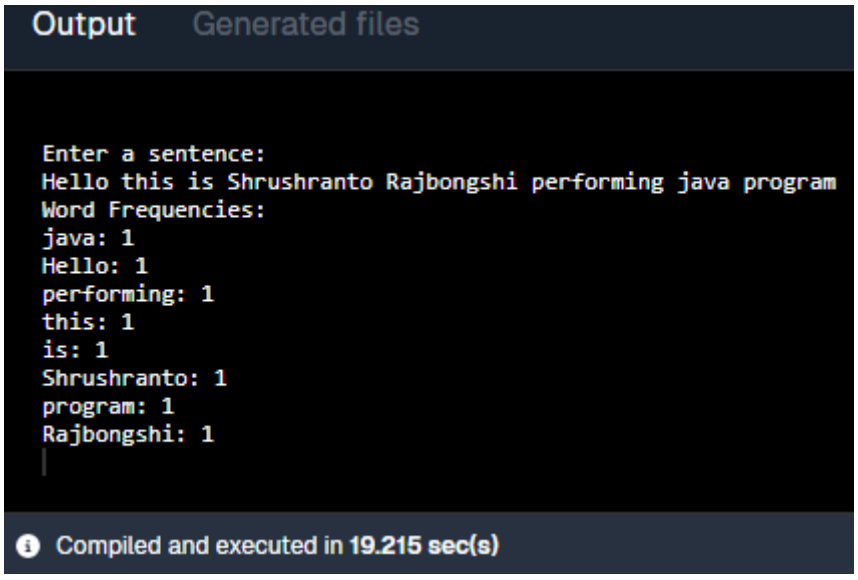
2: Write a Java program to display the message “This is just a test”. Pass these 5 strings as arguments in the main function while executing the program through cmd.

```
public class TestMessage {
  public static void main(String[] args) {
    if (args.length >= 5) {
      System.out.println("This is just a test, performed by Shrushranto");
    } else {
      System.out.println("Please provide at least 5 arguments.");
    }
  }
}
```



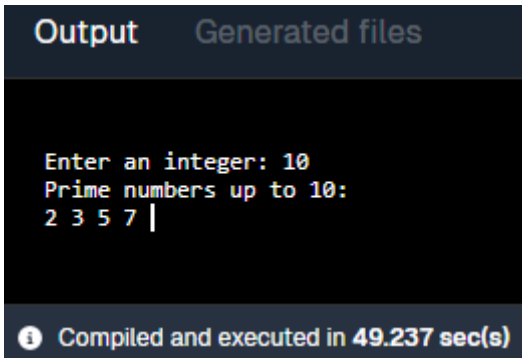
3: Write a Java Program to make a frequency count of words in a given text entered by the user.

```
import java.util.*;
public class WordFrequency {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter a sentence:");
        String text = sc.nextLine();
        String[] words = text.split("\\s+");
        Map<String, Integer> freq = new HashMap<>();
        for (String word : words) {
            freq.put(word, freq.getOrDefault(word, 0) + 1);
        }
        System.out.println("Word Frequencies:");
        for (String key : freq.keySet()) {
            System.out.println(key + ": " + freq.get(key));
        }
    }
}
```



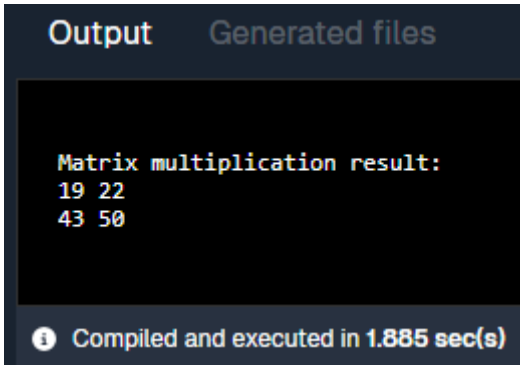
4: Write a Java program that prompts the user for an integer and then prints out all prime numbers up to that integer. (use Scanner class to read input).

```
import java.util.Scanner;
public class PrimeNumbers {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter an integer: ");
        int n = sc.nextInt();
        System.out.println("Prime numbers up to " + n + ":");
        for (int i = 2; i <= n; i++) {
            if (isPrime(i)) {
                System.out.print(i + " ");
            }
        }
    }
    public static boolean isPrime(int num) {
        if (num <= 1) return false;
        for (int i = 2; i <= num / 2; i++) {
            if (num % i == 0) return false;
        }
        return true;
    }
}
```



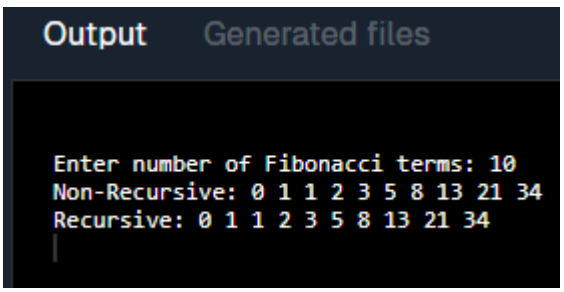
5: Write a Java program to multiply two given matrices.

```
public class MatrixMultiplication {
    public static void main(String[] args) {
        int[][] a = { {1, 2}, {3, 4} };
        int[][] b = { {5, 6}, {7, 8} };
        int[][] result = new int[2][2];
        for (int i = 0; i < 2; i++) {
            for (int j = 0; j < 2; j++) {
                result[i][j] = 0;
                for (int k = 0; k < 2; k++) {
                    result[i][j] += a[i][k] * b[k][j];
                }
            }
        }
        System.out.println("Matrix multiplication result:");
        for (int[] row : result) {
            for (int val : row) {
                System.out.print(val + " ");
            }
            System.out.println();
        }
    }
}
```



6: Write a Java program to find the Fibonacci series using recursive and non-recursive functions.

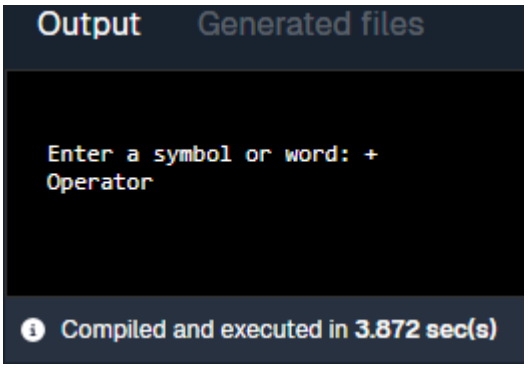
```
import java.util.Scanner;
public class FibonacciSeries {
    // Recursive
    public static int fibRecursive(int n) {
        if (n <= 1) return n;
        return fibRecursive(n - 1) + fibRecursive(n - 2);
    }
    // Non-recursive
    public static void fibNonRecursive(int n) {
        int a = 0, b = 1;
        System.out.print("Non-Recursive: ");
        for (int i = 0; i < n; i++) {
            System.out.print(a + " ");
            int next = a + b;
            a = b;
            b = next;
        }
        System.out.println();
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter number of Fibonacci terms: ");
        int n = sc.nextInt();
        fibNonRecursive(n);
        System.out.print("Recursive: ");
        for (int i = 0; i < n; i++) {
            System.out.print(fibRecursive(i) + " ");
        }
        System.out.println();
    }
}
```



Week 2 (To study Data types, Scope, and a lifetime of variables, operators, expressions, and control statements)

