

LAB RECORD

23CSE111- Object Oriented Programming

***Submitted by***

CH.SC.U4CSE24117-G.Teja Prathap Varma

# BACHELOR OF TECHNOLOGY

## IN

COMPUTER SCIENCE AND ENGINEERING

AMRITA VISHWA VIDYAPEETHAM AMRITA SCHOOL OF COMPUTING

CHENNAI

March - 2025



**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.U4CSE24117-G.Teja Prathap Varma*** 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

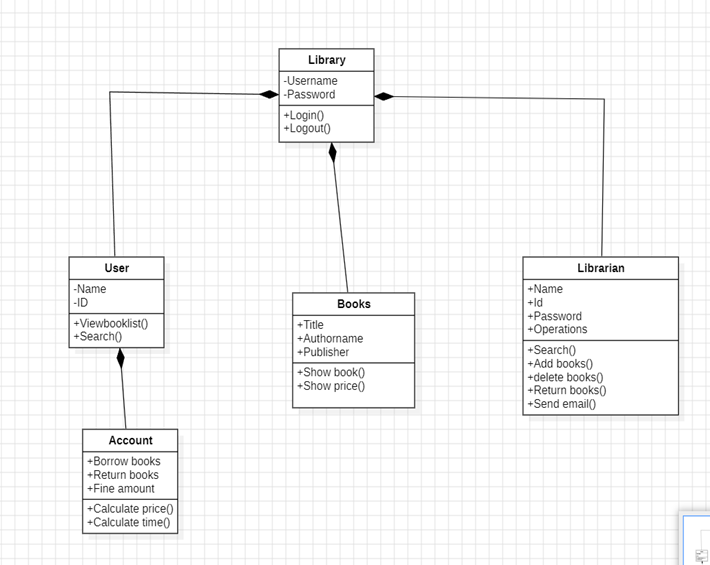
Internal Examiner 1 Internal Examiner 2

|  |  |  |  |
| --- | --- | --- | --- |
| **S.NO** | **TITLE** | **PAGE.NO** | |
| **UML DIAGRAM** | | |  |
| 1. | **TITLE OF UML DIAGRAM -1** | |  |
|  | 1.a) **CLASS Diagram** |  | |
|  | 1.b) **USE CASE DIAGRAM** |  | |
|  | 1.c) **state diagram** |  | |
|  | 1.d) **sequence diagram** |  | |
|  | 1.e) object diagram |  | |
| 2. | **TITLE OF UML DIAGRAM -2** | |  |
|  | 2.a) **CLASS Diagram** |  | |
|  | 2.b) **USE CASE DIAGRAM** |  | |
|  | 2.c) **state diagram** |  | |
|  | 2.d) **sequence diagram** |  | |
|  | 2.e) object diagram |  | |
|  |  |  | |
| 3. | **BASIC JAVA PROGRAMS** | |  |
|  | 3.a) chat bot |  | |
|  | 3.b) contact manger |  | |
|  | 3.c) digital lock |  | |
|  | 3.d) ) expense tracker |  | |
|  | 3.e) maze generator |  | |
|  | 3.f) password generator |  | |
|  | 3.g) SimpleEncryption |  | |
|  | 3.h)) Stopwatch |  | |
|  | 3.i) ) TicTacToe |  | |
|  | 3.j) Swap Two Numbers |  | |
|  | **INHERITANCE** |  | |

