

LAB RECORD

23CSE111- Object Oriented Programming

***Submitted by***

CH.SC.U4CSE24022 – KANIPAKAM POOJITHA

# BACHELOR OF TECHNOLOGY

## IN

COMPUTER SCIENCE AND ENGINEERING

AMRITA VISHWA VIDYAPEETHAM AMRITA SCHOOL OF COMPUTING

CHENNAI



**AMRITA VISHWA VIDYAPEETHAM AMRITA SCHOOL OF COMPUTING, CHENNAI**

**BONAFIDE CERTIFICATE**

This is to certify that the Lab Record work for 23CSE111- Object Oriented Programming Subject submitted by ***CH.SC.U4CSE24022 – KANIPAKAM POOJITHA*** in **“Computer Science and Engineering”** is a Bonafide record of the work carried out under my guidance and supervision at Amrita School of Computing, Chennai.

This Lab examination held on / /2025

Internal Examiner 1 Internal Examiner 2

## INDEX

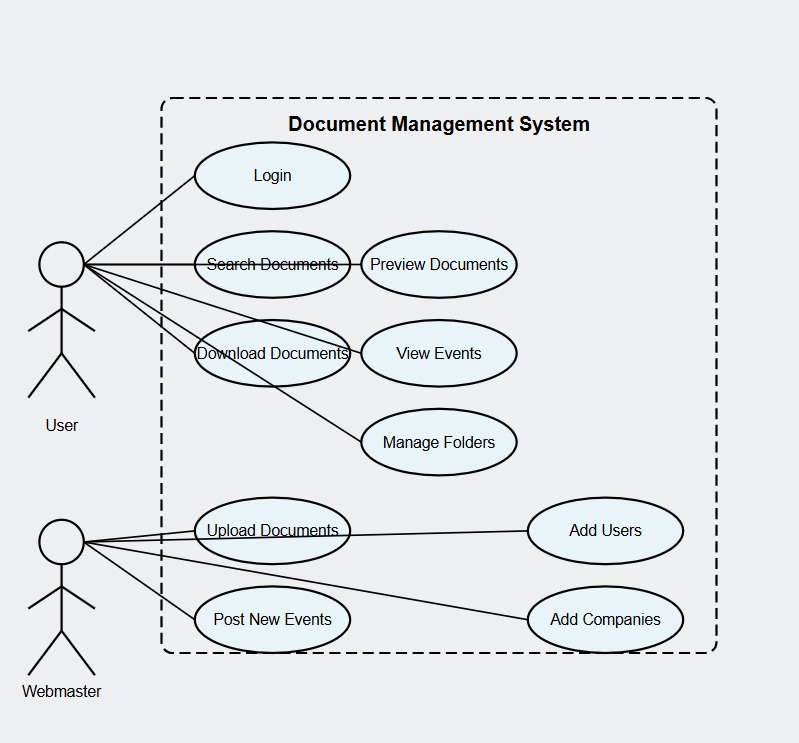
|  |  |  |
| --- | --- | --- |
| **S.NO** | **TITLE** | **PAGE.NO** |
| **UML DIAGRAM** | |  |
| 1. | **DOCUMENT MANAGEMENT SYSTEM** | **6-10** |
|  | 1.a)Use Case Diagram |  |
|  | 1.b)Class Diagram |  |
|  | 1.c) Sequence Diagram |  |
|  | 1.d) Collaboration Diagram |  |
|  | 1.e) Activity Diagram |  |
| 2. | **COURSE REGISTRATION SYSTEM** | **11-15** |
|  | 2.a) Use Case Diagram |  |
|  | 2.b) Class Diagram |  |
|  | 2.c) Sequence Diagram |  |
|  | 2.d) Collaboration Diagram |  |
|  | 2.e) Activity Diagram |  |
| 3. | **BASIC JAVA PROGRAMS** | **16-22** |
|  | 3.a) Guessing Number Game |  |
|  | 3.b) Grade Calculator |  |
|  | 3.c) BM1 |  |
|  | 3.d) Area |  |
|  | 3.e) Reversing Number |  |
|  | 3.f) Fibonacci series |  |
|  | 3.g) Prime Number |  |
|  | 3.h) Palindrome |  |
|  | 3.i) Sequence of stars |  |
|  | 3.j) Factorial of a number |  |
|  | **INHERITANCE** |  |

|  |  |  |
| --- | --- | --- |
| 4. | **SINGLE INHERITANCE PROGRAMS** | 22-24 |
|  | 4.a) Laptop Specifications |  |
|  | 4.b) Specialty of Doctors |  |
| 5. | **MULTILEVEL INHERITANCE PROGRAMS** | 25-28 |
|  | 5.a) User login |  |
|  | 5.b) Playing of Audio file |  |
| 6. | **HIERARCHICAL INHERITANCE PROGRAMS** | 28-32 |
|  | 6.a) Student and Teacher extends user |  |
|  | 6.b) Gadgets |  |
| 7. | **HYBRID INHERITANCE PROGRAMS** | 32-37 |
|  | 7.a) Video and Audio player |  |
|  | 7.b) Diagnosable and Treatable |  |
|  | **POLYMORPHISM** |  |
| 8. | **CONSTRUCTOR OVERRIDING PROGRAMS** | 37-39 |
|  | 8.a) Notification System |  |
|  | 8.b) Translator |  |
| 9. | **CONSTRUCTOR OVERLOADING PROGRAMS** | 40-42 |
|  | 9.a) Smart light |  |
|  | 9.b) Loan EMI calculator |  |
| 10. | **METHOD OVERLOADING PROGRAMS** | 42-44 |
|  | 10.a) Quiz Scorer |  |
|  | 10.b) User Login Logger |  |
| 11. | **METHOD OVERRIDING PROGRAMS** | 44-46 |
|  | 11.a) Game Character |  |
|  | 11.b) Ticket Booking |  |
|  | **ABSTRACTION** |  |
| 12. | **INTERFACE PROGRAMS** | 46-52 |
|  | 12.a) Delivery Partner |  |
|  | 12.b) Streaming of online platforms |  |
|  | 12.c) Controllable Device |  |
|  | 12.d) Ridable Vehicles |  |
| 13. | **ABSTRACT CLASS PROGRAMS** | 53-59 |
|  | 13.a) Online Courses |  |
|  | 13.b) Ticket Booking |  |
|  | 13.c) Musical Instruments |  |
|  | 13.d) Health Monitor |  |
|  | **ENCAPSULATION** |  |
| 14. | **ENCAPSULATION PROGRAMS** | 59-63 |
|  | 14.a) Locker Status |  |
|  | 14.b) Package Delivery Tracker |  |
|  | 14.c) ATM Machine |  |
|  | 14.d) Employee salary |  |
| 15. | **PACKAGES PROGRAMS** | 63-73 |
|  | 15.a)User Defined Packages |  |
|  | 15.b)User Defined Packages |  |
|  | 15.c)Built – in Package(3 Packages) |  |
|  | 15.d)Built – in Package(3 Packages) |  |
| 16. | **EXCEPTION HANDLING PROGRAMS** | 74-76 |
|  | 16.a) Under age Exceptions |  |
|  | 16.b) Invalid marks Exception |  |
|  | 16.c) Username too short exception |  |
|  | 16.d) File too large exception |  |
| 17. | **FILE HANDLING PROGRAMS** | 76-81 |
|  | 17.a) Length reader |  |
|  | 17.b) Word Search |  |
|  | 17.c) Encryption of text |  |
|  | 17.d) Stack Trace |  |