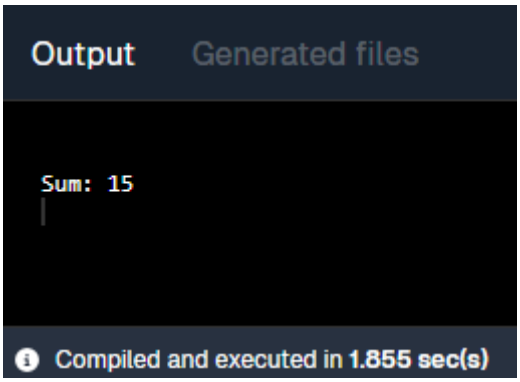
1: Write a Java program to identify the symbols of expression given by the user is the operator or of which datatype.

```
import java.util.Scanner;
public class SymbolIdentifier {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter a symbol or word: ");
        String input = sc.next();
        switch (input) {
            case "+": case "-": case "*": case "/": case "%":
                System.out.println("Operator");
                break;
            case "int": case "float": case "double": case "char": case "boolean":
                System.out.println("Data type");
                break;
            default:
                System.out.println("Unknown symbol");
        }
    }
}
```



2: Write a Java program to make a class A in that class declare function add perform addition through main.

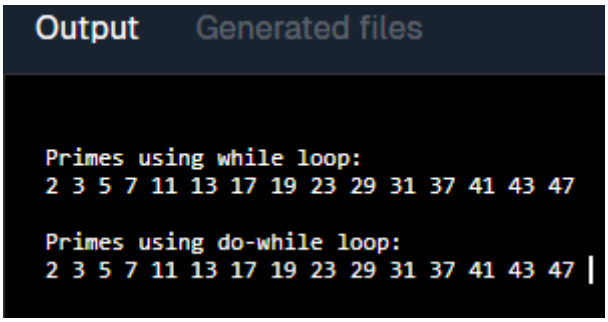
```
public class A {
    void add(int a, int b) {
        System.out.println("Sum: " + (a + b));
    }

    public static void main(String[] args) {
        A obj = new A();
        obj.add(5, 10);
    }
}
```



3: Write a Java program to print prime numbers from 1 to 50 using *while* & *do-while* loops.

```
public class PrimeWithLoops {
    public static void main(String[] args) {
        int i = 2;
        System.out.println("Primes using while loop:");
        while (i <= 50) {
            if (isPrime(i)) System.out.print(i + " ");
            i++;
        }
        System.out.println("\n\nPrimes using do-while loop:");
        i = 2;
        do {
            if (isPrime(i)) System.out.print(i + " ");
            i++;
        } while (i <= 50);
```

```
    }
    public static boolean isPrime(int n) {
        if (n < 2) return false;
        for (int j = 2; j <= n / 2; j++) {
            if (n % j == 0) return false;
        }
        return true;
    }
}
```

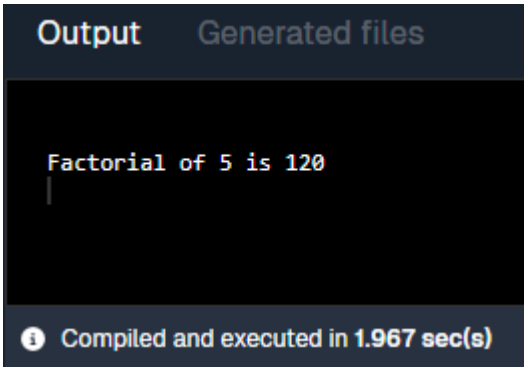


4: Write a Java program to print Diamond the following structure using *for* loop.

```
public class DiamondPattern {
    public static void main(String[] args) {
        int n = 5;  // Half height
        for (int i = 1; i <= n; i++) {
            for (int j = n - i; j > 0; j--) System.out.print(" ");
            for (int j = 1; j <= 2 * i - 1; j++) System.out.print("*");
            System.out.println();
        }
        for (int i = n - 1; i >= 1; i--) {
            for (int j = n - i; j > 0; j--) System.out.print(" ");
            for (int j = 1; j <= 2 * i - 1; j++) System.out.print("*");
            System.out.println();
        }
    }
}
```

5: Write a Java program to find the factorial of given number using command line.

```
public class Test {
    static int factorial(int n)
    {
        int res = 1, i;
        for (i = 2; i <= n; i++)
            res *= i;
        return res;
    }
    public static void main(String[] args)
    {
        int num = 5;
        System.out.println("Factorial of " + num + " is "
                           + factorial(5));
    }
}
```

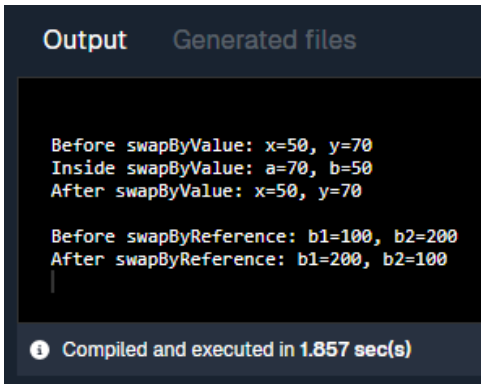


6: Write a Java program to find the volume of the cuboid using *this* pointer.

```
public class Cuboid {
    int length, width, height;
    Cuboid(int length, int width, int height) {
        this.length = length;
        this.width = width;
        this.height = height;
    }
    int volume() {
        return this.length * this.width * this.height;
    }
    public static void main(String[] args) {
        Cuboid c = new Cuboid(4, 3, 2);
        System.out.println("Volume: " + c.volume());
    }
}
```

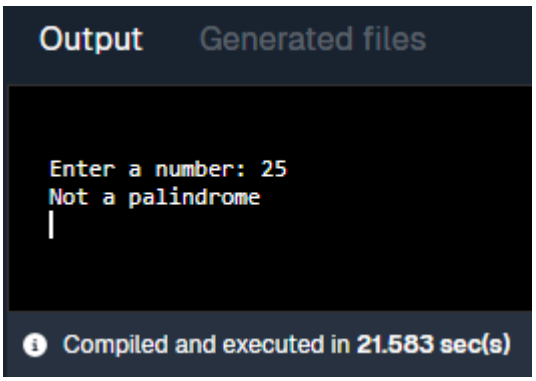
7: Write a Java program to swap the value of the speed of bikes by making class *bike* using call by value and call by reference

```
public class Bike {
    int speed;
    Bike(int speed) {
        this.speed = speed;
    }
    static void swapByValue(int a, int b) {
        int temp = a;
        a = b;
        b = temp;
        System.out.println("Inside swapByValue: a=" + a + ", b=" + b);
    }
    static void swapByReference(Bike b1, Bike b2) {
        int temp = b1.speed;
        b1.speed = b2.speed;
        b2.speed = temp;
    }
    public static void main(String[] args) {
        int x = 50, y = 70;
        System.out.println("Before swapByValue: x=" + x + ", y=" + y);
        swapByValue(x, y);
        System.out.println("After swapByValue: x=" + x + ", y=" + y);
        Bike b1 = new Bike(100);
        Bike b2 = new Bike(200);
        System.out.println("\nBefore swapByReference: b1=" + b1.speed + ", b2=" +
b2.speed);
        swapByReference(b1, b2);
        System.out.println("After swapByReference: b1=" + b1.speed + ", b2=" + b2.speed);
    }
}
```