|  |  |  |
| --- | --- | --- |
| 4. | **SINGLE INHERITANCE PROGRAMS** |  |
|  | 4.a) Program-1 |  |
|  | 4.b) Program-2 |  |
| 5. | **MULTILEVEL INHERITANCE PROGRAMS** |  |
|  | 5.a) Program-1 |  |
|  | 5.b) Program-2 |  |
| 6. | **HIERARCHICAL INHERITANCE PROGRAMS** |  |
|  | 6.a) Program-1 |  |
|  | 6.b) Program-2 |  |
| 7. | **HYBRID INHERITANCE PROGRAMS** |  |
|  | 7.a) Program-1 |  |
|  | 7.b) Program-2 |  |
|  | **POLYMORPHISM** |  |
| 8. | **CONSTRUCTOR PROGRAMS** |  |
|  | 8.a) Program-1 |  |
| 9. | **CONSTRUCTOR OVERLOADING PROGRAMS** |  |
|  | 9.a) Program-1 |  |
| 10. | **METHOD OVERLOADING PROGRAMS** |  |
|  | 10.a) Program-1 |  |
|  | 10.b) Program-2 |  |
| 11. | **METHOD OVERRIDING PROGRAMS** |  |
|  | 11.a) Program-1 |  |
|  | 11.b) Program-2 |  |
|  | **ABSTRACTION** |  |
| 12. | **INTERFACE PROGRAMS** |  |
|  | 12.a) Program-1 |  |
|  | 12.b) Program-2 |  |
|  | 12.c) Program-3 |  |
|  | 12.d) Program-4 |  |
| 13. | **ABSTRACT CLASS PROGRAMS** |  |
|  | 13.a) Program-1 |  |
|  | 13.b) Program-2 |  |
|  | 13.c) Program-3 |  |
|  | 13.d) Program-4 |  |
|  | **ENCAPSULATION** |  |
| 14. | **ENCAPSULATION PROGRAMS** |  |
|  | 14.a) Program-1 |  |
|  | 14.b) Program-2 |  |
|  | 14.c) Program-3 |  |
|  | 14.d) Program-4 |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
| 15. | **PACKAGES PROGRAMS** |  |
|  | 15.a) Program-1 |  |
|  | 15.b) Program-2 |  |
|  | 15.c) Program-3 |  |
|  | 15.d) Program-4 |  |
| 16. | **EXCEPTION HANDLING PROGRAMS** |  |
|  | 16.a) Program-1 |  |
|  | 16.b) Program-2 |  |
|  | 16.c) Program-3 |  |
|  | 16.d) Program-4 |  |
| 17. | **FILE HANDLING PROGRAMS** |  |
|  | 17.a) Program-1 |  |
|  | 17.b) Program-2 |  |
|  | 17.c) Program-3 |  |
|  | 17.d) Program-4 |  |

1.L**IBRARY MANAGEMENT SYSTEM**

* 1. **CLASS Diagram:**

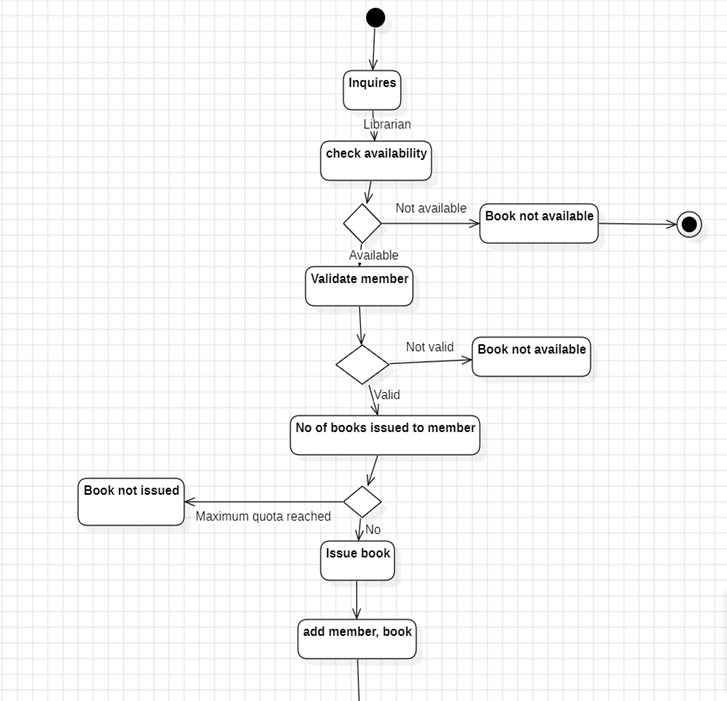


**1b) USE CASE DIAGRAM**

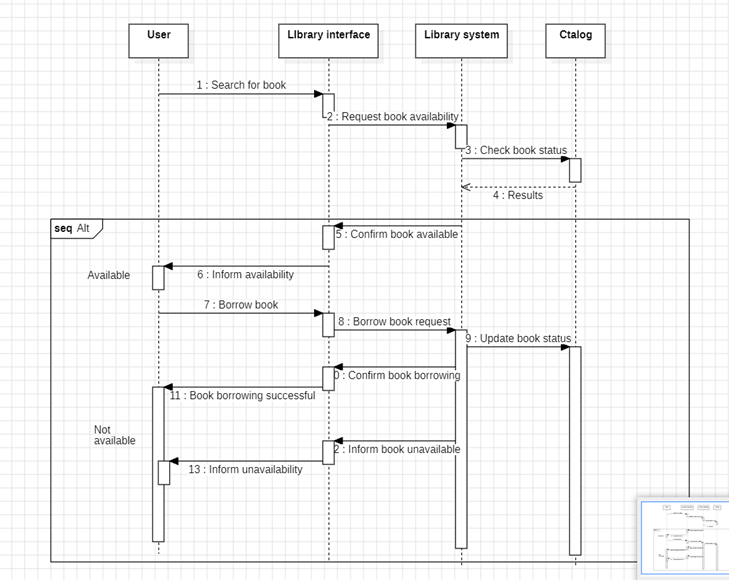
A diagram of a diagram

AI-generated content may be incorrect.