**1.Document Management System:**

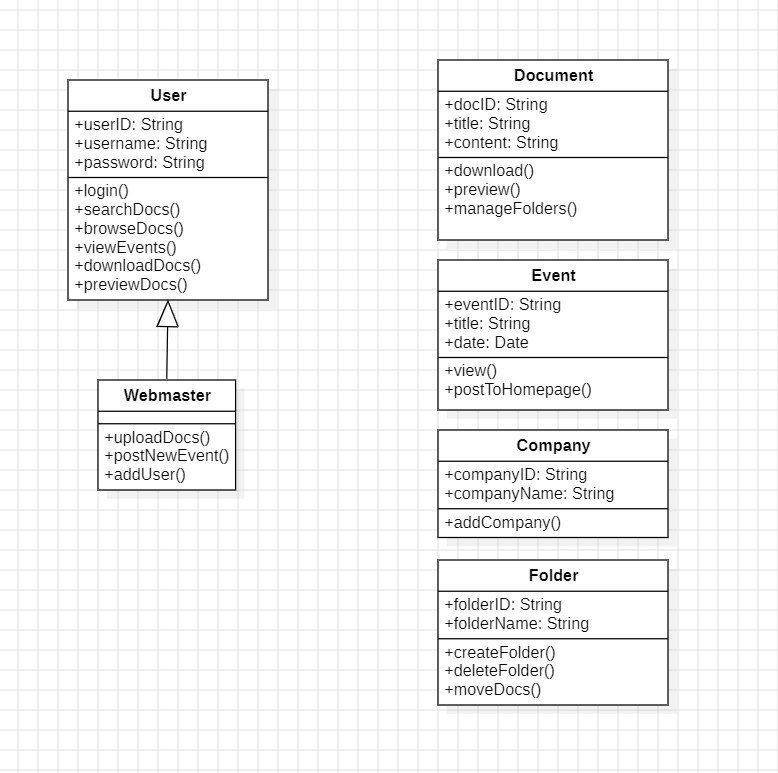
**1a:**

**USE CASE DIAGRAM:**



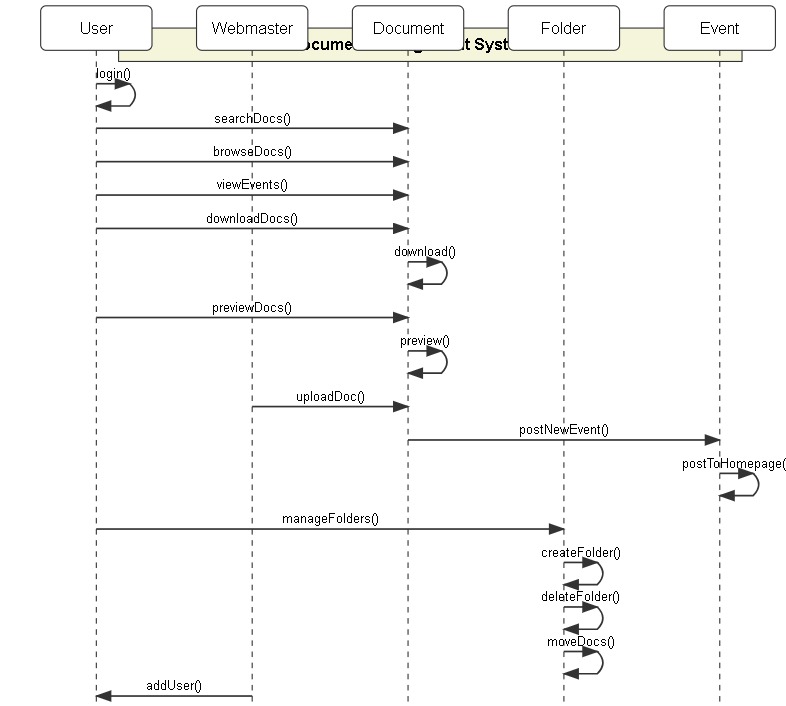
**1b:**

**Class Diagram:**

****

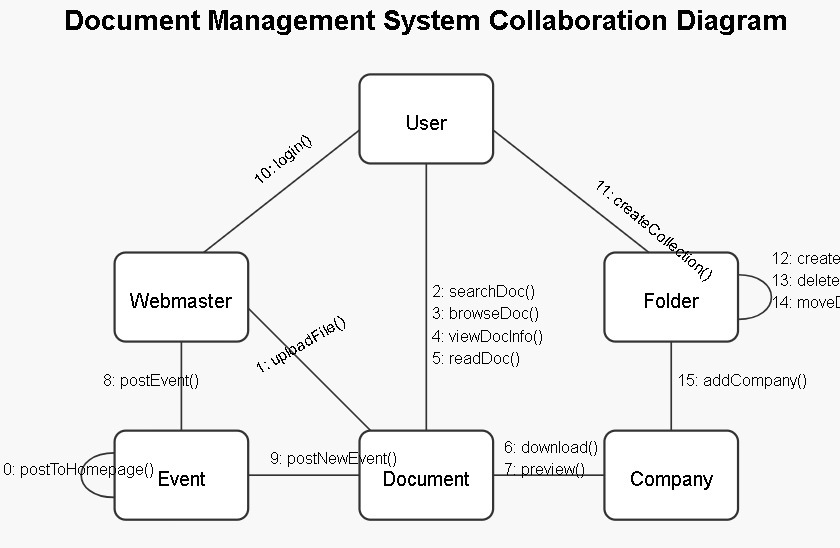
**1c:**

**Sequence Diagram:**



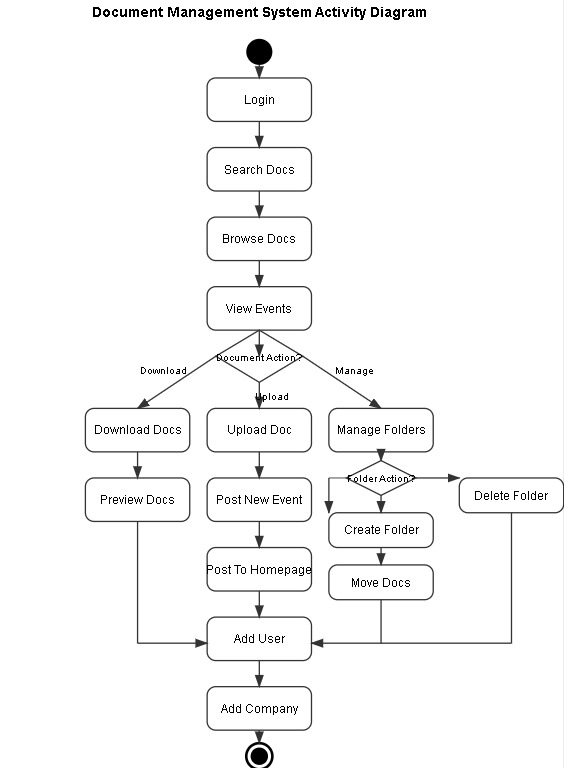
**1d:**

**Collaboration Diagram:**



**1e:**

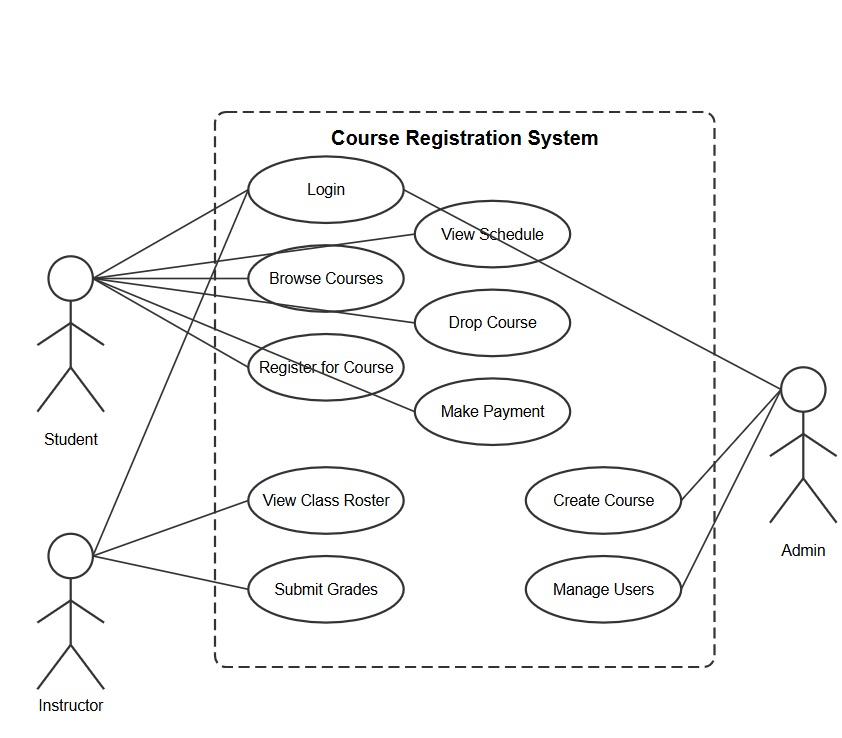
**Activity Diagram:**



**2.Course Registration System:**

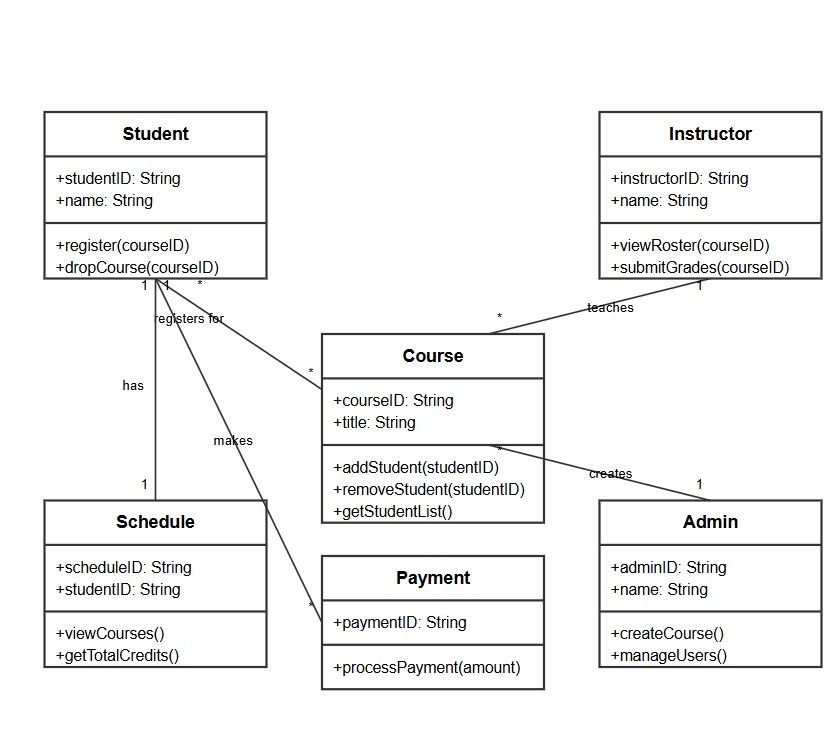
**2a:**

**USE CASE DIAGRAM:**



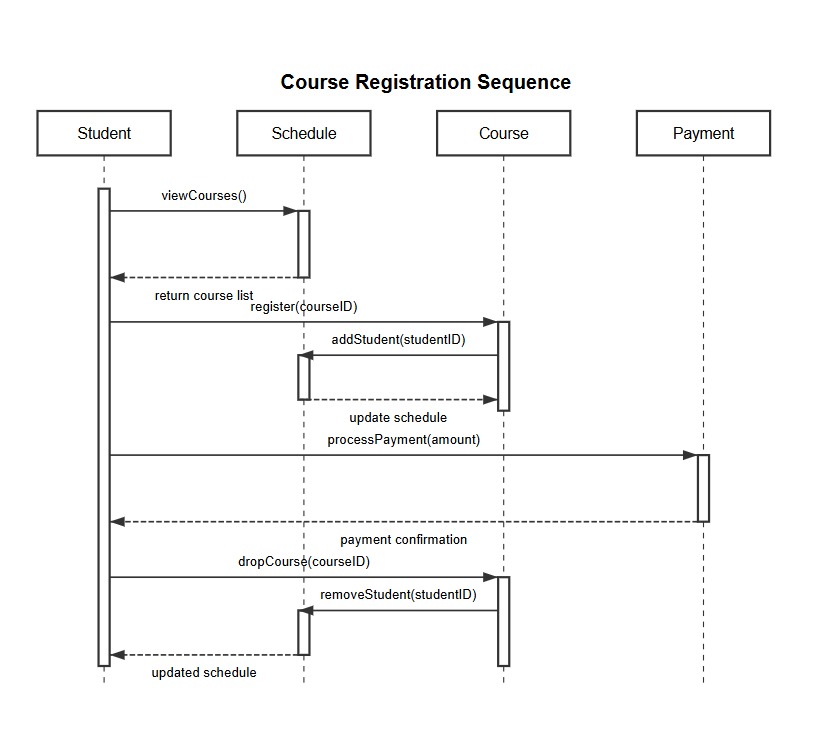
**2b:**

**Class Diagram:**



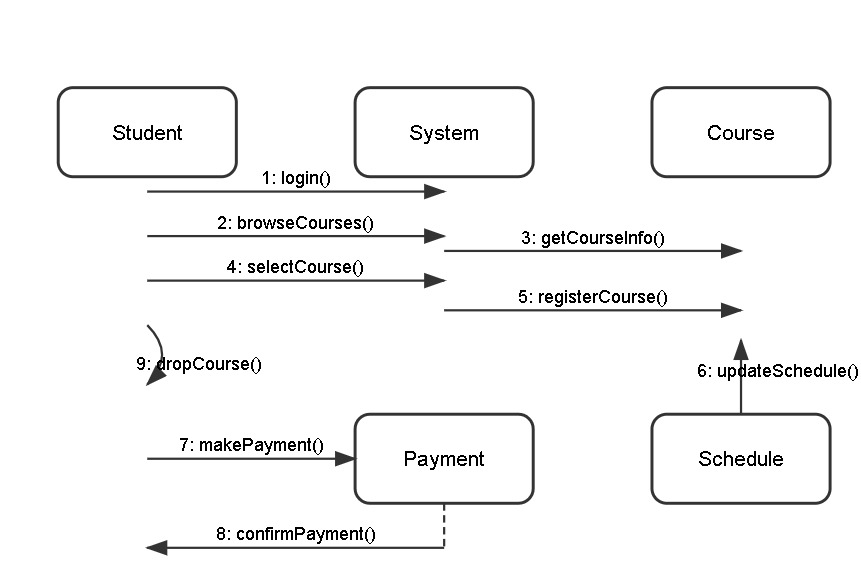
**2c:**

**Sequence Diagram:**



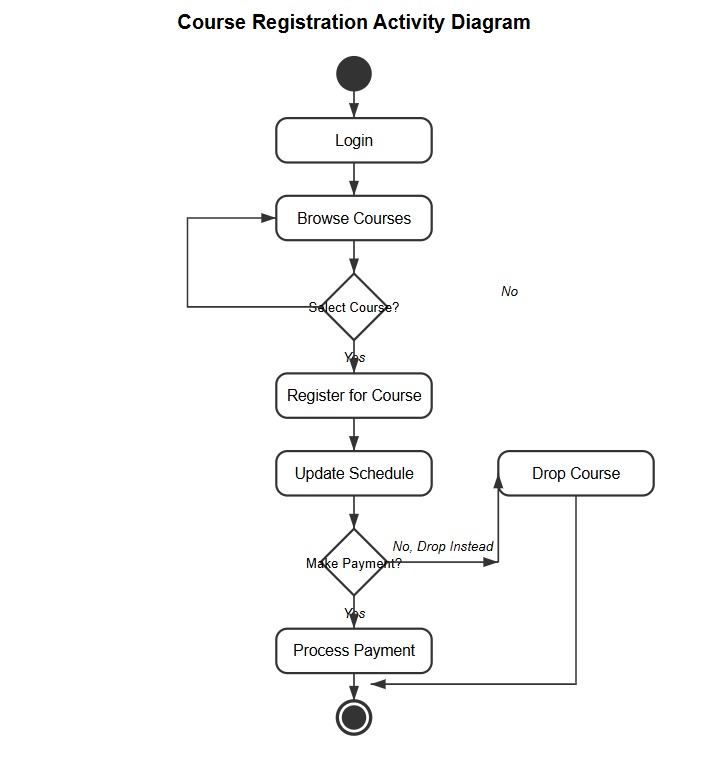
**2d:**

**Collaboration Diagram:**



**2e:**

**Activity Diagram:**



**3.a) Guessing Number Game**

**Code:**

import java.util.Scanner;

public class Task1 {

public static void

guessingNumberGame()

{

Scanner sc = new Scanner(System.in);

int number = 1 + (int)(100\* Math.random());

int K = 5;

int i, guess;

System.out.println("A number is chosen"+ " between 1 to 100."+ "Guess the number"+ " within 5 trials.");

for (i = 0; i < K; i++)

{

System.out.println("Guess the number:");

guess = sc.nextInt();

if (number == guess) {

System.out.println("Congratulations!"+ " You guessed the number.");

break;

}

else if (number > guess&& i != K - 1) {

System.out.println("The number is "+ "greater than " + guess);

}

else if (number < guess&& i != K - 1) {

System.out.println("The number is"+ " less than " + guess);

}

}

if (i == K) {

System.out.println("You have exhausted"+ " K trials.");

System.out.println("The number was " + number);

}

}

public static void

main(String arg[])

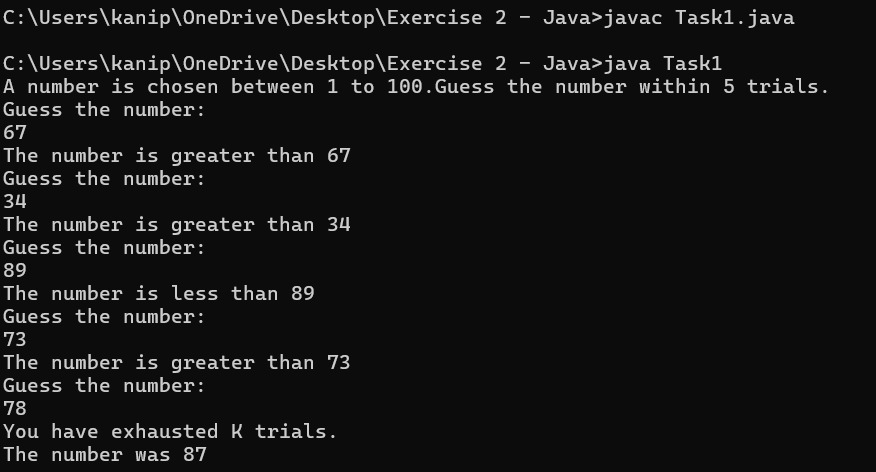
{

guessingNumberGame();

}

}

**Output:**



**3.b) Grade Calculator**

**CODE:**

import java.util.Scanner;

public class Task2 {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

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

int m = scanner.nextInt();

if (m <= 100) {

if (m >= 85) {

System.out.println("Excellent");

}

else if (m >= 70 && m <= 84) {

System.out.println("Good");

}

else if (m >= 60 && m < 70) {

System.out.println("Satisfactory");

}

else {

System.out.println("Not satisfactory");

}

}

else {

System.out.println("Marks should be less than 100!!");

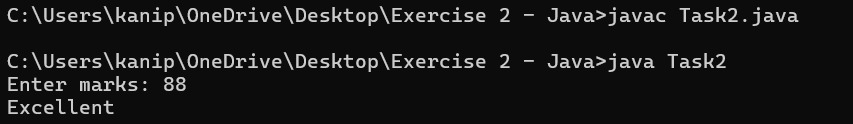
}

scanner.close();