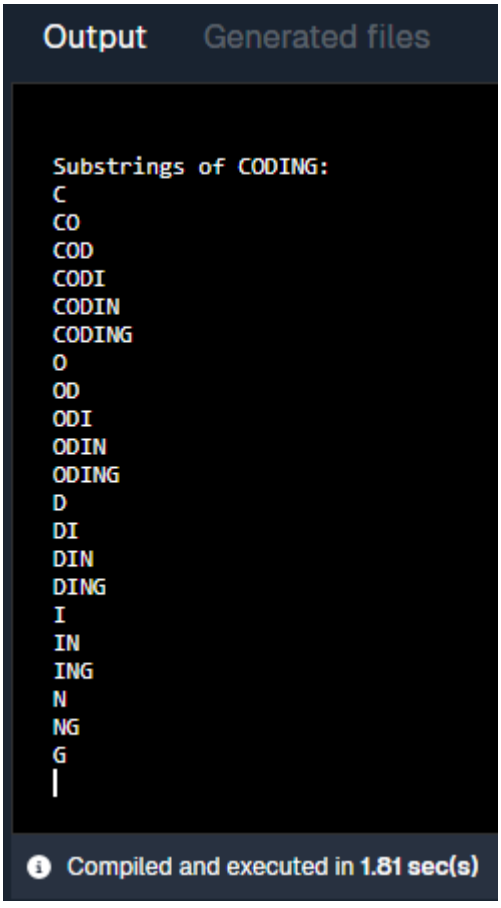
8: Write a Java program to find whether the number is palindrome or not using recursion.

```
import java.util.Scanner;
public class PalindromeRecursion {
    static boolean isPalindrome(String s, int start, int end) {
        if (start >= end) return true;
        if (s.charAt(start) != s.charAt(end)) return false;
        return isPalindrome(s, start + 1, end - 1);
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter a number: ");
        String input = sc.next();
        if (isPalindrome(input, 0, input.length() - 1))
            System.out.println("Palindrome");
        else
            System.out.println("Not a palindrome");
    }
}
```



9: Write a Java program to print all the substrings of the given string “CODING”.

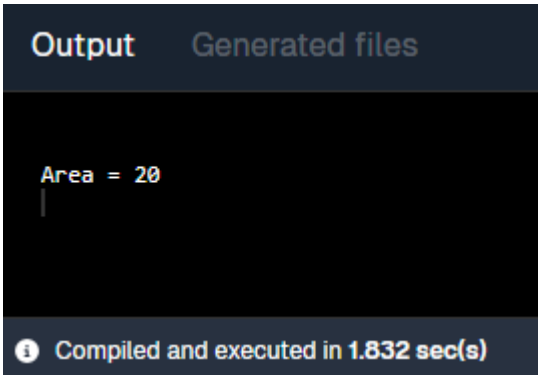
```
public class Substrings {
    public static void main(String[] args) {
        String str = "CODING";
        int len = str.length();
        System.out.println("Substrings of CODING:");
        for (int i = 0; i < len; i++) {
            for (int j = i + 1; j <= len; j++) {
                System.out.println(str.substring(i, j));
            }
        }
    }
}
```



Week 3 (To study the Classes and Objects in OOPJ)

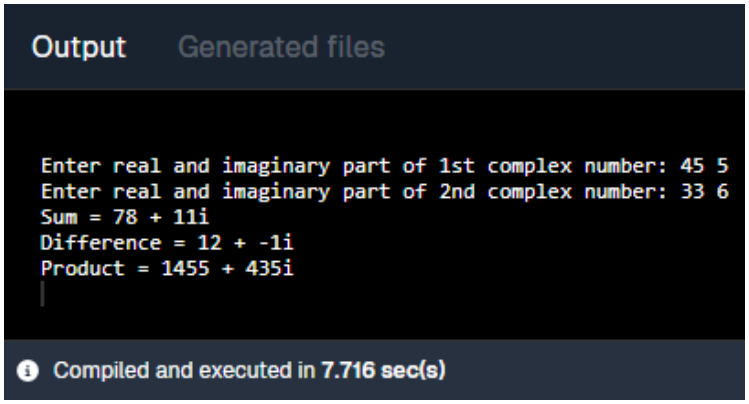
1: Write a Java program to create a class Rectangle with data members length and breadth. Create a method area () that finds the area of the rectangle. Use the constructor(s) to assign value to data members. Use “this” in a parameterized constructor.

```
class Rectangle {
  int length, breadth;
  Rectangle(int length, int breadth) {
    this.length = length;
    this.breadth = breadth;
  }
  int area() {
    return length * breadth;
  }
  public static void main(String[] args) {
    Rectangle r = new Rectangle(5, 4);
    System.out.println("Area = " + r.area());
  }
}
```



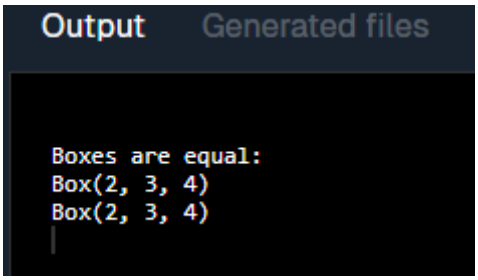
2: Print the sum, difference, and product of two complex numbers by creating a class named 'Complex' with separate methods for each operation whose real and imaginary parts are entered by the user.

```
import java.util.Scanner;
public class Complex {
    int real, imag;
    Complex(int r, int i) {
        real = r;
        imag = i;
    }
    void sum(Complex c) {
        System.out.println("Sum = " + (real + c.real) + " + " + (imag + c.imag) + "i");
    }
    void diff(Complex c) {
        System.out.println("Difference = " + (real - c.real) + " + " + (imag - c.imag) + "i");
    }
    void prod(Complex c) {
        int r = real * c.real - imag * c.imag;
        int i = real * c.imag + imag * c.real;
        System.out.println("Product = " + r + " + " + i + "i");
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter real and imaginary part of 1st complex number: ");
        Complex c1 = new Complex(sc.nextInt(), sc.nextInt());
        System.out.print("Enter real and imaginary part of 2nd complex number: ");
        Complex c2 = new Complex(sc.nextInt(), sc.nextInt());
        c1.sum(c2);
        c1.diff(c2);
        c1.prod(c2);
    }
}
```



3: Write a Java program to create a class Box with data members length, breadth, and height. Create multiple constructors to assign values to objects in different ways. Use the overridden "equals" method to compare objects, if found equal then display the objects using overridden to string method.

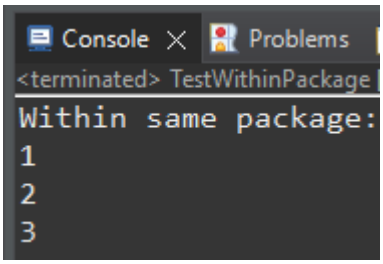
```
class Box {
    int l, b, h;
    Box() {
        l = b = h = 1;
    }
    Box(int x) {
        l = b = h = x;
    }
    Box(int l, int b, int h) {
        this.l = l; this.b = b; this.h = h;
    }
    public boolean equals(Object o) {
        Box other = (Box) o;
        return l == other.l && b == other.b && h == other.h;
    }
    public String toString() {
        return "Box(" + l + ", " + b + ", " + h + ")";
    }
    public static void main(String[] args) {
        Box b1 = new Box(2, 3, 4);
        Box b2 = new Box(2, 3, 4);
        if (b1.equals(b2)) {
            System.out.println("Boxes are equal:");System.out.println(b1);
            System.out.println(b2);
        } else {
            System.out.println("Boxes are not equal");
        }
    }
}
```



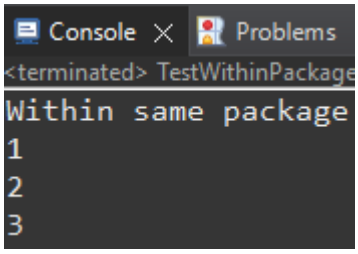
Week 4 (To study inheritance in Java)

1: Write a Java program to identify the accessibility of a variable by means of different access specifiers within and outside the package.