**1c) state diagram**



**1d) sequence diagram**



1e) object diagram

A diagram of a library

AI-generated content may be incorrect.

# **Bank management system**

2a) class diagram

A screenshot of a computer

AI-generated content may be incorrect.

2b) use case diagram

A diagram of a person with circles and lines

AI-generated content may be incorrect.

2c) state diagram

A diagram of a flowchart

AI-generated content may be incorrect.

2d) SEQUENCE DIAGRAM

A diagram of a bank server

AI-generated content may be incorrect.

2e) COLLABRATION DIAGRAM

A diagram of a bank card

AI-generated content may be incorrect.

BASIC JAVA

3A) chat bot

import java.util.Scanner;

public class Chatbot {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

while (true) { // Outer loop for conversation

System.out.print("You: ");

String input = sc.nextLine();

if (input.equalsIgnoreCase("exit")) {

System.out.println("Chatbot: Bye! Have a great day!");

break;

}

for (int i = 1; i <= 1; i++) { // Inner loop for extra responses

System.out.println("Chatbot: Hmm, interesting! Tell me more.");

}

}

}

}

OUTPUT:

You: Hello

Chatbot: Hmm, interesting! Tell me more.

You: How are you?

Chatbot: Hmm, interesting! Tell me more.

You: exit

Chatbot: Bye! Have a great day!

3B) ContactManager

import java.util.HashMap;

import java.util.Scanner;

public class ContactManager {

public static void main(String[] args) {

HashMap<String, String> contacts = new HashMap<>();

Scanner sc = new Scanner(System.in);

while (true) { // Outer loop: Handles the menu and user choices

System.out.println("1. Add Contact 2. View Contacts 3. Exit");

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

int choice = sc.nextInt();

sc.nextLine(); // Consume newline

switch (choice) {

case 1:

while (true) { // Inner loop: Allows multiple entries at once

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

String name = sc.nextLine();

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

String phone = sc.nextLine();

contacts.put(name, phone);

System.out.print("Do you want to add another contact? (yes/no): ");

String response = sc.nextLine();

if (!response.equalsIgnoreCase("yes")) break; // Exit inner loop

}

break;

case 2:

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

if (contacts.isEmpty()) {

System.out.println("No contacts available.");

} else {

for (String key : contacts.keySet()) { // Iterating through contacts using a loop

System.out.println(key + ": " + contacts.get(key));

}

}

break;

case 3:

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

return; // Exit the program

default:

System.out.println("Invalid choice! Please try again.");

}

}

}

}

OUTPUT:

1. Add Contact 2. View Contacts 3. Exit

Choose an option: 1

Enter name: Alice

Enter phone number: 1234567890

Do you want to add another contact? (yes/no): yes

Enter name: Bob

Enter phone number: 9876543210

Do you want to add another contact? (yes/no): no

1. Add Contact 2. View Contacts 3. Exit

Choose an option: 2

Contacts:

Alice: 1234567890

Bob: 9876543210

1. Add Contact 2. View Contacts 3. Exit

Choose an option: 3

Exiting...

3C) DigitalClock

import java.text.SimpleDateFormat;

import java.util.Date;

public class DigitalClock {

public static void main(String[] args) {

int hoursToDisplay = 1; // Display clock for 1 hour

for (int i = 0; i < hoursToDisplay \* 3600; i++) { // Outer loop for 1 hour

Date date = new Date();

SimpleDateFormat formatter = new SimpleDateFormat("HH:mm:ss");

System.out.print("\r" + formatter.format(date));

// Inner loop for 1-second intervals

try {

Thread.sleep(1000);

} catch (InterruptedException e) {

e.printStackTrace();

}

}

}

}

}

OUTPUT:

14:30:01

14:30:02

14:30:03

...