}

}

**OUTPUT:**



**3.c) BM1**

**CODE:**

import java.util.Scanner;

public class Task3 {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the Weight in kilograms: ");

double weight = scanner.nextDouble();

System.out.print("Enter the Height in meters: ");

double height = scanner.nextDouble();

double bmi = weight / (height \* height);

if (bmi < 18.5) {

System.out.println("Underweight");

}

else if (bmi <= 24.9) {

System.out.println("Normal weight");

}

else if (bmi <= 29.9) {

System.out.println("Overweight");

}

else {

System.out.println("Obese");

}

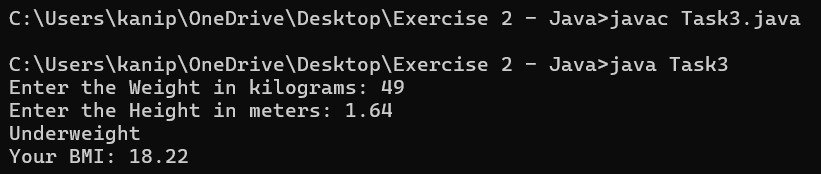
System.out.printf("Your BMI: %.2f\n", bmi);

scanner.close();

}

}

**OUTPUT:**



**3.d) Area**

**CODE:**

import java.util.Scanner;

public class Task4 {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the value of base: ");

double base = scanner.nextDouble();

System.out.print("Enter the value of height: ");

double height = scanner.nextDouble();

double area = 0.5 \* base \* height;

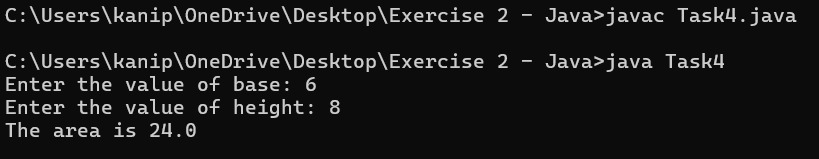
System.out.println("The area is " + area);

scanner.close();

}

}

**OUTPUT:**



**3.e) Reversing Number**

**CODE:**

import java.util.Scanner;

public class Task5 {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

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

int num = scanner.nextInt();

int reversed = 0;

while (num != 0) {

int digit = num % 10;

reversed = reversed \* 10 + digit;

num /= 10;

}

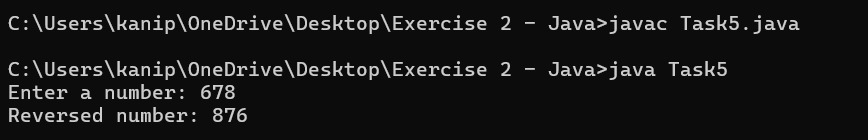
System.out.println("Reversed number: " + reversed);

scanner.close();

}

}

**OUTPUT:**



**3.f) Fibonacci series**

**CODE:**

import java.util.Scanner;

public class Task6 {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the number of terms: ");

int n = scanner.nextInt();

int a = 0, b = 1;

System.out.print("Fibonacci Series: " + a + " " + b + " ");

for (int i = 2; i < n; i++) {

int next = a + b;

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

a = b;

b = next;

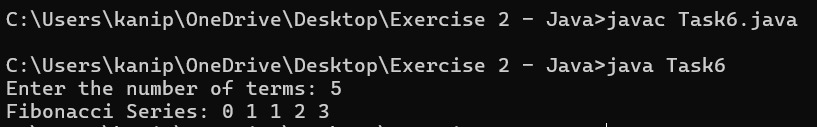
}

scanner.close();

}

}

**OUTPUT:**



**3.g) Prime Number**

**CODE:**

import java.util.Scanner;

public class Task7{

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

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

int num = scanner.nextInt();

boolean isPrime = true;

if (num <= 1) {

isPrime = false;

}

else {

for (int i = 2; i <= num / 2; i++) {

if (num % i == 0) {

isPrime = false;

break;

}

}

}

if (isPrime)

System.out.println(num + " is a Prime Number.");

else

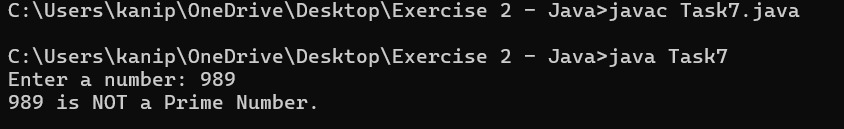
System.out.println(num + " is NOT a Prime Number.");

scanner.close();

}

}

**OUTPUT:**



**3.h) Palindrome**

**CODE:**

import java.util.Scanner;

public class Task8{

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

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

int num = scanner.nextInt();

int originalNum = num;

int reversed = 0;

while (num != 0) {

int digit = num % 10;

reversed = reversed \* 10 + digit;

num /= 10;

}

if (originalNum == reversed) {

System.out.println(originalNum + " is a Palindrome.");

}

else {

System.out.println(originalNum + " is NOT a Palindrome.");

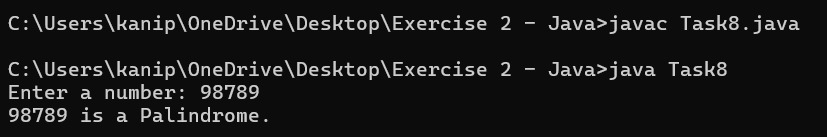
}

scanner.close();

}

}

**OUTPUT:**



**3.i) Sequence of stars**

**CODE:**

import java.util.Scanner;

public class Task9{

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter number of rows: ");

int rows = scanner.nextInt();

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

{

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

{

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

}

System.out.println();

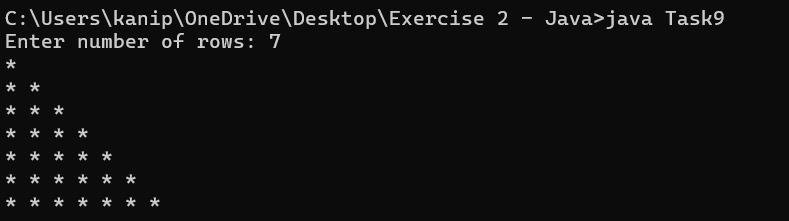
}

scanner.close();

}

}

**OUTPUT:**

****

**3.j) Factorial of a number**

**CODE:**

import java.util.Scanner;

public class Task10{

public static int factorial(int n) {

if (n == 0 || n == 1) {

return 1;

}

else(n>1) {

return n \* factorial(n - 1);

}

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

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

int num = scanner.nextInt();

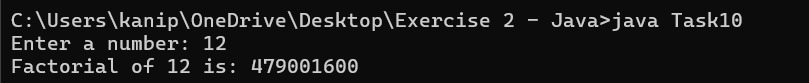
System.out.println("Factorial of " + num + " is: " + factorial(num));

scanner.close();