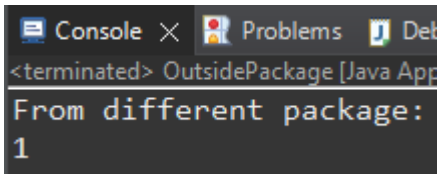
```
package mypack;
public class AccessTest {
    public int pub = 1;
    protected int prot = 2;
    int def = 3;
    private int priv = 4;
    public void show() {
        System.out.println("Inside same class:");
        System.out.println("Public: " + pub);
        System.out.println("Protected: " + prot);
        System.out.println("Default: " + def);
        System.out.println("Private: " + priv);
    }
}
```




```
package mypack;

public class TestWithinPackage {
    public static void main(String[] args) {
        AccessTest at = new AccessTest();
        System.out.println("Within same package:");
        System.out.println(at.pub);
        System.out.println(at.prot);
        System.out.println(at.def);
    }
}
```

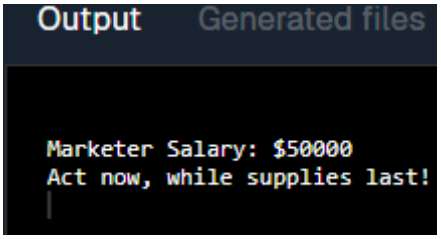


```
package com.studyjava;
import mypack.AccessTest;
public class OutsidePackage {
    public static void main(String[] args) {
        AccessTest at = new AccessTest();
        System.out.println("From different package:");
        System.out.println(at.pub);
    }
}
```



2: Write an employee class Marketer to accompany the other employees. Marketers make $50,000 ($10,000 more than general employees), and they have an additional method named advertise that prints "Act now, while supplies last!" Use the super keyword to interact with the Employee superclass as appropriate.

```
class Employee {
    int getSalary() {
        return 40000;
    }
}
public class Marketer extends Employee {
    int getSalary() return super.getSalary() + 10000;
    void advertise() {
        System.out.println("Act now, while supplies last!");
    }
    public static void main(String[] args) {
        Marketer m = new Marketer();
        System.out.println("Marketer Salary: $" + m.getSalary());
        m.advertise();
    }
}
```



3: Create a base class Person and two derived classes as Student and Teacher with their constructors and methods. Assume the student to be in the same package as that of Person and Teacher class to be in a different package.

The inheritance hierarchy would appear as follows:

a) Add methods "get" the instance variables in the Person class. These would consist of: getName, getAge, getGender.

b) Add methods to "set" and "get" the instance variables in the Student class. These would consist of: getIdNum, getGPA, setIdNum.

c) Write a Teacher class that extends the parent class Person.

Person.java

```
package mypack;
public class Person {
    protected String name;
    protected int age;
    protected String gender;
    public Person(String name, int age, String gender) {
        this.name = name;
        this.age = age;
        this.gender = gender;
    }
    public String getName() { return name; }
    public int getAge() { return age; }
    public String getGender() { return gender; }
}
```

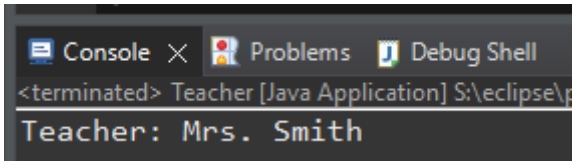
Student.java

```
package mypack;
public class Student extends Person {
    private String idNum;
    private double GPA;
    public Student(String name, int age, String gender, String idNum, double GPA) {
        super(name, age, gender);
        this.idNum = idNum;
        this.GPA = GPA;
    }
```

```
    public String getIdNum() { return idNum; }
    public void setIdNum(String id) { idNum = id; }
    public double getGPA() { return GPA; }
    public static void main(String[] args) {
        Student s = new Student("John", 20, "Male", "S101", 3.8);
        System.out.println("Name: " + s.getName());
        System.out.println("ID: " + s.getIdNum());
    }
}
```

Teacher.java

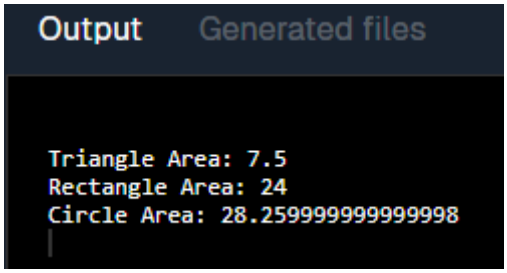
```
package com.studyjava;
import mypack.Person;
public class Teacher extends Person {
    public Teacher(String name, int age, String gender) {
        super(name, age, gender);
    }
    public void display() {
        System.out.println("Teacher: " + getName());
    }
    public static void main(String[] args) {
        Teacher t = new Teacher("Mrs. Smith", 40, "Female");
        t.display();
    }
}
```



4: Describe abstract class called Shape which has three subclasses say Triangle, Rectangle, Circle.

Define one method area() in the abstract class and override this area() in these three subclasses to calculate for specific object i.e. area() of Triangle subclass should calculate area of triangle etc. Same should be for Rectangle and Circle.

```
abstract class Shape {
    abstract void area();
}
class Triangle extends Shape {
    void area() {
        int b = 5, h = 3;
        System.out.println("Triangle Area: " + (0.5 * b * h));
    }
}
class Rectangle extends Shape {
    void area() {
        int l = 4, b = 6;System.out.println("Rectangle Area: " + (l * b));
    }
}
class Circle extends Shape {
    void area() {
        int r = 3; System.out.println("Circle Area: " + (3.14 * r * r));
    }
}
public class TestShape {
    public static void main(String[] args) {
        new Triangle().area();
        new Rectangle().area();
        new Circle().area();
    }
}
```



5: Assume that a bank maintains two kinds of accounts for its customers.

one called savings account and the other current account. The savings account provides compound interest and withdrawal facilities but no cheque book facility. The current account provides cheque book facility but no interest. Current account holders should also maintain a minimum balance and if the balance falls below this level, a service charge is imposed.

Create a class Account that stores customer name, account number and type of account. From this derive the classes Curr-acct and Sav-acct to make them more specific to their requirements. Include the necessary methods in order to achieve the following tasks:

. Accept deposit from a customer and update the balance.

. Display the balance.

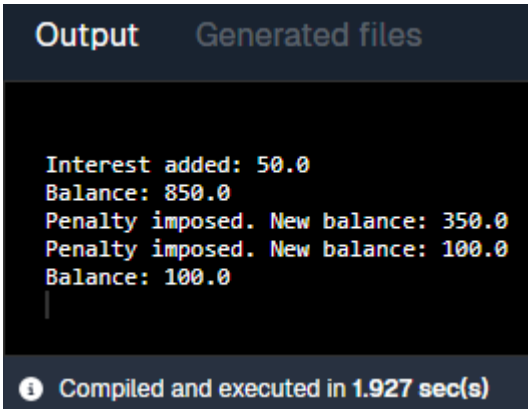
. Compute and deposit interest.

. Permit withdrawal and update the balance.

