**Experiment: 4**

PART A

(PART A: TO BE REFERRED BY STUDENTS)

**Aim : To study concept of access modifiers, packages, introduction to string & string buffer Class.**

**Learning Outcomes: Learner would be able to**

1. Understand the access modifiers and packages
2. Understand the string and string buffer constructs

**Theory:**

In Java, Access modifiers help to restrict the scope of a class, constructor, variable, method, or data member. It provides security, accessibility, etc. to the user depending upon the access modifier used with the element. In this article, let us learn about Java Access Modifiers, their types, and the uses of access modifiers.

Types of Access Modifiers

There are 4 types of access modifiers available in Java:

1. Default – No keyword required
2. Private
3. Protected
4. Public

Packages in Java are a mechanism that encapsulates a group of classes, sub-packages, and interfaces. Packages are used for:

* Prevent naming conflicts by allowing classes with the same name to exist in different packages, like college.staff.cse.Employee and college.staff.ee.Employee.
* They make it easier to organize, locate, and use classes, interfaces, and other components.
* Packages also provide controlled access for Protected members that are accessible within the same package and by subclasses.
* Also for default members (no access specifier) that are accessible only within the same package.

By grouping related classes into packages, Java promotes data encapsulation, making code reusable and easier to manage. Simply import the desired class from a package to use it in your program.

Working of Java Packages

Directory Structure: Package names and directory structures are closely related. For example, if a package name is college.staff.cse, then three directories are, college, staff, and cse, where cse is inside staff, and staff is inside the college.

Naming Conventions: Package names are written in reverse order of domain names, e.g., org.geeksforgeeks.practice. In a college, the convention might be:

* college.tech.cse
* college.tech.ee
* college.art.history

Example:

*import java.util.\*;*

Types of Java Packages

* Built-in Packages
* User-defined Packages

1. Built-in Packages

These packages consist of a large number of classes which are a part of Java API.Some of the commonly used built-in packages are:

* java.lang: Contains language support classes(e.g classes which defines primitive data types, math operations). This package is automatically imported.
* java.io: Contains classes for supporting input / output operations.
* java.util: Contains utility classes which implement data structures like Linked List, Dictionary and support ; for Date / Time operations.
* java.applet: Contains classes for creating Applets.
* java.awt: Contain classes for implementing the components for graphical user interfaces (like button , ;menus etc). 6)
* java.net: Contain classes for supporting networking operations.

2. User-defined Packages

These are the packages that are defined by the user. First we create a directory myPackage (name should be same as the name of the package). Then create the MyClass inside the directory with the first statement being the package names.

A **String** in Java is a sequence of characters that can be used to store and manipulate text data and It is basically an array of characters that are stored in a sequence of memory locations. All the strings in Java are [immutable](https://www.geeksforgeeks.org/java-string-is-immutable-what-exactly-is-the-meaning/)in nature, i.e. once the string is created we can’t change it. This article provides a variety of programs on strings, that are frequently asked in the technical round in various [software engineering interviews](https://www.geeksforgeeks.org/top-50-software-engineering-interview-questions-and-answers/) including various operations such as **reversing, Iteration, palindrome, swapping*, and*splitting of strings etc.**

StringBuffer is a class in Java that represents a mutable sequence of characters. It provides an alternative to the immutable String class, allowing you to modify the contents of a string without creating a new object every time.

**Features of StringBuffer Class**

Here are some important features and methods of the StringBuffer class:

* StringBuffer objects are mutable, meaning that you can change the contents of the buffer without creating a new object.
* The initial capacity of a StringBuffer can be specified when it is created, or it can be set later with the ensureCapacity() method.
* The append() method is used to add characters, strings, or other objects to the end of the buffer.
* The insert() method is used to insert characters, strings, or other objects at a specified position in the buffer.
* The delete() method is used to remove characters from the buffer.
* The reverse() method is used to reverse the order of the characters in the buffer.
* For the following Problem Statements write programs **using classes, objects and methods**

**Tasks:**

1. WAP to perform employee payroll processing using packages. In the java file, Emp.java creates a package employee and creates a class Emp. Declare the variables name, emp\_id, category, basic\_pay, hra, da, net\_pay, PF, gross\_pay, income\_tax, and allowance. Calculate the values in methods. Create another java file Emppay.java. Create an object e to call the methods to perform and print values.
2. Develop an **Employee Attendance Management System** using **Java packages**. Create a package attendance and define a class EmployeeAttendance in EmployeeAttendance.java. Declare variables such as name, empID, totalWorkingDays, daysPresent, daysAbsent, and attendancePercentage. Implement methods to **calculate attendance percentage and display attendance records**. In another file AttendanceReport.java, create an object, input attendance details, compute the percentage, and print the attendance summary.
3. Create a **Student Management System** with a package university and the following requirements:

* **Class Student** (inside university package)
  + Private attributes: name, roll\_no, course, marks[]
  + Methods to **calculate total marks, percentage, and grade**
* **Class Result** (in the same package)
  + Inherits Student
  + Can access marks and calculate grade
* **Class StudentPortal** (in another package portal)
  + Tries to access student details (but can't access private data)

1. Develop a **Vehicle Rental System** with a package rental and the following requirements:

* **Class Vehicle** (inside rental package)
  + Private attributes: vehicleNumber, model, rentalRate
  + Protected method calculateRental(int days)
* **Class Car & Bike** (same package, inherit Vehicle)
  + Override rental calculation based on type
* **Class Customer** (in another package customer)
  + Can only rent vehicles (not modify rental rates directly)

1. Develop a **password masking system** that replaces characters with \* using StringBuffer.replace().

* Take password input.
* Replace characters with \*.

**Example:**

* **Input:** "mypassword"
* **Output:** "\*\*\*\*\*\*\*\*\*\*"

1. Create a **program to check whether a given string is a palindrome** (same when reversed). Use StringBuffer.reverse() for checking.

* Convert input into lowercase.
* Remove spaces and special characters (optional).
* Reverse the string using reverse().
* Compare with the original string.

|  |  |
| --- | --- |
| **Roll No.: C126** | **Name: Rushabh Abhay Shah** |
| **Prog/Yr/Sem: BTI 4th Year 8th Sem** | **Batch: 2021-2027** |
| **Date of Experiment:01-03-2025** | **Date of Submission: 01-03-2025** |

**Code**

**TASK 1**

Emp.java

package employee;

public class Emp {

    private String name;

    private int emp\_id;

    private String category;

    private double basic\_pay;

    private double hra;

    private double da;

    private double net\_pay;

    private double PF;

    private double gross\_pay;

    private double income\_tax;

    private double allowance;

    public Emp(String name, int emp\_id, String category, double basic\_pay) {

        this.name = name;

        this.emp\_id = emp\_id;

        this.category = category;

        this.basic\_pay = basic\_pay;

    }

    public void calculateHRA() {

        this.hra = 0.2 \* basic\_pay;

    }

    public void calculateDA() {

        this.da = 0.1 \* basic\_pay;

    }

    public void calculatePF() {

        this.PF = 0.12 \* basic\_pay;

    }

    public void calculateIncomeTax() {

        this.income\_tax = 0.05 \* (basic\_pay + hra + da);

    }

    public void calculateAllowance() {

        this.allowance = 0.05 \* basic\_pay;

    }

    public void calculateGrossPay() {

        this.gross\_pay = basic\_pay + hra + da + allowance;

    }

    public void calculateNetPay() {

        this.net\_pay = gross\_pay - PF - income\_tax;

    }

    public void printDetails() {

        System.out.println("Name: " + name);

        System.out.println("Employee ID: " + emp\_id);

        System.out.println("Category: " + category);

        System.out.println("Basic Pay: " + basic\_pay);

        System.out.println("HRA: " + hra);

        System.out.println("DA: " + da);

        System.out.println("PF: " + PF);

        System.out.println("Gross Pay: " + gross\_pay);

        System.out.println("Income Tax: " + income\_tax);

        System.out.println("Allowance: " + allowance);

        System.out.println("Net Pay: " + net\_pay);

    }

}

Emppay.java

import employee.\*;

public class Emppay {

    public static void main(String[] args) {

        Emp e = new Emp("John Doe", 12345, "Manager", 50000.0);

        e.calculateHRA();

        e.calculateDA();

        e.calculatePF();

        e.calculateIncomeTax();

        e.calculateAllowance();

        e.calculateGrossPay();

        e.calculateNetPay();

        e.printDetails();

    }

}

**TASK 2**

EmployeeAttendance.java

// EmployeeAttendance.java

package attendance;

import java.util.Scanner;

public class EmployeeAttendance {

    private String name;

    private int empID;

    private int totalWorkingDays;

    private int daysPresent;

    private int daysAbsent;

    private double attendancePercentage;

    // Constructor

    public EmployeeAttendance(String name, int empID, int totalWorkingDays, int daysPresent) {

        this.name = name;

        this.empID = empID;

        this.totalWorkingDays = totalWorkingDays;

        this.daysPresent = daysPresent;

        this.daysAbsent = totalWorkingDays - daysPresent;

        calculateAttendancePercentage();