}

}

**OUTPUT:**



#### 4A: SINGLE INHERITANCE 1

#### CODE:

#### class Laptop {

#### String brand;

#### int ram;

#### int storage;

#### Laptop(String brand, int ram, int storage) {

#### this.brand = brand;

#### this.ram = ram;

#### this.storage = storage;

#### }

#### void specs() {

#### System.out.println("Brand: " + brand);

#### System.out.println("RAM: " + ram + "GB");

#### System.out.println("Storage: " + storage + "GB");

#### }

#### }

#### class GamingLaptop extends Laptop {

#### String graphicsCard;

#### GamingLaptop(String brand, int ram, int storage, String graphicsCard) {

#### super(brand, ram, storage);

#### this.graphicsCard = graphicsCard;

#### }

#### void gamingSpecs() {

#### specs(); // Call base class method

#### System.out.println("Graphics Card: " " + graphicsCard);

#### }

#### }

#### public class SI1 {

#### public static void main(String[] args) {

#### GamingLaptop gl = new GamingLaptop("ASUS", 16, 512, "NVIDIA RTX 3060");

#### gl.gamingSpecs();

#### }

#### }

#### OUTPUT:

#### 

#### 4B:SINGLE INHERITANCE 2

#### CODE:

#### class Doctor {

#### String name;

#### String specialty;

#### Doctor(String name, String specialty) {

#### this.name = name;

#### this.specialty = specialty;

#### }

#### void treatPatient() {

#### System.out.println(name + " is treating a patient in " + specialty);

#### }

#### }

#### class Surgeon extends Doctor {

#### String surgeryType;

#### Surgeon(String name, String specialty, String surgeryType) {

#### super(name, specialty);

#### this.surgeryType = surgeryType;

#### }

#### void performSurgery() {

#### System.out.println(name + " is performing a " + surgeryType + " surgery.");

#### }

#### }

#### public class SI2 {

#### public static void main(String[] args) {

#### Surgeon surgeon = new Surgeon("Dr. Smith", "Cardiology", "Heart");

#### surgeon.treatPatient();

#### surgeon.performSurgery();

#### }

#### }

#### OUTPUT:

#### 

#### 5A: MULTILEVEL INHERITANCE 1

#### CODE:

#### class User {

#### String username;

#### User(String username) {

#### this.username = username;

#### }

#### void login() {

#### System.out.println(username + " logged in.");

#### }

#### void logout() {

#### System.out.println(username + " logged out.");

#### }

#### }

#### class Admin extends User {

#### Admin(String username) {

#### super(username);

#### }

#### void resetPassword() {

#### System.out.println(username + " reset a user's password.");

#### }

#### void blockUser() {

#### System.out.println(username + " blocked a user.");

#### }

#### }

#### class SuperAdmin extends Admin {

#### 

#### SuperAdmin(String username) {

#### super(username);

#### }

#### void grantPermissions() {

#### System.out.println(username + " granted permissions.");

#### }

#### void accessLogs() {

#### System.out.println(username + " accessed system logs.");

#### }

#### }

#### public class MI1 {

#### public static void main(String[] args) {

#### SuperAdmin superAdmin = new SuperAdmin("SuperUser");

#### superAdmin.login();

#### superAdmin.resetPassword();

#### superAdmin.blockUser();

#### superAdmin.grantPermissions();

#### superAdmin.accessLogs();

#### superAdmin.logout();

#### }

#### }

#### OUTPUT:

#### 

#### 5B: MULTILEVEL INHERITANCE 2

#### CODE:

#### class AudioFile {

#### String title;

#### AudioFile(String title) {

#### this.title = title;

#### }

#### void play() {

#### System.out.println("Playing: " + title);

#### }

#### void pause() {

#### System.out.println("Paused: " + title);

#### }

#### }

#### class MusicTrack extends AudioFile {

#### String artist;

#### MusicTrack(String title, String artist) {

#### super(title);

#### this.artist = artist;

#### }

#### void displayArtist() {

#### System.out.println("Artist: " + artist);

#### }

#### }

#### class PremiumMusicTrack extends MusicTrack {

#### PremiumMusicTrack(String title, String artist) {

#### super(title, artist);

#### }

#### void download() {

#### System.out.println("Downloading: " + title);

#### }

#### void enableLosslessAudio() {

#### System.out.println("Lossless audio enabled for: " + title);

#### }

#### }

#### public class MI2 {

#### public static void main(String[] args) {

#### PremiumMusicTrack track = new PremiumMusicTrack("Shape of You", "Ed Sheeran");

#### track.play();

#### track.displayArtist();

#### track.download();

#### track.enableLosslessAudio();

#### track.pause();

#### }

#### }

#### OUTPUT:

#### 

#### 6A: HIERARCHICAL INHERITANCE 1

#### CODE:

#### class User {

#### String username;

#### User(String username) {

#### this.username = username;

#### }

#### void login() {

#### System.out.println(username + " logged in.");

#### }

#### void logout() {

#### System.out.println(username + " logged out.");

#### }

#### }

#### class Student extends User {

#### Student(String username) {

#### super(username);

#### }

#### void enrollCourse() {

#### System.out.println(username + " enrolled in a course.");

#### }

#### void submitAssignment() {

#### System.out.println(username + " submitted an assignment.");

#### }

#### }

#### class Teacher extends User {

#### Teacher(String username) {

#### super(username);

#### }

#### void createCourse() {

#### System.out.println(username + " created a course.");

#### }

#### void gradeAssignment() {

#### System.out.println(username + " graded an assignment.");

#### }

#### }

#### class Admin extends User {

#### Admin(String username) {

#### super(username);

#### }

#### void manageUsers() {

#### System.out.println(username + " managed users.");

#### }

#### void viewReports() {

#### System.out.println(username + " viewed reports.");

#### }

#### }

#### public class HI1 {

#### public static void main(String[] args) {

#### Student student = new Student("Alice");

#### student.login();

#### student.enrollCourse();

#### student.submitAssignment();

#### student.logout();

#### Teacher teacher = new Teacher("Bob");

#### teacher.login();

#### teacher.createCourse();

#### teacher.gradeAssignment();

#### teacher.logout();

#### Admin admin = new Admin("Charlie");

#### admin.login();

#### admin.manageUsers();

#### admin.viewReports();

#### admin.logout();

#### }

#### }

#### OUTPUT:

#### 

#### 6B: HIERARCHICAL INHERITANCE 2

#### CODE:

#### class Gadget {

#### String brand;

#### Gadget(String brand) {

#### this.brand = brand;

#### }

#### void powerOn() {

#### System.out.println(brand + " gadget powered on.");

#### }

#### void powerOff() {

#### System.out.println(brand + " gadget powered off.");

#### }

#### }

#### class Smartphone extends Gadget {

#### Smartphone(String brand) {

#### super(brand);

#### }

#### void makeCall() {

#### System.out.println(brand + " smartphone making a call.");

#### }

#### void installApp() {

#### System.out.println(brand + " smartphone installing an app.");

#### }

#### }

#### class Laptop extends Gadget {

#### Laptop(String brand) {

#### super(brand);

#### }

#### void compileCode() {

#### System.out.println(brand + " laptop compiling code.");

#### }

#### void runSoftware() {

#### System.out.println(brand + " laptop running software.");

#### }

#### }

#### class Tablet extends Gadget {

#### Tablet(String brand) {

#### super(brand);

#### }

#### void drawWithStylus() {

#### System.out.println(brand + " tablet drawing with stylus.");

#### }

#### void watchMovie() {

#### System.out.println(brand + " tablet watching a movie.");

#### }

#### }

#### public class HI2 {

#### public static void main(String[] args) {

#### Smartphone phone = new Smartphone("Samsung");

#### phone.powerOn();

#### phone.makeCall();

#### phone.installApp();

#### phone.powerOff();

#### Laptop laptop = new Laptop("Dell");

#### laptop.powerOn();

#### laptop.compileCode();

#### laptop.runSoftware();

#### laptop.powerOff();

#### Tablet tablet = new Tablet("Apple");

#### tablet.powerOn();

#### tablet.drawWithStylus();

#### tablet.watchMovie();

#### tablet.powerOff();

#### }

#### }

#### OUTPUT:

#### 

#### 7A: HYBRID INHERITANCE 1

#### CODE:

#### interface VideoPlayer {

#### void pause();

#### }

#### interface AudioPlayer {

#### void changeVolume(int level);

#### }

#### class MediaFile {

#### String title;

#### MediaFile(String title) {

#### this.title = title;

#### }

#### void play() {

#### System.out.println("Playing: " + title);

#### }

#### void stop() {

#### System.out.println("Stopped: " + title);

#### }

#### }

#### class MP3Player extends MediaFile implements AudioPlayer {

#### MP3Player(String title) {

#### super(title);

#### }

#### public void changeVolume(int level) {

#### System.out.println("MP3 Player volume set to: " + level);

#### }

#### }

#### class MP4Player extends MediaFile implements VideoPlayer, AudioPlayer {

#### MP4Player(String title) {

#### super(title);

#### }

#### public void pause() {

#### System.out.println("MP4 Player paused: " + title);

#### }

#### public void changeVolume(int level) {

#### System.out.println("MP4 Player volume set to: " + level);

#### }

#### }

#### class StreamingApp extends MediaFile implements VideoPlayer, AudioPlayer {

#### StreamingApp(String title) {

#### super(title);

#### }

#### public void pause() {

#### System.out.println("Streaming app paused: " + title);

#### }

#### public void changeVolume(int level) {

#### System.out.println("Streaming app volume set to: " + level);

#### }

#### }

#### public class HyI1{

#### public static void main(String[] args) {

#### MP3Player mp3 = new MP3Player("Song.mp3");

#### mp3.play();

#### mp3.changeVolume(50);

#### mp3.stop();

#### MP4Player mp4 = new MP4Player("Video.mp4");

#### mp4.play();

#### mp4.pause();

#### mp4.changeVolume(70);

#### mp4.stop();

#### StreamingApp app = new StreamingApp("Live Stream");

#### app.play();

#### app.pause();

#### app.changeVolume(80);

#### app.stop();

#### }

#### }

#### OUTPUT:

#### 

#### 7B: HYBRID INHERITANCE 2

#### CODE:

#### interface Diagnosable {

#### void performDiagnosis();

#### }

#### interface Treatable {

#### void giveTreatment();

#### }

#### class Person {

#### String name;

#### int age;

#### Person(String name, int age) {

#### this.name = name;

#### this.age = age;

#### }

#### void getDetails() {

#### System.out.println("Name: " + name + ", Age: " + age);

#### }

#### }

#### class Doctor extends Person implements Diagnosable, Treatable {

#### Doctor(String name, int age) {

#### super(name, age);

#### }

#### public void performDiagnosis() {

#### System.out.println(name + " is diagnosing a patient.");

#### }

#### public void giveTreatment() {

#### System.out.println(name + " is giving treatment.");

#### }

#### }

#### class Surgeon extends Doctor {

#### Surgeon(String name, int age) {

#### super(name, age);

#### }

#### void performSurgery() {

#### System.out.println(name + " is performing surgery.");

#### }

#### }

#### class Nurse extends Person implements Treatable {

#### Nurse(String name, int age) {

#### super(name, age);

#### }

#### public void giveTreatment() {

#### System.out.println(name + " is administering medication.");

#### }

#### }

#### public class HyI2 {

#### public static void main(String[] args) {

#### Doctor doctor = new Doctor("Dr. Smith", 45);

#### doctor.getDetails();

#### doctor.performDiagnosis();

#### doctor.giveTreatment();

#### Surgeon surgeon = new Surgeon("Dr. Adams", 50);

#### surgeon.getDetails();

#### surgeon.performDiagnosis();

#### surgeon.giveTreatment();

#### surgeon.performSurgery();

#### Nurse nurse = new Nurse("Alice", 30);

#### nurse.getDetails();

#### nurse.giveTreatment();

#### }

#### }

#### OUTPUT:

#### 

#### 8A: CONSTRUCTOR OVERRIDING PROGRAM 1

#### CODE:

#### class Notification {

#### Notification() {

#### sendAlert();

#### }

#### 

#### void sendAlert() {

#### System.out.println("Sending general notification");

#### }

#### }

#### class EmailNotification extends Notification {

#### EmailNotification() {

#### super();

#### }

#### 

#### @Override

#### void sendAlert() {

#### System.out.println("Sending email notification");

#### }

#### }

#### class SMSNotification extends Notification {

#### SMSNotification() {

#### super();

#### }

#### 

#### @Override

#### void sendAlert() {

#### System.out.println("Sending SMS notification");

#### }

#### }

#### class PushNotification extends Notification {

#### PushNotification() {

#### super();

#### }

#### 

#### @Override

#### void sendAlert() {

#### System.out.println("Sending push notification");

#### }

#### }

#### public class COR1 {

#### public static void main(String[] args) {

#### new EmailNotification();

#### new SMSNotification();

#### new PushNotification();

#### }

#### }

#### OUTPUT:

#### 

#### 8B: CONSTRUCTOR OVERRIDING PROGRAM 2

#### CODE:

class Translator {

Translator(String text) {

translate(text);

}

void translate(String text) {

System.out.println("Translating text: " + text);

}

}

class EnglishToFrench extends Translator {

EnglishToFrench(String text) {

super(text);

}

@Override

void translate(String text) {

System.out.println("Translation to French: Traduction en français");

}

}

class EnglishToSpanish extends Translator {

EnglishToSpanish(String text) {

super(text);

}

@Override

void translate(String text) {

System.out.println("Translation to Spanish: Traducción al español");

}

}

class EnglishToGerman extends Translator {

EnglishToGerman(String text) {

super(text);

}

@Override

void translate(String text) {

System.out.println("Translation to German: Übersetzung ins Deutsche");

}

}

public class COR2 {

public static void main(String[] args) {

new EnglishToFrench("Hello");

new EnglishToSpanish("Hello");

new EnglishToGerman("Hello");

}

#### }}

#### OUTPUT:

#### 

#### 9A: Constructor Overloading Program 1

#### CODE:

#### class SmartLight {

#### String color;

#### int brightness;

#### int duration;

#### SmartLight() {

#### this.color = "White";

#### this.brightness = 100;

#### this.duration = 0;

#### System.out.println("Light turned on: " + color + " at brightness " + brightness);

#### }

#### SmartLight(String color) {

#### this.color = color;

#### this.brightness = 100;

#### this.duration = 0;

#### System.out.println("Light turned on: " + color + " at brightness " + brightness);

#### }

#### SmartLight(String color, int brightness) {

#### this.color = color;

#### this.brightness = brightness;

#### this.duration = 0;

#### System.out.println("Light turned on: " + color + " at brightness " + brightness);

#### }

#### SmartLight(String color, int brightness, int duration) {

#### this.color = color;

#### this.brightness = brightness;

#### this.duration = duration;

#### System.out.println("Light turned on: " + color + " at brightness " + brightness + " for " + duration + " minutes");

#### }

#### }

#### public class COL1 {

#### public static void main(String[] args) {

#### new SmartLight();

#### new SmartLight("Blue");

#### new SmartLight("Red", 75);

#### new SmartLight("Green", 50, 30);

#### }

#### }

#### OUTPUT:

#### 

#### 9B: Constructor Overloading Program 2

#### CODE:

#### class LoanEMICalculator {

#### double principal;

#### double rate;

#### int time;

#### int downPayment;

#### LoanEMICalculator(double principal, double rate) {

#### this.principal = principal;

#### this.rate = rate;

#### this.time = 12;

#### this.downPayment = 0;

#### calculateEMI();

#### }

#### LoanEMICalculator(double principal, double rate, int time) {

#### this.principal = principal;

#### this.rate = rate;

#### this.time = time;

#### this.downPayment = 0;

#### calculateEMI();

#### }

#### LoanEMICalculator(double principal, double rate, int time, int downPayment) {

#### this.principal = principal - downPayment;

#### this.rate = rate;

#### this.time = time;

#### this.downPayment = downPayment;

#### calculateEMI();

#### }

#### void calculateEMI() {

#### double monthlyRate = rate / (12 \* 100);

#### double emi = (principal \* monthlyRate \* Math.pow(1 + monthlyRate, time)) / (Math.pow(1 + monthlyRate, time) - 1);

#### System.out.println("EMI: " + emi);

#### }

#### }

#### public class COL2 {

#### public static void main(String[] args) {

#### new LoanEMICalculator(50000, 5.5);

#### new LoanEMICalculator(50000, 5.5, 24);

#### new LoanEMICalculator(50000, 5.5, 24, 5000);

#### }

#### }

#### OUTPUT:

#### 

#### 10A: Method overloading 1

#### CODE:

#### class QuizScorer {

#### int calculateScore(int correct) {

#### return correct \* 10;

#### }

#### int calculateScore(int correct, int total) {

#### return (int) (((double) correct / total) \* 100);

#### }

#### int calculateScore(int correct, int total, int difficultyLevel) {

#### return (int) (((double) correct / total) \* 100 \* difficultyLevel);

#### }

#### }

#### public class OL1{

#### public static void main(String[] args) {

#### QuizScorer scorer = new QuizScorer();

#### System.out.println("Score: " + scorer.calculateScore(8));

#### System.out.println("Score: " + scorer.calculateScore(8, 10));

#### System.out.println("Score: " + scorer.calculateScore(8, 10, 2));

#### }

#### }

#### OUTPUT:

#### 

**10B:** **method overloading 2**

**CODE:**

class UserLoginLogger {

void logIn(String username) {

System.out.println(username + " logged in.");

}

void logIn(String username, String password) {

System.out.println(username + " logged in with password.");

}

void logIn(String username, String password, String location) {

System.out.println(username + " logged in from " + location + ".");

}

}

public class OL2 {

public static void main(String[] args) {

UserLoginLogger logger = new UserLoginLogger();

logger.logIn("Alice");

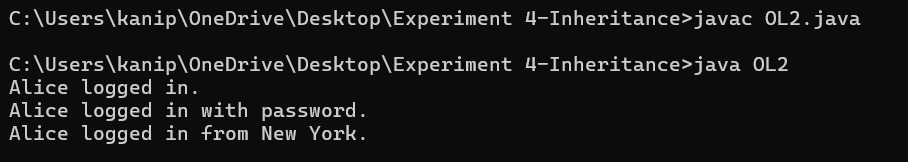
logger.logIn("Alice", "password123");

logger.logIn("Alice", "password123", "New York");

}

}

**OUTPUT:**



**11A:** **method overriding 1**

**CODE:**

class GameCharacter {

void attackMove() {

System.out.println("GameCharacter attacks!");

}

}

class Warrior extends GameCharacter {

@Override

void attackMove() {

System.out.println("Warrior swings a sword!");

}

}

class Archer extends GameCharacter {

@Override

void attackMove() {

System.out.println("Archer shoots an arrow!");

}

}

class Mage extends GameCharacter {

@Override

void attackMove() {

System.out.println("Mage casts a spell!");

}

}

public class OR1 {

public static void main(String[] args) {

GameCharacter warrior = new Warrior();

GameCharacter archer = new Archer();

GameCharacter mage = new Mage();

warrior.attackMove();

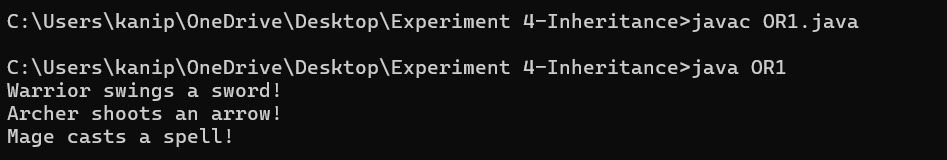
archer.attackMove();

mage.attackMove();

}

}

**OUTPUT:**



**11B:** **method overriding 2**

**CODE:**

class Booking {

void calculateFare() {

System.out.println("Calculating fare for booking");

}

}

class FlightBooking extends Booking {

@Override

void calculateFare() {

System.out.println("Calculating fare for flight booking");

}

}

class TrainBooking extends Booking {

@Override

void calculateFare() {

System.out.println("Calculating fare for train booking");

}

}

class BusBooking extends Booking {

@Override

void calculateFare() {

System.out.println("Calculating fare for bus booking");

}

}

public class OR2 {

public static void main(String[] args) {

Booking flight = new FlightBooking();

Booking train = new TrainBooking();

Booking bus = new BusBooking();

flight.calculateFare();