. Check for the minimum balance, impose penalty, if necessary and update the balance.

```
class Account {
    String name;
    int accNo;
    double balance;
    void deposit(double amt) {
        balance += amt;
    }
    void displayBalance() {
        System.out.println("Balance: " + balance);
    }
}
class SavAcc extends Account {
    void computeInterest() {
        double interest = balance * 0.05; deposit(interest);
        System.out.println("Interest added: " + interest);
    }

    void withdraw(double amt) {
```

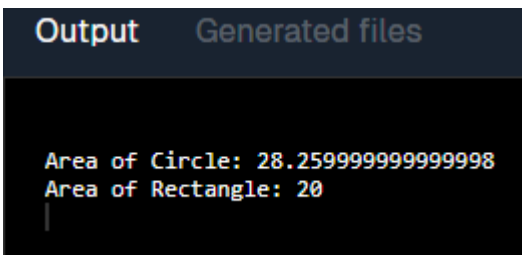
```
        if (amt <= balance) balance -= amt;
        else System.out.println("Insufficient funds.");
    }
}
class CurrAcc extends Account {
    void withdraw(double amt) {
        if (amt <= balance) balance -= amt;
        else System.out.println("Insufficient funds."); checkMinimum();
    }
    void checkMinimum() {
        if (balance < 500) {
            balance -= 50; // penalty
            System.out.println("Penalty imposed. New balance: " + balance);
        }
    }
}
public class Bank {
    public static void main(String[] args) {
        SavAcc s = new SavAcc();
        s.name = "Alice";
        s.accNo = 101;
        s.deposit(1000);
        s.computeInterest();
        s.withdraw(200);
        s.displayBalance();
        CurrAcc c = new CurrAcc();
        c.name = "Bob";
        c.accNo = 102;
        c.deposit(600);
        c.withdraw(200);
        c.withdraw(200);
        c.displayBalance();
    }
}
```



Week 5 (To study Interfaces and Exception Handling in Java)

1: Write a Java program to compute the area of circle and rectangle using Interfaces.

```
interface Shape {
    void area();
}
class Circle implements Shape {
    public void area() {
        double r = 3.0;
        System.out.println("Area of Circle: " + (3.14 * r * r));
    }
}
class Rectangle implements Shape {
    public void area() {
        int l = 5, b = 4;
        System.out.println("Area of Rectangle: " + (l * b));
    }
}
public class TestShapes {
    public static void main(String[] args) {
        Shape c = new Circle();
        Shape r = new Rectangle();
        c.area();
        r.area();
    }
}
```



2: Write a Java program that uses interface for the implementation of fixed-size and dynamic-size stacks (for dynamic stack size should be redefined as per the number of elements).

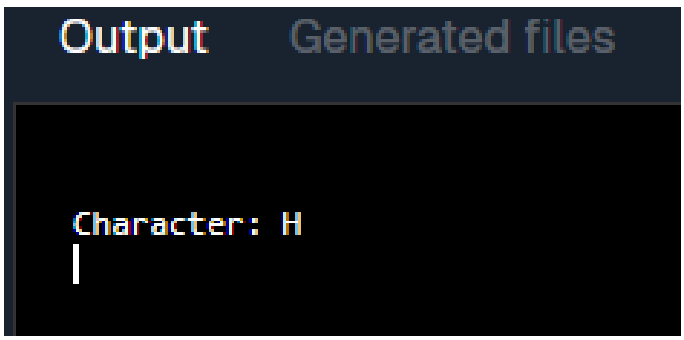
```
interface Stack {
    void push(int val);
    void pop();
}
class FixedStack implements Stack {
    int[] stack = new int[5];
    int top = -1;
    public void push(int val) {
        if (top == 4) System.out.println("Stack Overflow");
        else stack[++top] = val;
    }
    public void pop() {
        if (top == -1) System.out.println("Stack Underflow");
        else System.out.println("Popped: " + stack[top--]);
    }
}
class DynamicStack implements Stack {
    int[] stack = new int[2];
    int top = -1;
    public void push(int val) {
        if (top == stack.length - 1) {
            int[] newStack = new int[stack.length * 2];
            System.arraycopy(stack, 0, newStack, 0, stack.length);
            stack = newStack;
        }
        stack[++top] = val;
    }
    public void pop() {
        if (top == -1) System.out.println("Stack Underflow");
        else System.out.println("Popped: " + stack[top--]);
    }
}
```

```
public class TestStack {
    public static void main(String[] args) {
        Stack fs = new FixedStack();
        Stack ds = new DynamicStack();
        fs.push(10); fs.push(20); fs.push(30);
        fs.pop(); fs.pop();
        ds.push(1); ds.push(2); ds.push(3);
        ds.pop(); ds.pop();
    }
}
```

3: Write a Java program for exception handling with StringIndexOutOfBound exception.

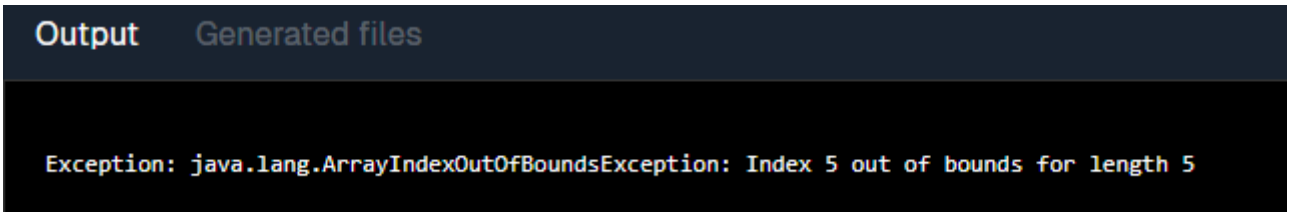
- Create an object of the class having StringIndexOutOfBound exception whenever an index is invoked of a string, which is not in the range.
- Each character of a string object is stored in a particular index starting from 0.
- To get a character present in a particular index of a string you can use a method charAt(int) of java.lang.String where int argument is the index.

```
public class StringIndexDemo {
    public static void main(String[] args) {
        String str = "HELLO";
        try {
            char ch = str.charAt(5); // Invalid index
            System.out.println("Character: " + ch);
        } catch (StringIndexOutOfBoundsException e) {
            System.out.println("Exception: " + e);
        }
    }
}
```



4: An array is declared with 5 elements. Then the code tries to access the 6th element of the array which throws an exception. Write the program for this.

```
public class ArrayExceptionDemo {
    public static void main(String[] args) {
        int[] arr = {1, 2, 3, 4, 5};
        try {
            System.out.println("6th element: " + arr[5]);
        } catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("Exception: " + e);
        }
    }
}
```