    }

    // Method to calculate attendance percentage

    private void calculateAttendancePercentage() {

        if (totalWorkingDays > 0) {

            attendancePercentage = ((double) daysPresent / totalWorkingDays) \* 100;

        } else {

            attendancePercentage = 0;

        }

    }

    // Method to display attendance records

    public void displayAttendance() {

        System.out.println("Employee ID: " + empID);

        System.out.println("Name: " + name);

        System.out.println("Total Working Days: " + totalWorkingDays);

        System.out.println("Days Present: " + daysPresent);

        System.out.println("Days Absent: " + daysAbsent);

        System.out.printf("Attendance Percentage: %.2f%%\n", attendancePercentage);

    }

}

AttendanceReport.java

import attendance.\*;

import java.util.Scanner;

public class AttendanceReport {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        // Taking input for employee attendance

        System.out.print("Enter Employee Name: ");

        String name = scanner.nextLine();

        System.out.print("Enter Employee ID: ");

        int empID = scanner.nextInt();

        System.out.print("Enter Total Working Days: ");

        int totalWorkingDays = scanner.nextInt();

        System.out.print("Enter Days Present: ");

        int daysPresent = scanner.nextInt();

        // Creating EmployeeAttendance object

        EmployeeAttendance employee = new EmployeeAttendance(name, empID, totalWorkingDays, daysPresent);

        // Displaying attendance details

        System.out.println("\nAttendance Summary:");

        employee.displayAttendance();

        scanner.close();

    }

}

**TASK 3**

StudentPortal.java

package portal;

import university.Student;

public class StudentPortal {

    public static void main(String[] args) {

        // Accessing private attributes directly is not allowed

        int[] marks = {85, 78, 92, 80, 90};

        Student student = new Student("Rushabh Shah", 126, "Computer Science", marks);

        // Only public methods can be accessed

        System.out.println("Student Name: " + student.getName());

        System.out.println("Roll No: " + student.getRollNo());

        System.out.println("Course: " + student.getCourse());

    }

}

Result.java

package university;

public class Result extends Student {

    public Result(String name, int roll\_no, String course, int[] marks) {

        super(name, roll\_no, course, marks);

    }

    // Calculate grade

    public char getGrade() {

        double percentage = getPercentage();

        if (percentage >= 90) return 'A';

        else if (percentage >= 75) return 'B';

        else if (percentage >= 60) return 'C';

        else return 'D';

    }

    // Display result

    public void displayResult() {

        System.out.println("Student: " + getName());

        System.out.println("Roll No: " + getRollNo());

        System.out.println("Course: " + getCourse());

        System.out.println("Total Marks: " + getTotalMarks());

        System.out.println("Percentage: " + getPercentage() + "%");

        System.out.println("Grade: " + getGrade());

    }

}

Student.java

package university;

public class Student {

    private String name;

    private int roll\_no;

    private String course;

    private int[] marks;

    // Constructor

    public Student(String name, int roll\_no, String course, int[] marks) {

        this.name = name;

        this.roll\_no = roll\_no;

        this.course = course;

        this.marks = marks;

    }

    // Calculate total marks

    public int getTotalMarks() {

        int total = 0;

        for (int mark : marks) {

            total += mark;

        }

        return total;

    }

    // Calculate percentage

    public double getPercentage() {

        return (double) getTotalMarks() / marks.length;

    }

    // Getters for derived class

    protected int[] getMarks() {

        return marks;

    }

    // Public method to get name

    public String getName() {

        return name;

    }

    public int getRollNo() {

        return roll\_no;

    }

    public String getCourse() {

        return course;

    }

}

**TASK 4**

Customer.java

package customer;

import rental.Car;

import rental.Bike;

public class Customer {

    public static void main(String[] args) {

        Car car = new Car("KA-01-1234", "Honda City", 1000);

        Bike bike = new Bike("KA-02-5678", "Yamaha FZ", 500);

        System.out.println("Car Rental for 3 days: $" + car.calculateRental(3));

        System.out.println("Bike Rental for 5 days: $" + bike.calculateRental(5));

    }

}

Bike.java

package rental;

public class Bike extends Vehicle {

    public Bike(String vehicleNumber, String model, double rentalRate) {

        super(vehicleNumber, model, rentalRate);

    }

    @Override

    public double calculateRental(int days) {

        return super.calculateRental(days); // No extra charge for bikes

    }

}

Car.java

package rental;

public class Car extends Vehicle {

    public Car(String vehicleNumber, String model, double rentalRate) {

        super(vehicleNumber, model, rentalRate);