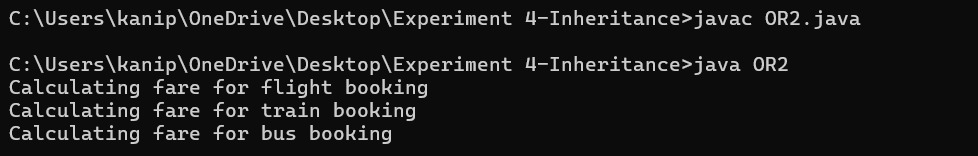
train.calculateFare();

bus.calculateFare();

}

}

**OUTPUT:**



**12A:** **interface program 1**

**CODE:**

interface DeliveryPartner {

void acceptOrder();

void trackOrder();

void deliverOrder();

}

class Zomato implements DeliveryPartner {

@Override

public void acceptOrder() {

System.out.println("Zomato: Order Accepted");

}

@Override

public void trackOrder() {

System.out.println("Zomato: Tracking Order");

}

@Override

public void deliverOrder() {

System.out.println("Zomato: Order Delivered");

}

}

class Swiggy implements DeliveryPartner {

@Override

public void acceptOrder() {

System.out.println("Swiggy: Order Accepted");

}

@Override

public void trackOrder() {

System.out.println("Swiggy: Tracking Order");

}

@Override

public void deliverOrder() {

System.out.println("Swiggy: Order Delivered");

}

}

class UberEats implements DeliveryPartner {

@Override

public void acceptOrder() {

System.out.println("UberEats: Order Accepted");

}

@Override

public void trackOrder() {

System.out.println("UberEats: Tracking Order");

}

@Override

public void deliverOrder() {

System.out.println("UberEats: Order Delivered");

}

}

public class I1 {

public static void main(String[] args) {

DeliveryPartner zomato = new Zomato();

zomato.acceptOrder();

zomato.trackOrder();

zomato.deliverOrder();

DeliveryPartner swiggy = new Swiggy();

swiggy.acceptOrder();

swiggy.trackOrder();

swiggy.deliverOrder();

DeliveryPartner uberEats = new UberEats();

uberEats.acceptOrder();

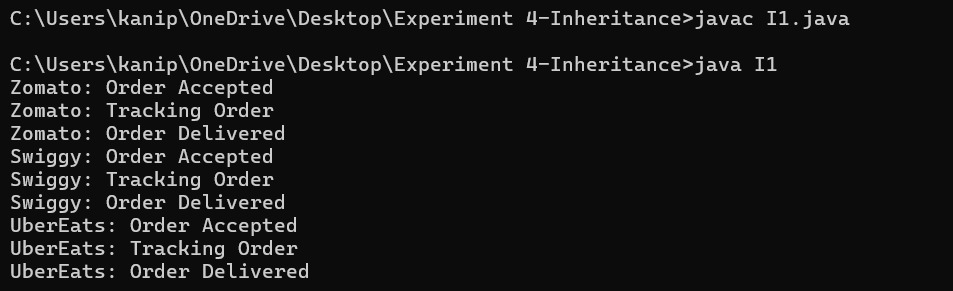
uberEats.trackOrder();

uberEats.deliverOrder();

}

}

**OUTPUT:**



**12B:** **interface program 2**

**CODE:**

interface Streamable {

void play();

void pause();

void recommendContent();

}

class Netflix implements Streamable {

@Override

public void play() {

System.out.println("Netflix: Playing video");

}

@Override

public void pause() {

System.out.println("Netflix: Video paused");

}

@Override

public void recommendContent() {

System.out.println("Netflix: Recommending movies and shows");

}

}

class Spotify implements Streamable {

@Override

public void play() {

System.out.println("Spotify: Playing music");

}

@Override

public void pause() {

System.out.println("Spotify: Music paused");

}

@Override

public void recommendContent() {

System.out.println("Spotify: Recommending playlists and songs");

}

}

class YouTube implements Streamable {

@Override

public void play() {

System.out.println("YouTube: Playing video");

}

@Override

public void pause() {

System.out.println("YouTube: Video paused");

}

@Override

public void recommendContent() {

System.out.println("YouTube: Recommending trending videos");

}

}

public class I2 {

public static void main(String[] args) {

Streamable netflix = new Netflix();

netflix.play();

netflix.pause();

netflix.recommendContent();

Streamable spotify = new Spotify();

spotify.play();

spotify.pause();

spotify.recommendContent();

Streamable youtube = new YouTube();

youtube.play();

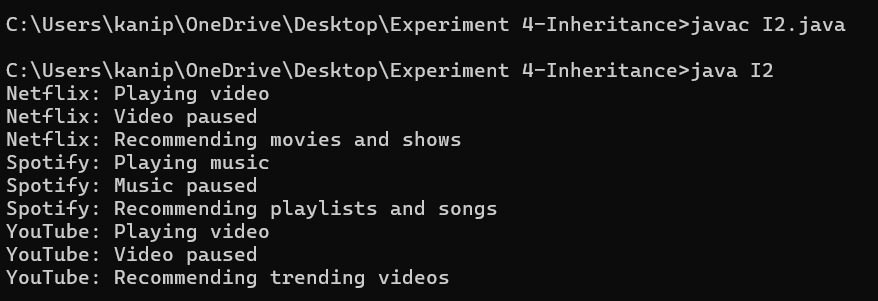
youtube.pause();

youtube.recommendContent();

}

}

**OUTPUT:**



**12C:** **interface program 3**

**CODE:**

interface ControllableDevice {

void turnOn();

void turnOff();

void connectToApp();

}

class SmartFan implements ControllableDevice {

@Override

public void turnOn() {

System.out.println("SmartFan: Turned On");

}

@Override

public void turnOff() {

System.out.println("SmartFan: Turned Off");

}

@Override

public void connectToApp() {

System.out.println("SmartFan: Connected to App");

}

}

class SmartBulb implements ControllableDevice {

@Override

public void turnOn() {

System.out.println("SmartBulb: Turned On");

}

@Override

public void turnOff() {

System.out.println("SmartBulb: Turned Off");

}

@Override

public void connectToApp() {

System.out.println("SmartBulb: Connected to App");

}

}

class SmartDoor implements ControllableDevice {

@Override

public void turnOn() {

System.out.println("SmartDoor: Unlocked");

}

@Override

public void turnOff() {

System.out.println("SmartDoor: Locked");

}

@Override

public void connectToApp() {

System.out.println("SmartDoor: Connected to App");

}

}

public class I3{

public static void main(String[] args) {

ControllableDevice fan = new SmartFan();

fan.turnOn();

fan.turnOff();

fan.connectToApp();

ControllableDevice bulb = new SmartBulb();

bulb.turnOn();

bulb.turnOff();

bulb.connectToApp();

ControllableDevice door = new SmartDoor();

door.turnOn();

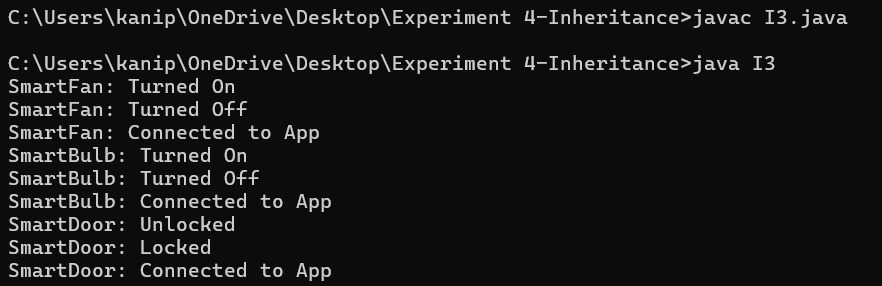
door.turnOff();

door.connectToApp();

}

}

**OUTPUT:**



**12D:** **interface program 4**

**CODE:**

interface Ridable {

void bookRide();

void calculateFare();

void cancelRide();

}

class Bike implements Ridable {

@Override

public void bookRide() {

System.out.println("Bike: Ride Booked");

}

@Override

public void calculateFare() {

System.out.println("Bike: Calculating Fare");

}

@Override

public void cancelRide() {

System.out.println("Bike: Ride Cancelled");

}

}

class Auto implements Ridable {

@Override

public void bookRide() {

System.out.println("Auto: Ride Booked");

}

@Override

public void calculateFare() {

System.out.println("Auto: Calculating Fare");

}

@Override

public void cancelRide() {

System.out.println("Auto: Ride Cancelled");

}

}

class Cab implements Ridable {

@Override

public void bookRide() {

System.out.println("Cab: Ride Booked");

}

@Override

public void calculateFare() {

System.out.println("Cab: Calculating Fare");

}

@Override

public void cancelRide() {

System.out.println("Cab: Ride Cancelled");

}

}

public class I4 {

public static void main(String[] args) {

Ridable bike = new Bike();

bike.bookRide();

bike.calculateFare();

bike.cancelRide();

Ridable auto = new Auto();

auto.bookRide();

auto.calculateFare();

auto.cancelRide();

Ridable cab = new Cab();

cab.bookRide();

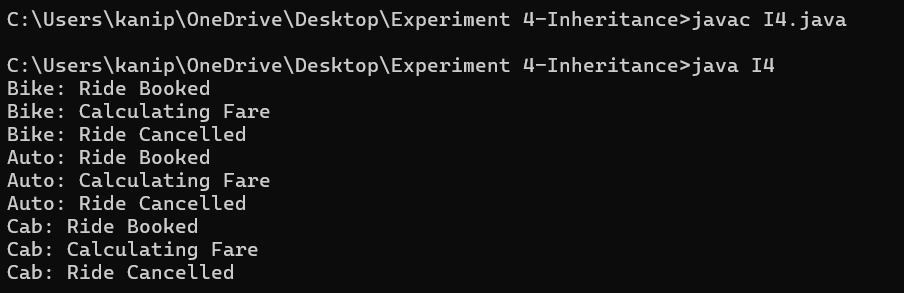
cab.calculateFare();

cab.cancelRide();

}

}

**OUTPUT:**



**13A:** **abstract program 1**

**CODE:**

abstract class HealthMonitor {

abstract void startTracking();

abstract void getStats();

abstract void alert();

}

class HeartRateMonitor extends HealthMonitor {

@Override

void startTracking() {

System.out.println("Heart Rate Monitoring Started");

}

@Override

void getStats() {

System.out.println("Current Heart Rate: 75 BPM");

}

@Override

void alert() {

System.out.println("Heart Rate Alert: Abnormal reading detected!");

}

}

class SleepTracker extends HealthMonitor {

@Override

void startTracking() {

System.out.println("Sleep Tracking Started");

}

@Override

void getStats() {

System.out.println("Last Night's Sleep: 7 hours");

}

@Override

void alert() {

System.out.println("Sleep Alert: Poor sleep quality detected!");

}

}

class StepCounter extends HealthMonitor {

@Override

void startTracking() {

System.out.println("Step Counting Started");

}

@Override

void getStats() {

System.out.println("Today's Steps: 10,000");

}

@Override

void alert() {

System.out.println("Step Goal Alert: Daily step goal reached!");

}

}

public class AC4 {

public static void main(String[] args) {

HealthMonitor heartRate = new HeartRateMonitor();

heartRate.startTracking();

heartRate.getStats();

heartRate.alert();

HealthMonitor sleep = new SleepTracker();

sleep.startTracking();

sleep.getStats();

sleep.alert();

HealthMonitor step = new StepCounter();

step.startTracking();

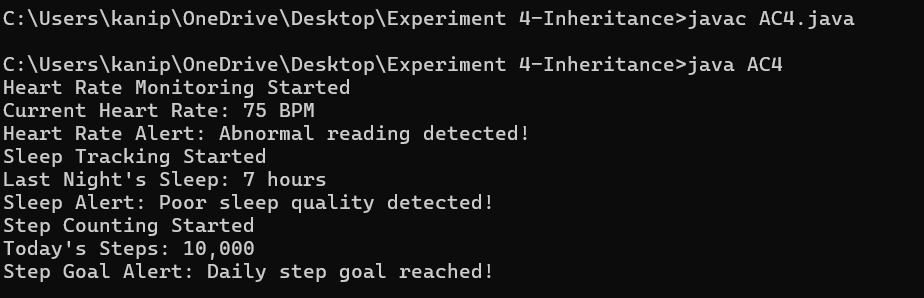
step.getStats();

step.alert();

}

}

**OUTPUT:**



**13B:** **abstract program 2**

**CODE:**

abstract class Instrument {

abstract void playNote();

abstract void record();

abstract void adjustVolume();

}

class Piano extends Instrument {

@Override

void playNote() {

System.out.println("Playing note on Piano");

}

@Override

void record() {

System.out.println("Recording Piano performance");

}

@Override

void adjustVolume() {

System.out.println("Adjusting Piano volume");

}

}

class Guitar extends Instrument {

@Override

void playNote() {

System.out.println("Playing note on Guitar");

}

@Override

void record() {

System.out.println("Recording Guitar performance");

}

@Override

void adjustVolume() {

System.out.println("Adjusting Guitar volume");

}

}

class DrumMachine extends Instrument {

@Override

void playNote() {

System.out.println("Playing beat on Drum Machine");

}

@Override

void record() {

System.out.println("Recording Drum Machine performance");

}

@Override

void adjustVolume() {

System.out.println("Adjusting Drum Machine volume");

}

}

public class AC3 {

public static void main(String[] args) {

Instrument piano = new Piano();

piano.playNote();

piano.record();

piano.adjustVolume();

Instrument guitar = new Guitar();

guitar.playNote();

guitar.record();

guitar.adjustVolume();

Instrument drumMachine = new DrumMachine();

drumMachine.playNote();