5: Write a suitable program for the following conditions:

a) A try block followed by multiple catch blocks

b) Catching multiple type of exceptions

c) Using throws/throw keywords

d) Using finally block

e) Using try-with-resources

f) User-defined exceptions

```
import java.io.*;
// a) Try block with multiple catch blocks
class MultipleCatch {
    public static void run() {
        try {
            int a = 5 / 0;
            int[] arr = new int[2];
            arr[5] = 10;
        } catch (ArithmeticException e) {
```

```
            System.out.println("Arithmetic error!");
        } catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("Array index error!");
        }
    }
}
// b) Catching multiple types of exceptions
class MultiCatch {
    public static void run() {
        try {
            String s = null;
            System.out.println(s.length());
        } catch (NullPointerException | ArithmeticException e) {
            System.out.println("Caught exception: " + e);
        }
    }
}
// c) Using throw/throws
class ThrowExample {
    static void checkAge(int age) throws Exception {
        if (age < 18)
            throw new Exception("Not eligible to vote");
        else
            System.out.println("Eligible to vote");
    }
    public static void run() {
        try {
            checkAge(16);
        } catch (Exception e) {
            System.out.println("Caught: " + e.getMessage());
        }
    }
}
```

```
// d) Using finally block
class FinallyBlock {
    public static void run() {
        try {
            int a = 10 / 0;
        } catch (ArithmeticException e) {
            System.out.println("Error: " + e);
        } finally {
            System.out.println("Finally block executed");
        }
    }
}
// e) Try-with-resources (reads from a file)
class TryWithResources {
    public static void run() {
        try (BufferedReader br = new BufferedReader(new FileReader("test.txt"))) {
            System.out.println("First line: " + br.readLine());
        } catch (IOException e) {
            System.out.println("IO Error: " + e);
        }
    }
}
// f) User-defined exception
class MyException extends Exception {
    MyException(String msg) {
        super(msg);
    }
}
class CustomExceptionDemo {
    public static void run() {
        try {
            int marks = 30;
            if (marks < 40) throw new MyException("Marks too low!");
            else System.out.println("Passed");
        } catch (MyException e) {
```

```
            System.out.println("Caught: " + e.getMessage());
        }
    }
}
// Main class to run all demos
public class ExceptionHandlingDemo {
    public static void main(String[] args) {
        System.out.println("a) Multiple Catch Blocks:");
        MultipleCatch.run();
        System.out.println("\nb) Multi-type Catch:");
        MultiCatch.run();
        System.out.println("\nc) Throw/Throws:");
        ThrowExample.run();
        System.out.println("\nd) Finally Block:");
        FinallyBlock.run();
        System.out.println("\ne) Try-with-Resources:");
        TryWithResources.run(); // Make sure test.txt exists
        System.out.println("\nf) User-defined Exception:");
        CustomExceptionDemo.run();
    }
```

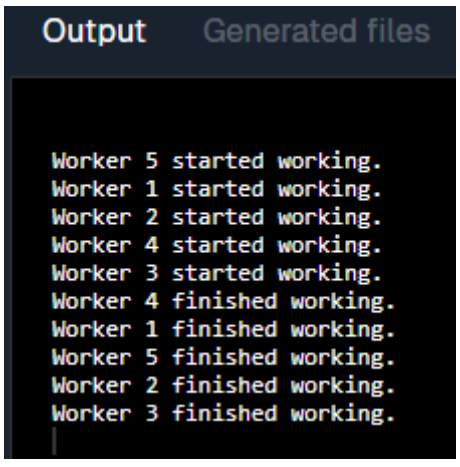
Output Generated files

```
a) Multiple Catch Blocks:
Arithmetic error!

b) Multi-type Catch:
Caught exception: java.lang.NullPointerException: Cannot invoke "String.length()" because "<local0>"

c) Throw/Throws:
Caught: Not eligible to vote

d) Finally Block:
Error: java.lang.ArithmeticException: / by zero
Finally block executed

e) Try-with-Resources:
IO Error: java.io.FileNotFoundException: test.txt (No such file or directory)

f) User-defined Exception:
Caught: Marks too low!
```

Week 6 (To study about Multi-Threading in Java)

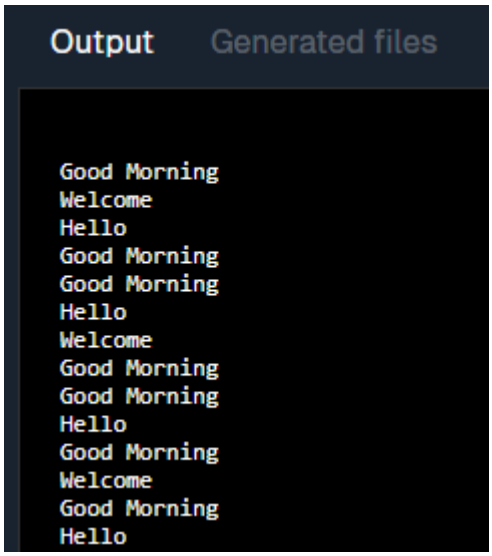
1: Suppose there are 5 workers who carry out work at the same time. Develop a program that shows this scenario with the concept of multithreading.

```
public class Worker extends Thread {
    private String name;
    Worker(String name) {
        this.name = name;
    }
    public void run() {
        System.out.println(name + " started working.");
        try {
            Thread.sleep(1000);
        } catch (InterruptedException e) {}
        System.out.println(name + " finished working.");
    }
    public static void main(String[] args) {
        for (int i = 1; i <= 5; i++) {
            new Worker("Worker " + i).start();
        }
    }
}
```



2: Write a Java program that creates three threads. The first thread displays “Good Morning” every second, the second thread displays “Hello” every two seconds, and the third thread displays “Welcome” every three seconds.

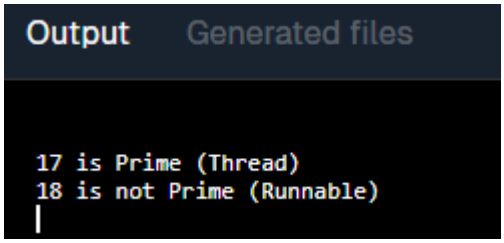
```
class MessagePrinter extends Thread {
    private String message;
    private int interval;
    MessagePrinter(String message, int interval) {
        this.message = message;
        this.interval = interval;
    }
    public void run() {
        while (true) {
            System.out.println(message);
            try {
                Thread.sleep(interval);
            } catch (InterruptedException e) {
                break;
            }
        }
    }
    public static void main(String[] args) {
        new MessagePrinter("Good Morning", 1000).start();
        new MessagePrinter("Hello", 2000).start();
        new MessagePrinter("Welcome", 3000).start();
    }
}
```



3: Implement a class that checks whether a given number is prime using both the Thread class and Runnable interface.

```
// Using Thread
class PrimeCheckThread extends Thread {
    int number;
    PrimeCheckThread(int number) {
        this.number = number;
    }
    public void run() {
        if (isPrime(number)) System.out.println(number + " is Prime (Thread)");
        else System.out.println(number + " is not Prime (Thread)");
    }
    boolean isPrime(int n) {
        if (n <= 1) return false;
        for (int i = 2; i <= n/2; i++)
            if (n % i == 0) return false;
        return true;
    }
}
// Using Runnable
class PrimeCheckRunnable implements Runnable {
    int number;

    PrimeCheckRunnable(int number) {
        this.number = number;
    }
    public void run() {
        if (isPrime(number)) System.out.println(number + " is Prime (Runnable)");
        else System.out.println(number + " is not Prime (Runnable)");
    }
    boolean isPrime(int n) {
        if (n <= 1) return false;
        for (int i = 2; i <= n/2; i++)
            if (n % i == 0) return false;
        return true;
```

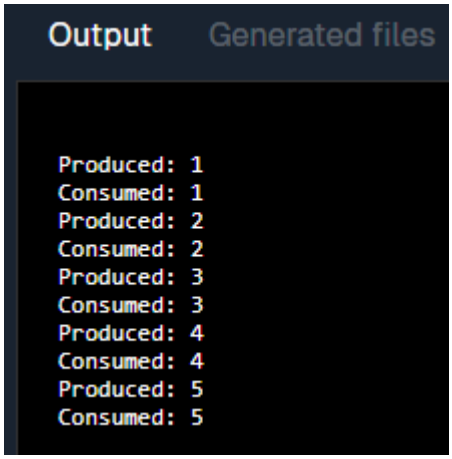
```
    }
    public static void main(String[] args) {
        new PrimeCheckThread(17).start();
        new Thread(new PrimeCheckRunnable(18)).start();
    }
}
```