    }

    @Override

    public double calculateRental(int days) {

        return super.calculateRental(days) \* 1.1; // 10% extra for cars

    }

}

Vehicle.java

package rental;

public class Vehicle {

    private String vehicleNumber;

    private String model;

    private double rentalRate;

    public Vehicle(String vehicleNumber, String model, double rentalRate) {

        this.vehicleNumber = vehicleNumber;

        this.model = model;

        this.rentalRate = rentalRate;

    }

    public double calculateRental(int days) {

        return rentalRate \* days;

    }

}

**TASK 5**

PasswordMasker.java

package security;

public class PasswordMasker {

    // Method to mask password

    public static String maskPassword(String password) {

        StringBuffer maskedPassword = new StringBuffer(password);

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

            maskedPassword.setCharAt(i, '\*');

        }

        return maskedPassword.toString();

    }

}

Main.java

import security.PasswordMasker;

import java.util.Scanner;

public class Main {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter Password: ");

        String password = scanner.nextLine();

        // Mask password using PasswordMasker class

        String maskedPassword = PasswordMasker.maskPassword(password);

        System.out.println("Masked Password: " + maskedPassword);

        scanner.close();

    }

}

**TASK 6**

PalindromeChecker.java

package utility;

public class PalindromeChecker {

    // Method to check if a string is a palindrome

    public static boolean isPalindrome(String input) {

        // Convert to lowercase and remove non-alphanumeric characters

        String cleaned = input.toLowerCase().replaceAll("[^a-zA-Z0-9]", "");

        StringBuffer reversed = new StringBuffer(cleaned).reverse();

        return cleaned.equals(reversed.toString());

    }

}

Main.java

import utility.PalindromeChecker;

import java.util.Scanner;

public class Main {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

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

        String input = scanner.nextLine();

        // Check if input is a palindrome

        if (PalindromeChecker.isPalindrome(input)) {

            System.out.println("The string is a palindrome.");

        } else {

            System.out.println("The string is not a palindrome.");

        }

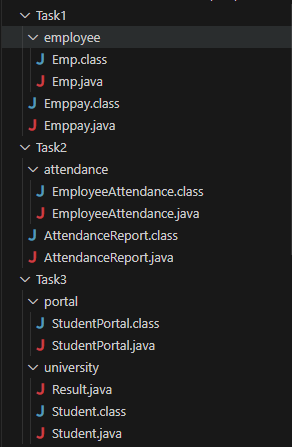
        scanner.close();

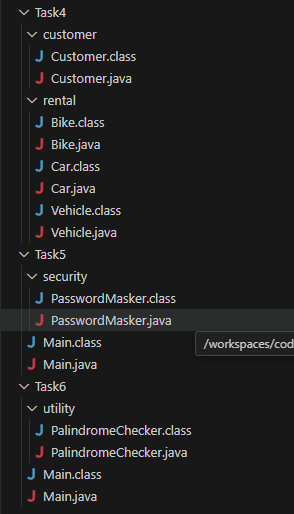
    }

}

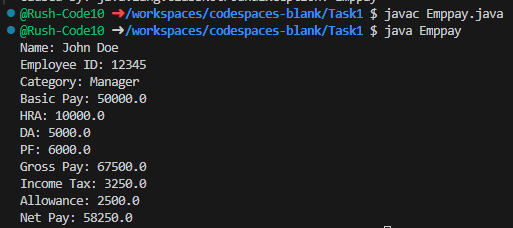
**Output**

**Directories**

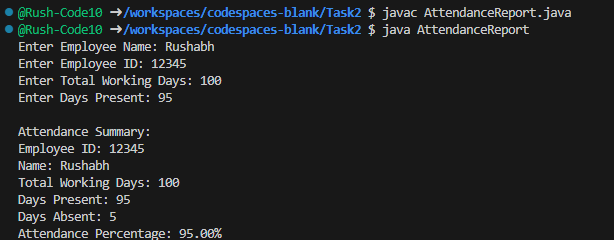




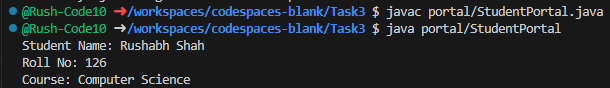
TASK 1



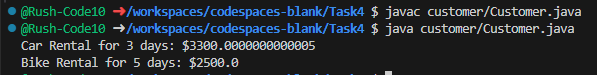
TASK 2



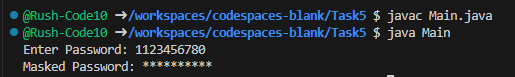
TASK 3



TASK 4



TASK 5



TASK 6