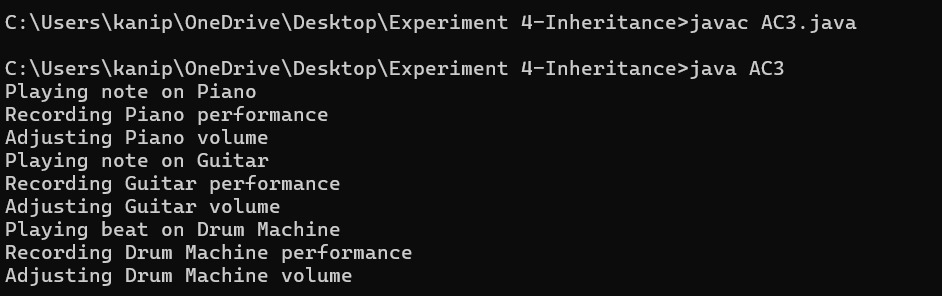
drumMachine.record();

drumMachine.adjustVolume();

}

}

**OUTPUT:**



**13C:** **abstract program 3**

**CODE:**

abstract class Booking {

abstract void confirmBooking();

abstract void cancelBooking();

abstract void getDetails();

}

class FlightBooking extends Booking {

@Override

void confirmBooking() {

System.out.println("Flight booking confirmed");

}

@Override

void cancelBooking() {

System.out.println("Flight booking canceled");

}

@Override

void getDetails() {

System.out.println("Flight booking details");

}

}

class HotelBooking extends Booking {

@Override

void confirmBooking() {

System.out.println("Hotel booking confirmed");

}

@Override

void cancelBooking() {

System.out.println("Hotel booking canceled");

}

@Override

void getDetails() {

System.out.println("Hotel booking details");

}

}

class TrainBooking extends Booking {

@Override

void confirmBooking() {

System.out.println("Train booking confirmed");

}

@Override

void cancelBooking() {

System.out.println("Train booking canceled");

}

@Override

void getDetails() {

System.out.println("Train booking details");

}

}

public class AC2 {

public static void main(String[] args) {

Booking flight = new FlightBooking();

flight.confirmBooking();

flight.getDetails();

flight.cancelBooking();

Booking hotel = new HotelBooking();

hotel.confirmBooking();

hotel.getDetails();

hotel.cancelBooking();

Booking train = new TrainBooking();

train.confirmBooking();

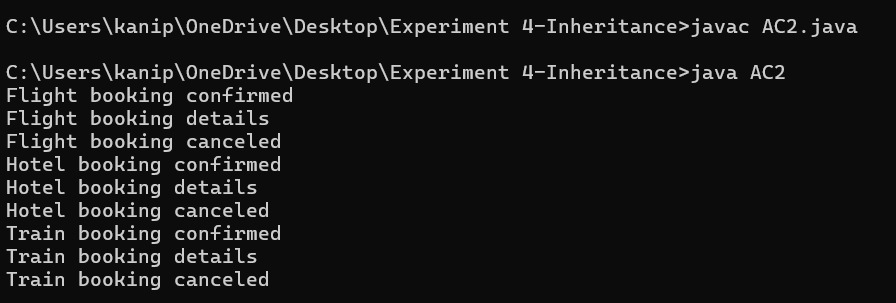
train.getDetails();

train.cancelBooking();

}

}

**OUTPUT:**



**13D:** **abstract program 4**

**CODE:**

abstract class Course {

abstract void enroll();

abstract void getContent();

abstract void completeModule();

}

class VideoCourse extends Course {

@Override

void enroll() {

System.out.println("Enrolled in Video Course");

}

@Override

void getContent() {

System.out.println("Accessing Video Content");

}

@Override

void completeModule() {

System.out.println("Video Module Completed");

}

}

class LiveCourse extends Course {

@Override

void enroll() {

System.out.println("Enrolled in Live Course");

}

@Override

void getContent() {

System.out.println("Joining Live Session");

}

@Override

void completeModule() {

System.out.println("Live Session Completed");

}

}

class TextCourse extends Course {

@Override

void enroll() {

System.out.println("Enrolled in Text Course");

}

@Override

void getContent() {

System.out.println("Reading Text Material");

}

@Override

void completeModule() {

System.out.println("Text Module Completed");

}

}

public class AC1 {

public static void main(String[] args) {

Course video = new VideoCourse();

video.enroll();

video.getContent();

video.completeModule();

Course live = new LiveCourse();

live.enroll();

live.getContent();

live.completeModule();

Course text = new TextCourse();

text.enroll();

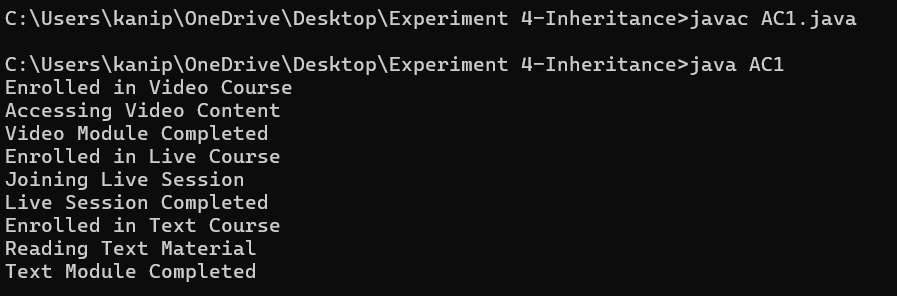
text.getContent();

text.completeModule();

}

}

**OUTPUT:**



**14A:** **encapsulation 1**

**CODE:**

class Employee {

private String name;

private String employeeId;

private double salary;

public Employee(String name, String employeeId, double salary) {

this.name = name;

this.employeeId = employeeId;

this.salary = salary;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getEmployeeId() {

return employeeId;

}

public void setEmployeeId(String employeeId) {

this.employeeId = employeeId;

}

public double getSalary() {

return salary;

}

public void setSalary(double salary) {

this.salary = salary;

}

public void generateSalarySlip() {

System.out.println("Employee Salary Slip");

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

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

System.out.println("Salary: $" + salary);

}

}

public class E4 {

public static void main(String[] args) {

Employee emp = new Employee("John Doe", "EMP123", 5000);

emp.generateSalarySlip();

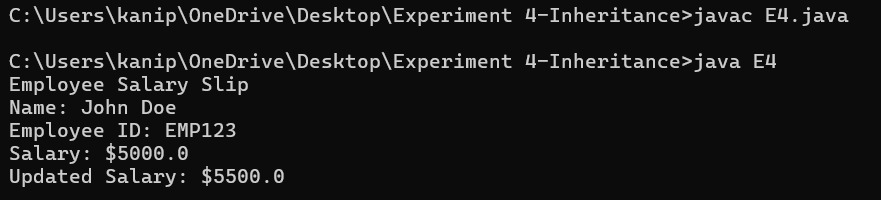
emp.setSalary(5500);

System.out.println("Updated Salary: $" + emp.getSalary());

}

}

**OUTPUT:**



**14B:** **encapsulation 2**

**CODE:**

class ATMMachine {

private double balance;

private String pin;

public ATMMachine(double balance, String pin) {

this.balance = balance;

this.pin = pin;

}

public double getBalance(String enteredPin) {

if (this.pin.equals(enteredPin)) {

return balance;

} else {

System.out.println("Incorrect PIN");

return -1;

}

}

public void setBalance(String enteredPin, double amount) {

if (this.pin.equals(enteredPin)) {

this.balance = amount;

System.out.println("Balance updated successfully");

} else {

System.out.println("Incorrect PIN");

}

}

public void withdraw(String enteredPin, double amount) {

if (this.pin.equals(enteredPin)) {

if (amount <= balance) {

balance -= amount;

System.out.println("Withdrawal successful. Remaining balance: " + balance);

} else {

System.out.println("Insufficient balance");

}

} else {

System.out.println("Incorrect PIN");

}

}

}

public class E3 {

public static void main(String[] args) {

ATMMachine atm = new ATMMachine(5000, "1234");

System.out.println("Current Balance: " + atm.getBalance("1234"));

atm.withdraw("1234", 1000);

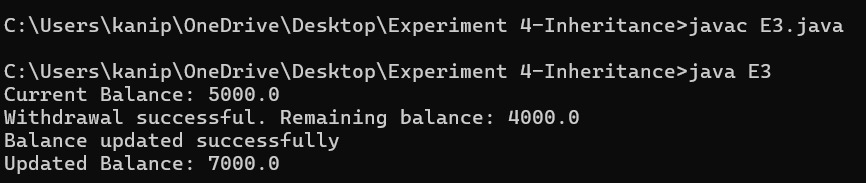
atm.setBalance("1234", 7000);

System.out.println("Updated Balance: " + atm.getBalance("1234"));

}

}

**OUTPUT:**



**14C:** **encapsulation 3**

**CODE:**

class PackageDeliveryTracker {

private String trackingNumber;

private String status;

public PackageDeliveryTracker(String trackingNumber, String status) {

this.trackingNumber = trackingNumber;

this.status = status;

}

public String getTrackingNumber() {

return trackingNumber;

}

public void setTrackingNumber(String trackingNumber) {

this.trackingNumber = trackingNumber;

}

public String getStatus() {

return status;

}

public void setStatus(String status) {

this.status = status;

}

}

public class E2 {

public static void main(String[] args) {

PackageDeliveryTracker package1 = new PackageDeliveryTracker("123456", "In Transit");

System.out.println("Tracking Number: " + package1.getTrackingNumber());

System.out.println("Status: " + package1.getStatus());

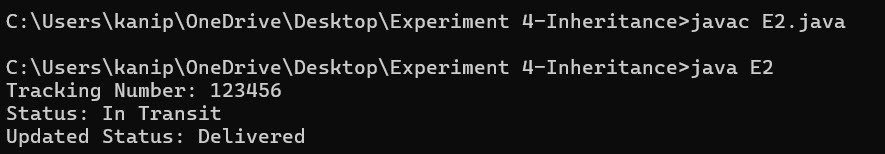
package1.setStatus("Delivered");

System.out.println("Updated Status: " + package1.getStatus());

}

}

**OUTPUT:**



**14D:** **encapsulation 4**

**CODE:**

import java.util.Scanner;

public class E1 {

private String lockCode;

private boolean isLocked;

public String getLockCode() {

return lockCode;

}

public void setLockCode(String lockCode) {

this.lockCode = lockCode;

}

public boolean isLocked() {

return isLocked;

}

public void setLocked(boolean locked) {

isLocked = locked;

}

public boolean unlock(String code) {

if (code.equals(lockCode)) {

isLocked = false;

return true;

}

return false;

}

public void lock() {

isLocked = true;

}

public static void main(String[] args) {

E1 lock = new E1();

Scanner scanner = new Scanner(System.in);

lock.setLockCode("1234");

lock.setLocked(true);

System.out.println("Enter code to unlock:");

String input = scanner.nextLine();

if (lock.unlock(input)) {

System.out.println("Unlocked successfully!");

} else {

System.out.println("Incorrect code. Lock remains engaged.");

}

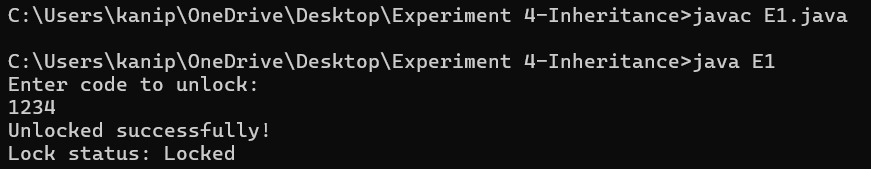
lock.lock();

System.out.println("Lock status: " + (lock.isLocked() ? "Locked" : "Unlocked"));

}

}

**OUTPUT;**



**15A:** **user-defined package 1**

**CODE:**

**1.**

package mathematics;

public class Calculator {

public int add(int a, int b) {

return a + b;

}

public int subtract(int a, int b) {

return a - b;

}

public int multiply(int a, int b) {

return a \* b;

}

public double divide(int a, int b) {

if (b == 0) {

throw new ArithmeticException("Cannot divide by zero");

}

return (double) a / b;

}

}

**2.**

package mathematics;

public class AdvancedCalculator {

public double power(double base, double exponent) {

return Math.pow(base, exponent);

}

public double squareRoot(double number) {

if (number < 0) {

throw new IllegalArgumentException("Cannot calculate square root of negative number");

}

return Math.sqrt(number);

}

public long factorial(int n) {

if (n < 0) {

throw new IllegalArgumentException("Cannot calculate factorial of negative number");

}

if (n == 0 || n == 1) {

return 1;

}

return n \* factorial(n - 1);

}

}

**Main program:**

import mathematics.Calculator;

import mathematics.AdvancedCalculator;

public class P3 {

public static void main(String[] args) {

// Using basic calculator

Calculator calc = new Calculator();

System.out.println("Basic Calculator Operations:");

System.out.println("Addition: " + calc.add(10, 5));

System.out.println("Subtraction: " + calc.subtract(10, 5));

System.out.println("Multiplication: " + calc.multiply(10, 5));

System.out.println("Division: " + calc.divide(10, 5));

// Using advanced calculator

AdvancedCalculator advCalc = new AdvancedCalculator();

System.out.println("\nAdvanced Calculator Operations:");

System.out.println("Power (2^3): " + advCalc.power(2, 3));

System.out.println("Square Root of 16: " + advCalc.squareRoot(16));