4: Write a Java program that correctly implements the producer-consumer problem using the concept of inter-thread communication.

```
class SharedData {
    int data;
    boolean available = false;
    synchronized void produce(int value) {
        while (available) {
            try { wait(); } catch (InterruptedException e) {}
        }
        data = value;
        available = true;
        System.out.println("Produced: " + data);
        notify();
    }
    synchronized void consume() {
        while (!available) {
            try { wait(); } catch (InterruptedException e) {}
        }
        System.out.println("Consumed: " + data);
        available = false;
        notify();
    }
}
```

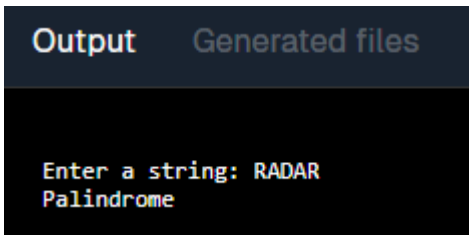
```
class Producer extends Thread {
    SharedData s;
    Producer(SharedData s) {
        this.s = s;
    }
    public void run() {
        for (int i = 1; i <= 5; i++) {
            s.produce(i);
        }
    }
}
public class Consumer extends Thread {
    SharedData s;
    Consumer(SharedData s) {
        this.s = s;
    }
    public void run() {
        for (int i = 1; i <= 5; i++) {
            s.consume();
        }
    }
    public static void main(String[] args) {
        SharedData s = new SharedData();
        new Producer(s).start();
        new Consumer(s).start();
    }
}
```

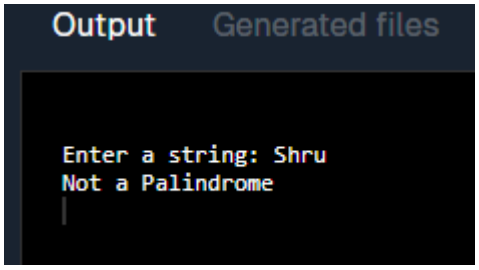


Week 7 (To study Strings in Java)

1: Write a Java program to find whether a given string is palindrome or not.

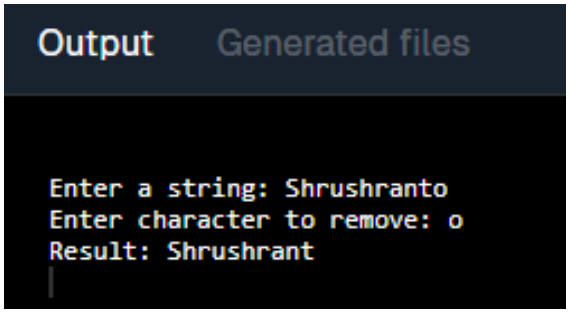
```
import java.util.Scanner;
public class PalindromeCheck {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter a string: ");
        String original = sc.nextLine();
        String reversed = new StringBuilder(original).reverse().toString();
        if (original.equalsIgnoreCase(reversed)) {
            System.out.println("Palindrome");
        } else {
            System.out.println("Not a Palindrome");
        }
    }
}
```





2: Write a method that will remove given character from the String.

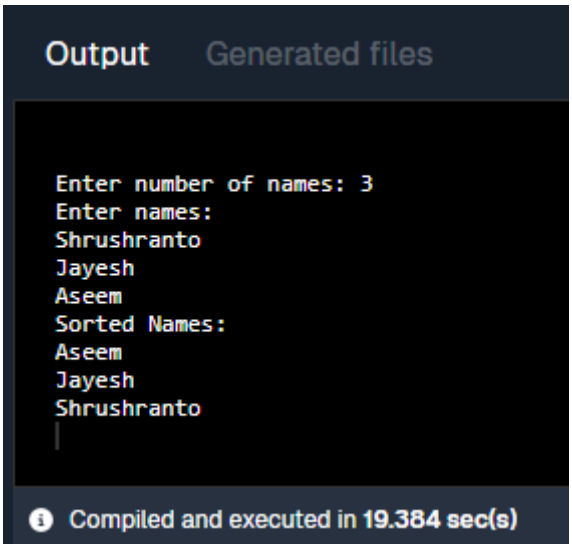
```
import java.util.Scanner;
public class RemoveChar {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter a string: ");
        String str = sc.nextLine();
        System.out.print("Enter character to remove: ");
        char ch = sc.next().charAt(0);
        String result = str.replace(String.valueOf(ch), "");
        System.out.println("Result: " + result);
    }
}
```



3: Write a Java program for sorting a given list of names.

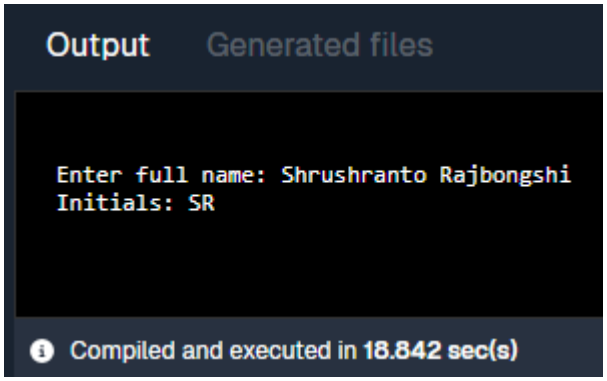
```
import java.util.*;
public class SortNames {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter number of names: ");
        int n = sc.nextInt();
        sc.nextLine(); // consume newline
        String[] names = new String[n];
        System.out.println("Enter names:");
        for (int i = 0; i < n; i++) {
            names[i] = sc.nextLine();
        }
        Arrays.sort(names);
        System.out.println("Sorted Names:");
        for (String name : names) {
```

```
            System.out.println(name);
        }
    }
}
```



4: Write a Java program that computes your initials from your full name and displays them.

```
import java.util.Scanner;
public class Initials {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter full name: ");
        String fullName = sc.nextLine();
        String[] parts = fullName.trim().split(" ");
        System.out.print("Initials: ");
        for (String part : parts) {
            if (!part.isEmpty())
                System.out.print(part.charAt(0));
        }
        System.out.println();
    }
}
```



5: An anagram is a word or a phrase made by transposing the letters of another word or phrase;

for example, "parliament" is an anagram of "partial men," and "software" is an anagram of "swear oft." Write a program that figures out whether one string is an anagram of another string. The program should ignore white space and punctuation.