System.out.println("Factorial of 5: " + advCalc.factorial(5));

try {

System.out.println(calc.divide(10, 0)); // This will throw an exception

} catch (ArithmeticException e) {

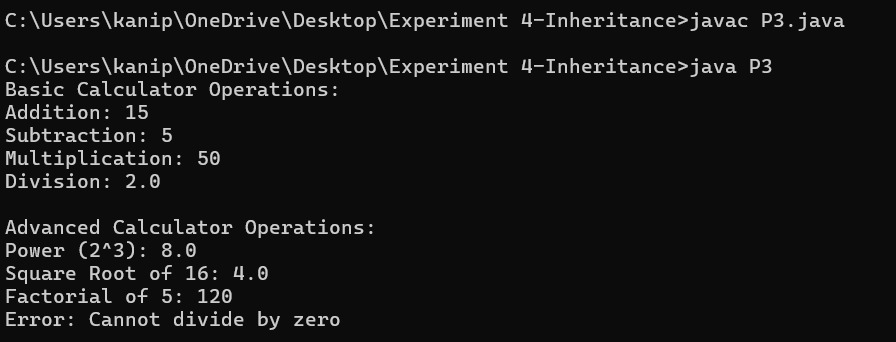
System.out.println("Error: " + e.getMessage());

}

}

}

**OUTPUT:**



**15B:** **user-defined package 2**

**CODE:**

**1.**

package school;

public class Student {

private int rollNo;

private String name;

private int age;

private String[] subjects;

private int[] marks;

public Student(int rollNo, String name, int age, String[] subjects, int[] marks) {

this.rollNo = rollNo;

this.name = name;

this.age = age;

this.subjects = subjects;

this.marks = marks;

}

public double calculateAverage() {

int sum = 0;

for(int mark : marks) {

sum += mark;

}

return (double) sum / marks.length;

}

public String getGrade() {

double avg = calculateAverage();

if(avg >= 90) return "A+";

else if(avg >= 80) return "A";

else if(avg >= 70) return "B";

else if(avg >= 60) return "C";

else if(avg >= 50) return "D";

else return "F";

}

public void displayInfo() {

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

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

System.out.println("Age: " + age);

System.out.println("\nSubject-wise Marks:");

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

System.out.println(subjects[i] + ": " + marks[i]);

}

System.out.println("Average: " + calculateAverage());

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

}

}

**2.**

package school;

import java.util.ArrayList;

public class StudentManager {

private ArrayList<Student> students;

public StudentManager() {

students = new ArrayList<>();

}

public void addStudent(Student student) {

students.add(student);

}

public void displayAllStudents() {

if(students.isEmpty()) {

System.out.println("No students registered yet!");

return;

}

System.out.println("\n=== All Students Information ===\n");

for(Student student : students) {

student.displayInfo();

System.out.println("================================");

}

}

public int getTotalStudents() {

return students.size();

}

}

**main program :**

import school.Student;

import school.StudentManager;

public class P4 {

public static void main(String[] args) {

StudentManager manager = new StudentManager();

// Create sample student data

String[] subjects1 = {"Math", "Science", "English", "History"};

int[] marks1 = {95, 87, 89, 92};

Student student1 = new Student(101, "John Doe", 18, subjects1, marks1);

String[] subjects2 = {"Math", "Science", "English", "History"};

int[] marks2 = {78, 85, 80, 75};

Student student2 = new Student(102, "Jane Smith", 17, subjects2, marks2);

// Add students to manager

manager.addStudent(student1);

manager.addStudent(student2);

// Display total number of students

System.out.println("Total Students: " + manager.getTotalStudents());

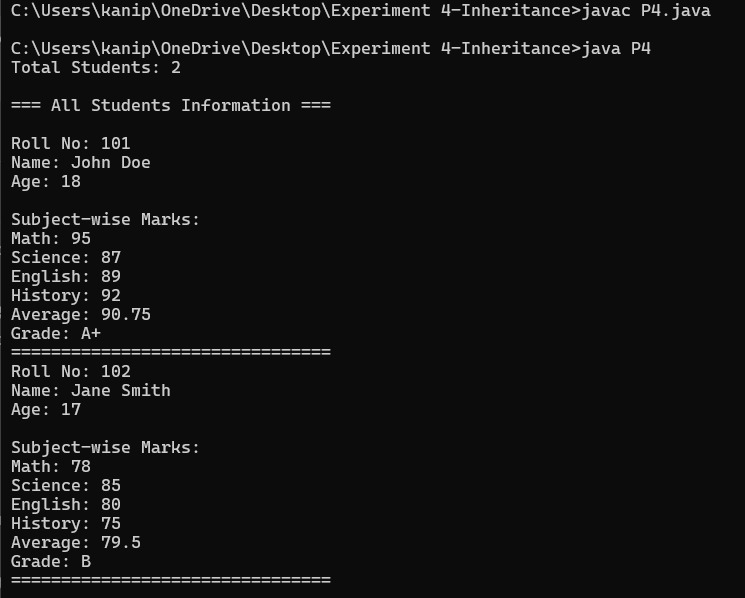
// Display all students' information

manager.displayAllStudents();

}

}

**OUTPUT:**



**15C:** **in-built package 3**

**CODE:**

import java.util.\*;

class Task {

private String title;

private String description;

private Date dueDate;

private boolean isCompleted;

public Task(String title, String description, Date dueDate) {

this.title = title;

this.description = description;

this.dueDate = dueDate;

this.isCompleted = false;

}

public String getTitle() { return title; }

public String getDescription() { return description; }

public Date getDueDate() { return dueDate; }

public boolean isCompleted() { return isCompleted; }

public void setCompleted(boolean completed) { isCompleted = completed; }

@Override

public String toString() {

return String.format("Task: %s\nDescription: %s\nDue Date: %s\nStatus: %s\n",

title, description, dueDate, isCompleted ? "Completed" : "Pending");

}

}

public class P1 {

private static ArrayList<Task> tasks = new ArrayList<>();

private static Scanner scanner = new Scanner(System.in);

public static void main(String[] args) {

while (true) {

System.out.println("\n=== Task Manager ===");

System.out.println("1. Add Task");

System.out.println("2. View Tasks");

System.out.println("3. Mark Task as Completed");

System.out.println("4. Sort Tasks by Due Date");

System.out.println("5. Exit");

System.out.print("Choose an option: ");

int choice = scanner.nextInt();

scanner.nextLine(); // Consume newline

switch (choice) {

case 1:

addTask();

break;

case 2:

viewTasks();

break;

case 3:

markTaskCompleted();

break;

case 4:

sortTasksByDueDate();

break;

case 5:

System.out.println("Goodbye!");

scanner.close();

System.exit(0);

default:

System.out.println("Invalid option!");

}

}

}

private static void addTask() {

System.out.print("Enter task title: ");

String title = scanner.nextLine();

System.out.print("Enter task description: ");

String description = scanner.nextLine();

// Creating a current date as due date for simplicity

Date dueDate = new Date();

Task task = new Task(title, description, dueDate);

tasks.add(task);

System.out.println("Task added successfully!");

}

private static void viewTasks() {

if (tasks.isEmpty()) {

System.out.println("No tasks available!");

return;

}

System.out.println("\n=== All Tasks ===");

for (int i = 0; i < tasks.size(); i++) {

System.out.println("\nTask #" + (i + 1));

System.out.println(tasks.get(i));

}

}

private static void markTaskCompleted() {

if (tasks.isEmpty()) {

System.out.println("No tasks available!");

return;

}

viewTasks();

System.out.print("Enter task number to mark as completed: ");

int taskNum = scanner.nextInt();

if (taskNum > 0 && taskNum <= tasks.size()) {

Task task = tasks.get(taskNum - 1);

task.setCompleted(true);

System.out.println("Task marked as completed!");

} else {

System.out.println("Invalid task number!");

}

}

private static void sortTasksByDueDate() {

Collections.sort(tasks, new Comparator<Task>() {

@Override

public int compare(Task t1, Task t2) {

return t1.getDueDate().compareTo(t2.getDueDate());

}

});

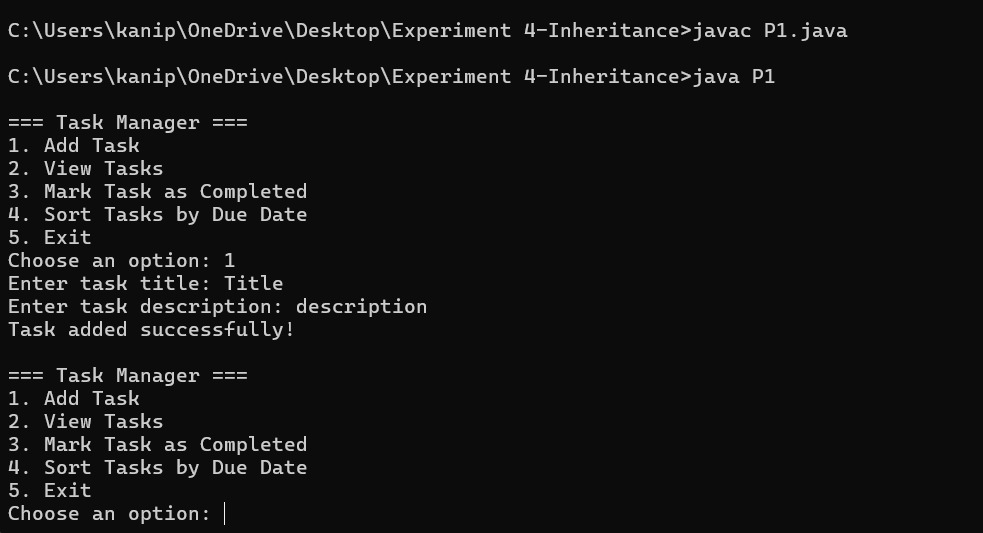
System.out.println("Tasks sorted by due date!");

viewTasks();

}

}

**OUTPUT:**



**15D:** **in-built package 4**

**CODE:**

import java.io.\*;

import java.util.Scanner;

public class P2 {

private static Scanner scanner = new Scanner(System.in);

private static final String BASE\_DIRECTORY = "files";

public static void main(String[] args) {

// Create base directory if it doesn't exist

createBaseDirectory();

while (true) {

System.out.println("\n=== File Management System ===");

System.out.println("1. Create a new file");

System.out.println("2. Write to a file");

System.out.println("3. Read from a file");

System.out.println("4. List all files");

System.out.println("5. Delete a file");

System.out.println("6. File information");

System.out.println("7. Exit");

System.out.print("Choose an option: ");

int choice = scanner.nextInt();

scanner.nextLine(); // Consume newline

switch (choice) {

case 1: createFile(); break;

case 2: writeToFile(); break;

case 3: readFromFile(); break;

case 4: listFiles(); break;

case 5: deleteFile(); break;

case 6: fileInformation(); break;

case 7:

System.out.println("Goodbye!");

scanner.close();

System.exit(0);

default:

System.out.println("Invalid option!");

}

}

}

private static void createBaseDirectory() {

File directory = new File(BASE\_DIRECTORY);

if (!directory.exists()) {

directory.mkdir();

System.out.println("Base directory created successfully!");

}

}

private static void createFile() {

System.out.print("Enter file name: ");

String fileName = scanner.nextLine();

File file = new File(BASE\_DIRECTORY + File.separator + fileName);

try {

if (file.createNewFile()) {

System.out.println("File created successfully!");

} else {

System.out.println("File already exists!");

}

} catch (IOException e) {

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

}

}

private static void writeToFile() {

System.out.print("Enter file name: ");

String fileName = scanner.nextLine();

File file = new File(BASE\_DIRECTORY + File.separator + fileName);

if (!file.exists()) {

System.out.println("File does not exist!");

return;

}

System.out.println("Enter text (type 'END' on a new line to finish):");

try (BufferedWriter writer = new BufferedWriter(new FileWriter(file, true))) {

String line;

while (!(line = scanner.nextLine()).equals("END")) {

writer.write(line);

writer.newLine();

}

System.out.println("Content written successfully!");

} catch (IOException e) {

System.out.println("Error writing to file: " + e.getMessage());

}

}

private static void readFromFile() {

System.out.print("Enter file name: ");

String fileName = scanner.nextLine();

File file = new File(BASE\_DIRECTORY + File.separator + fileName);

if (!file.exists()) {

System.out.println("File does not exist!");

return;

}

try (BufferedReader reader = new BufferedReader(new FileReader(file))) {

System.out.println("\nFile contents:");

System.out.println("==============");

String line;

while ((line = reader.readLine()) != null) {

System.out.println(line);

}

System.out.println("==============");

} catch (IOException e) {

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

}

}

private static void listFiles() {

File directory = new File(BASE\_DIRECTORY);

File[] files = directory.listFiles();

if (files == null || files.length == 0) {

System.out.println("No files found!");

return;

}

System.out.println("\nFiles in directory:");

for (File file : files) {

System.out.println(file.getName());

}

}

private static void deleteFile() {

System.out.print("Enter file name to delete: ");

String fileName = scanner.nextLine();

File file = new File(BASE\_DIRECTORY + File.separator + fileName);

if (file.exists()) {

if (file.delete()) {

System.out.println("File deleted successfully!");

} else {

System.out.println("Failed to delete file!");

}

} else {

System.out.println("File does not exist!");

}

}

private static void fileInformation() {

System.out.print("Enter file name: ");

String fileName = scanner.nextLine();

File file = new File(BASE\_DIRECTORY + File.separator + fileName);

if (!file.exists()) {

System.out.println("File does not exist!");

return;

}

System.out.println("\nFile Information:");

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

System.out.println("Path: " + file.getAbsolutePath());

System.out.println("Size: " + file.length() + " bytes");

System.out.println("Last Modified: " + new java.util.Date(file.lastModified()));

System.out.println("Is Directory: " + file.isDirectory());