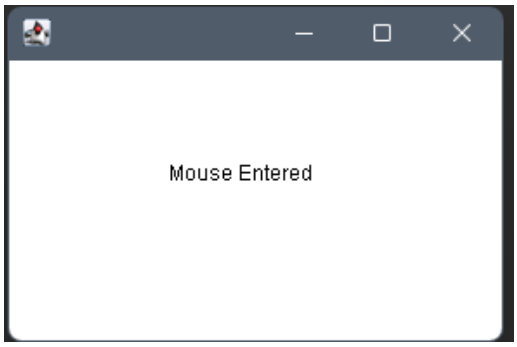
```
import java.util.Arrays;
public class AnagramCheck {
    public static void main(String[] args) {
        String str1 = "parliament";
        String str2 = "partial men";
        // Remove spaces and convert to lowercase
        str1 = str1.replaceAll("[\\W]", "").toLowerCase();
        str2 = str2.replaceAll("[\\W]", "").toLowerCase();
        // Convert to char array and sort
        char[] arr1 = str1.toCharArray();
        char[] arr2 = str2.toCharArray();
        Arrays.sort(arr1);
        Arrays.sort(arr2);
        if (Arrays.equals(arr1, arr2)) {
            System.out.println("Anagram");
        } else {
            System.out.println("Not an Anagram");
        }
    }
}
```

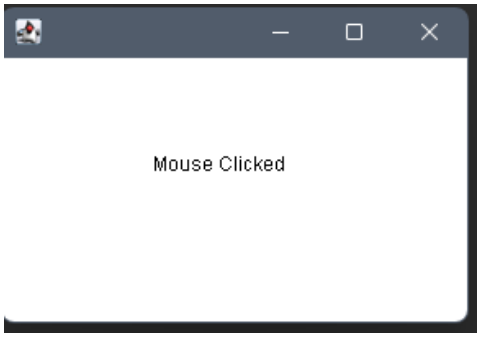
Week 8 (To study Event Handling & AWT in Java)

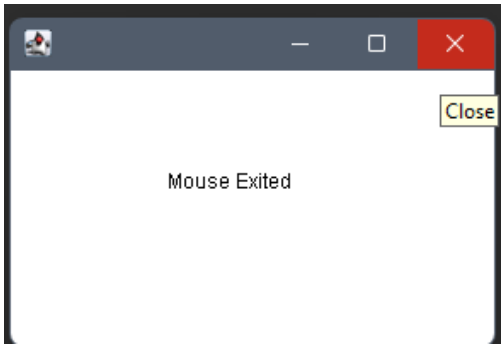
1: Write a Java program that handles all mouse events and shows the event name at the centre of the window when a mouse event is fired (Use Adapter classes).

```
import java.awt.*;
import java.awt.event.*;
public class MouseEvents extends Frame {
    String msg = "";
    public MouseEvents() {
        addMouseListener(new MouseAdapter() {
            public void mouseClicked(MouseEvent e) {
                msg = "Mouse Clicked";
                repaint();
            }
            public void mousePressed(MouseEvent e) {
                msg = "Mouse Pressed";
                repaint();
            }
            public void mouseReleased(MouseEvent e) {
                msg = "Mouse Released";
                repaint();
            }
            public void mouseEntered(MouseEvent e) {
                msg = "Mouse Entered";
                repaint();

            public void mouseExited(MouseEvent e) {
                msg = "Mouse Exited";
                repaint();
            }
        });
        setSize(300, 200);
        setVisible(true);
    }
    public void paint(Graphics g) {
        g.drawString(msg, 100, 100);
```

```
    }
    public static void main(String[] args) {
        new MouseEvents();
    }
}
```



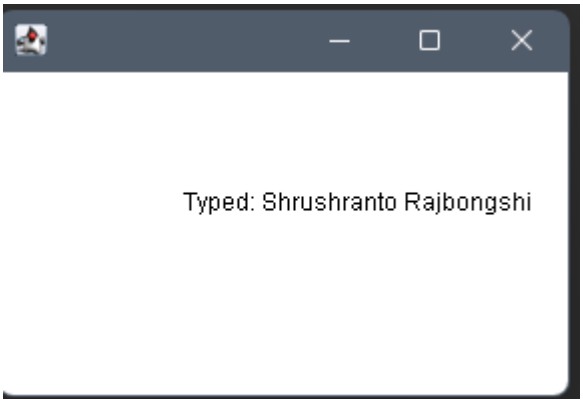




2: Write a Java program for handling Key events.

```
package mypack;
import java.awt.*;
import java.awt.event.*;
public class KeyEvents extends Frame implements KeyListener {
    String msg = "";
    public KeyEvents() {
        addKeyListener(this);
        setSize(300, 200);
        setVisible(true);
    }
    public void keyTyped(KeyEvent e) {
        msg += e.getKeyChar();
        repaint();
    }
```

```
    public void keyPressed(KeyEvent e) {}
    public void keyReleased(KeyEvent e) {}
    public void paint(Graphics g) {
        g.drawString("Typed: " + msg, 100, 100);
    }
    public static void main(String[] args) {
        new KeyEvents();
    }
}
```

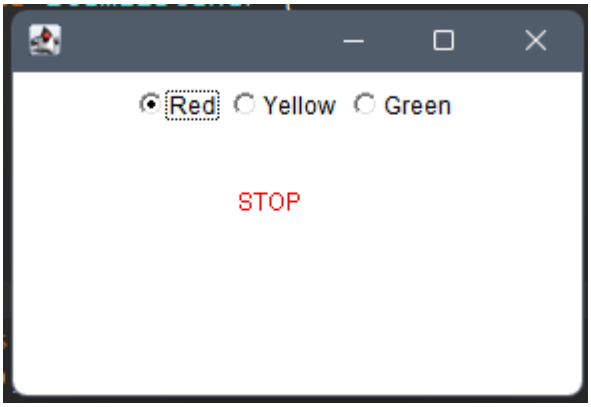


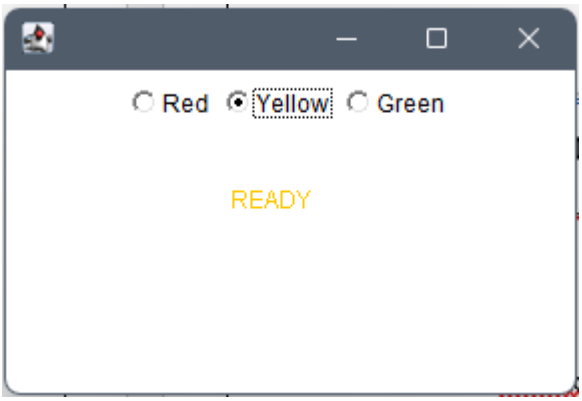
3: Write a java program that simulates a traffic light.

The program lets the user select one of three lights: Red, Yellow or Green with radio buttons. On selecting a button an appropriate message with “STOP” or “READY” or ”GO” should appear above the buttons in selected color. Initially, there is no message shown.

```
package mypack;
import java.awt.*;
import java.awt.event.*;
public class TrafficLight extends Frame implements ItemListener {
    String msg = "";
    Color color = Color.black;
    Checkbox red, yellow, green;
    CheckboxGroup cbg;
    public TrafficLight() {
        cbg = new CheckboxGroup();
        red = new Checkbox("Red", cbg, false);
```

```
        yellow = new Checkbox("Yellow", cbg, false);
        green = new Checkbox("Green", cbg, false);
        setLayout(new FlowLayout());
        add(red); add(yellow); add(green);
        red.addItemListener(this);
        yellow.addItemListener(this);
        green.addItemListener(this);
        setSize(300, 200);
        setVisible(true);
    }
    public void itemStateChanged(ItemEvent e) {
        if (red.getState()) {
            msg = "STOP";
            color = Color.red;
        } else if (yellow.getState()) {
            msg = "READY";
            color = Color.orange;
        } else if (green.getState()) {
            msg = "GO";
            color = Color.green;
        }
        repaint();
    }
    public void paint(Graphics g) {
        g.setColor(color);
        g.drawString(msg, 120, 100);
    }
    public static void main(String[] args) {
        new TrafficLight();
    }
}
```


Red
Yellow
Green
STOP


Red
Yellow
Green
READY


Red
Yellow
Green
GO