System.out.println("Is File: " + file.isFile());

System.out.println("Can Read: " + file.canRead());

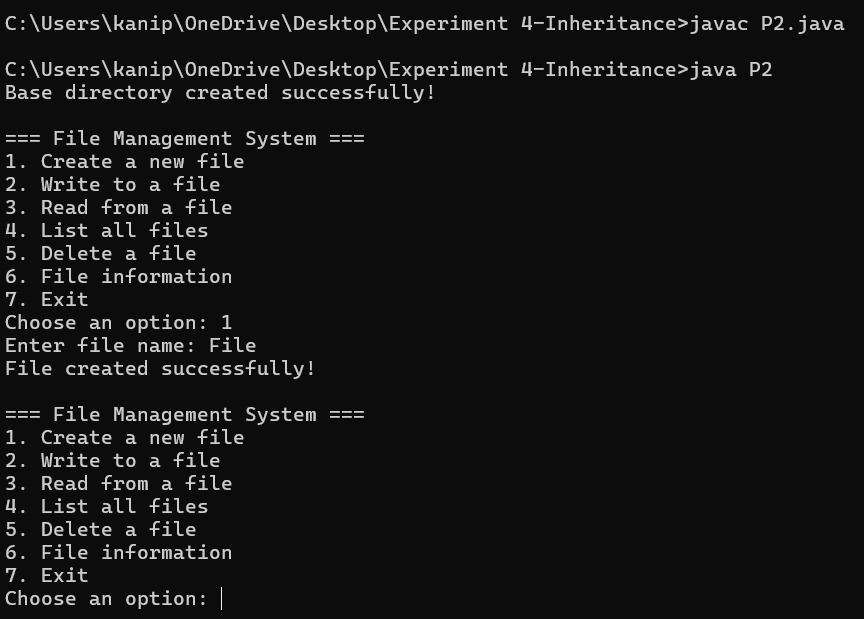
System.out.println("Can Write: " + file.canWrite());

System.out.println("Can Execute: " + file.canExecute());

}

}

**OUTPUT:**



**16A:** **exception handling 1**

**CODE:**

import java.util.Scanner;

class UnderageException extends Exception {

public UnderageException(String message) {

super(message);

}

}

public class EH1 {

public static void main(String[] args) {

Scanner scan = new Scanner(System.in);

System.out.print("Enter your age: ");

int age = scan.nextInt();

try {

if (age < 18) {

throw new UnderageException("You must be 18 or older to apply for Voter ID.");

} else {

System.out.println("You are eligible to apply for a Voter ID.");

}

} catch (UnderageException e) {

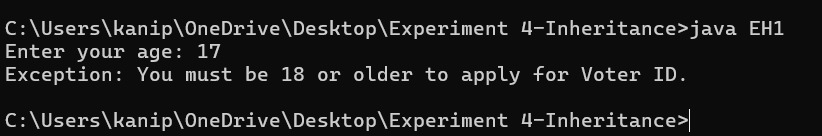
System.out.println("Exception: " + e.getMessage());

}

}

}

**OUTPUT:**



**16B:** **exception handling 2**

**CODE:**

import java.util.Scanner;

class InvalidMarksException extends Exception {

public InvalidMarksException(String message) {

super(message);

}

}

public class EH2 {

public static void main(String[] args) {

Scanner scan = new Scanner(System.in);

System.out.print("Enter marks (0 to 100): ");

int marks = scan.nextInt();

try {

if (marks < 0 || marks > 100) {

throw new InvalidMarksException("Marks must be between 0 and 100.");

} else {

System.out.println("Marks recorded: " + marks);

}

} catch (InvalidMarksException e) {

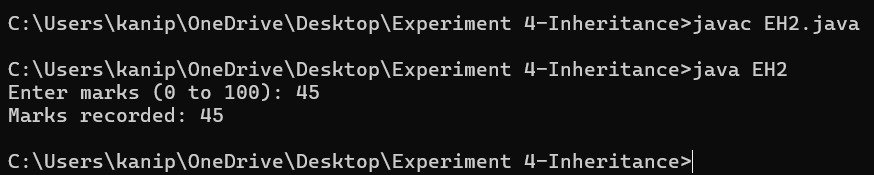
System.out.println("Exception: " + e.getMessage());

}

}

}

**OUTPUT:**



**16C:** **exception handling 3**

**CODE:**

import java.util.Scanner;

class UsernameTooShortException extends Exception {

public UsernameTooShortException(String message) {

super(message);

}

}

public class EH3 {

public static void main(String[] args) {

Scanner scan = new Scanner(System.in);

System.out.print("Enter your username: ");

String username = scan.nextLine();

try {

if (username.length() < 5) {

throw new UsernameTooShortException("Username must be at least 5 characters long.");

} else {

System.out.println("Username accepted: " + username);

}

} catch (UsernameTooShortException e) {

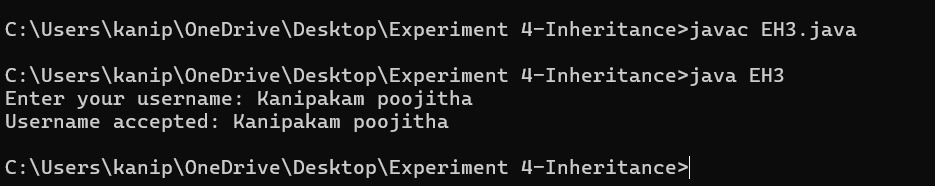
System.out.println("Error: " + e.getMessage());

}

}

}

**OUTPUT:**



**16D:** **exception handling 4**

**CODE:**

class FileTooLargeException extends Exception {

public FileTooLargeException(String msg) {

super(msg);

}

}

public class EH4 {

public static void main(String[] args) {

int fileSizeInMB = 25; // Example file size

try {

if (fileSizeInMB > 20) {

throw new FileTooLargeException("File size exceeds 20MB limit!");

} else {

System.out.println("File uploaded successfully.");

}

} catch (FileTooLargeException e) {

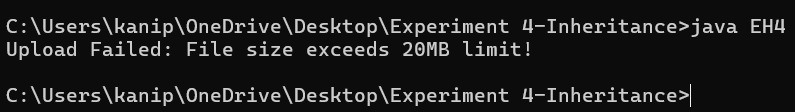
System.out.println("Upload Failed: " + e.getMessage());

}

}

}

**OUTPUT:**



**17A:** **file handling 1**

**CODE:**

import java.io.\*;

public class FH1 {

public static void main(String[] args) {

try {

BufferedReader br = new BufferedReader(new FileReader("source.txt"));

String line;

int words = 0, lines = 0, chars = 0;

while ((line = br.readLine()) != null) {

lines++;

chars += line.length();

words += line.split("\\s+").length;

}

br.close();

System.out.println("Lines: " + lines);

System.out.println("Words: " + words);

System.out.println("Characters: " + chars);

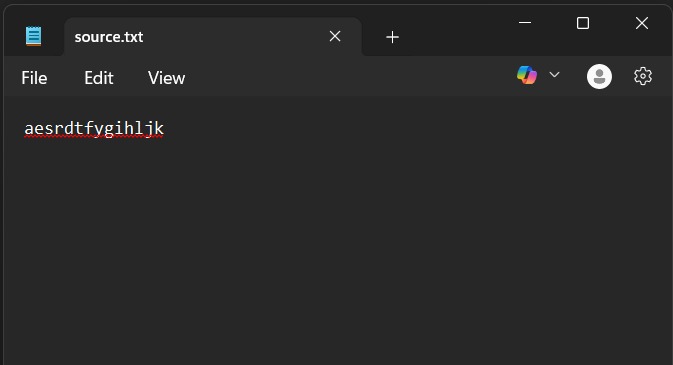
} catch (IOException e) {

e.printStackTrace();

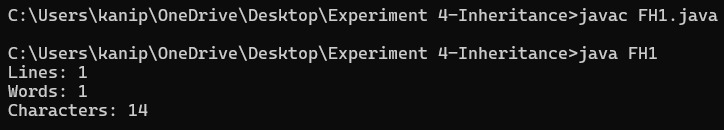
}

}

}



**OUTPUT:**



**17B:** **file handling 2**

**CODE:**

import java.io.\*;

import java.util.Scanner;

public class FH2 {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("Enter word to search: ");

String word = sc.nextLine();

int count = 0;

try {

BufferedReader br = new BufferedReader(new FileReader("source.txt"));

String line;

while ((line = br.readLine()) != null) {

for (String w : line.split("\\s+")) {

if (w.equalsIgnoreCase(word)) {

count++;

}

}

}

br.close();

System.out.println("The word '" + word + "' appears " + count + " times.");

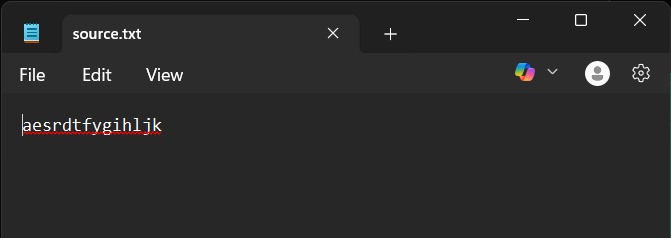
} catch (IOException e) {

e.printStackTrace();

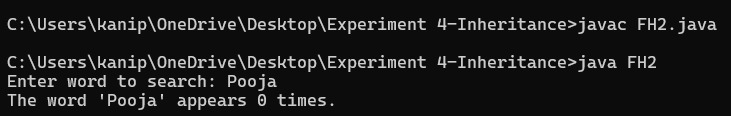
}

}

}



**OUTPUT:**



**17C:** **file handling 3**

**CODE:**

import java.io.\*;

public class FH3 {

public static void main(String[] args) {

try {

FileReader fr = new FileReader("plain.txt");

FileWriter fw = new FileWriter("encrypted.txt");

int ch;

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

fw.write(ch + 1);

}

fr.close();

fw.close();

System.out.println("File encrypted!");

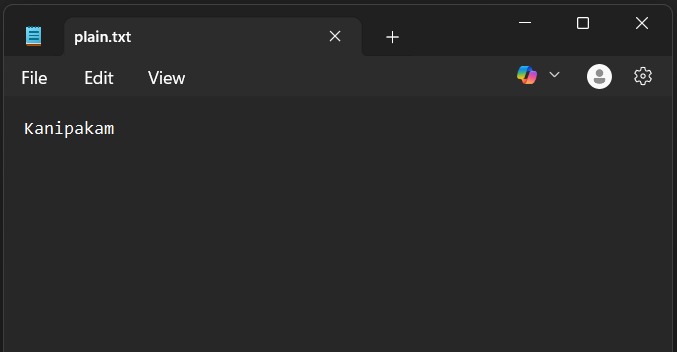
} catch (IOException e) {

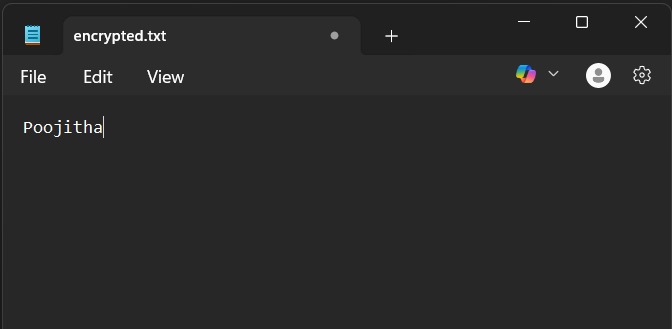
e.printStackTrace();

}

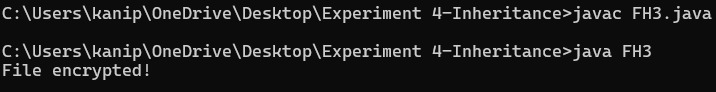
}

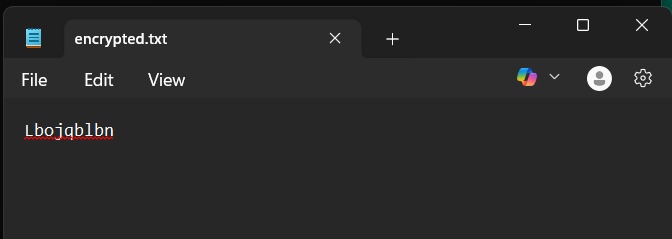
}





**OUTPUT:**





**17D:** **file handling 4**

**CODE:**

import java.io.\*;

public class FH4{

public static void main(String[] args) {

String topStudent = "";

double highestAvg = 0;

try (BufferedReader br = new BufferedReader(new FileReader("marks.txt"))) {

String line;

while ((line = br.readLine()) != null) {

String[] data = line.split(",");

String name = data[0];

int total = 0;

for (int i = 1; i < data.length; i++) {

total += Integer.parseInt(data[i]);

}

double avg = total / (double)(data.length - 1);

System.out.println(name + "'s Average: " + avg);

if (avg > highestAvg) {

highestAvg = avg;

topStudent = name;

}

}

System.out.println("Topper is: " + topStudent + " with average " + highestAvg);

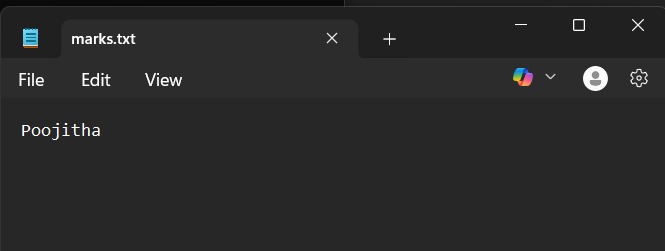
} catch (IOException e) {

e.printStackTrace();

}

}

}



**OUTPUT:**