3D) ExpenseTracker

import java.util.Random;

public class MazeGenerator {

public static void main(String[] args) {

int rows = 10, cols = 10;

char[][] maze = new char[rows][cols];

Random random = new Random();

for (int i = 0; i < rows; i++) { // Outer loop for rows

for (int j = 0; j < cols; j++) { // Inner loop for columns

maze[i][j] = random.nextBoolean() ? '#' : ' ';

}

}

maze[0][0] = 'S'; // Start

maze[rows - 1][cols - 1] = 'E'; // End

for (char[] row : maze) { // Loop to print the maze

for (char cell : row) {

System.out.print(cell);

}

System.out.println();

}

}

}

OUTPUT:

S # # #

# # ## #

# # # #

### ## #

# ## #

## ##

# # #

## ## #

# ###

# # E

3E) MazeGenerator

import java.util.Random;

public class MazeGenerator {

public static void main(String[] args) {

int rows = 10, cols = 10;

char[][] maze = new char[rows][cols];

Random random = new Random();

for (int i = 0; i < rows; i++) {

for (int j = 0; j < cols; j++) {

maze[i][j] = random.nextBoolean() ? '#' : ' ';

}

}

maze[0][0] = 'S'; // Start

maze[rows - 1][cols - 1] = 'E'; // End

for (char[] row : maze) {

for (char cell : row) {

System.out.print(cell);

}

System.out.println();

}

}

OUTPUT:

S # # #

# # ## #

# # # #

### ## #

# ## #

## ##

# # #

## ## #

# ###

# # E

3F) PasswordGenerator

import java.util.Random;

import java.util.Scanner;

public class PasswordGenerator {

public static void main(String[] args) {

String chars = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789!@#$%^&\*()";

Scanner sc = new Scanner(System.in);

System.out.print("How many passwords do you want to generate? ");

int numPasswords = sc.nextInt();

for (int p = 0; p < numPasswords; p++) { // Outer loop for multiple passwords

StringBuilder password = new StringBuilder();

Random random = new Random();

int length = 12; // Desired password length

for (int i = 0; i < length; i++) { // Inner loop for characters in each password

int index = random.nextInt(chars.length());

password.append(chars.charAt(index));

}

System.out.println("Generated Password " + (p + 1) + ": " + password);

}

}

}

OUTPUT:

How many passwords do you want to generate? 3

Generated Password 1: G7#kd8@Pq9!X

Generated Password 2: Xy&8Lp$23AoM

Generated Password 3: B!m3^KqL7@9Z

3G) SimpleEncryption

import java.util.Scanner;

public class SimpleEncryption {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("Enter text to encrypt: ");

String text = sc.nextLine();

System.out.print("Enter shift value: ");

int shift = sc.nextInt();

StringBuilder encrypted = new StringBuilder();

for (char c : text.toCharArray()) {

if (Character.isLetter(c)) {

char base = Character.isLowerCase(c) ? 'a' : 'A';

c = (char) ((c - base + shift) % 26 + base);

}

encrypted.append(c);

}

System.out.println("Encrypted Text: " + encrypted);

}

}

OUTPUT:

Enter text to encrypt: Hello World

Enter shift value: 3

Encrypted Text: Khoor Zruog

3H) Stopwatch

import java.util.Scanner;

public class Stopwatch {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println("Press Enter to start the stopwatch.");

sc.nextLine();

long startTime = System.currentTimeMillis();

System.out.println("Press Enter to stop the stopwatch.");

sc.nextLine();

long endTime = System.currentTimeMillis();

System.out.println("Elapsed Time: " + (endTime - startTime) / 1000.0 + " seconds.");

}

}

OUTPUT:

Press Enter to start the stopwatch.

(You press Enter)

Press Enter to stop the stopwatch.

(You press Enter after some time)

Elapsed Time: 5.32 seconds.

3I) TicTacToe

import java.util.Scanner;

public class TicTacToe {

static char[][] board = { {'1', '2', '3'}, {'4', '5', '6'}, {'7', '8', '9'} };

static char currentPlayer = 'X';

public static void main(String[] args) {

playGame();

}

static void playGame() {

while (true) {

printBoard();

playerMove();

if (checkWin()) {

printBoard();

System.out.println("Player " + currentPlayer + " wins!");

break;

}

if (isBoardFull()) {

printBoard();

System.out.println("The game is a tie!");

break;

}

currentPlayer = (currentPlayer == 'X') ? 'O' : 'X';

}

}

static void printBoard() {

for (char[] row : board) {

for (char cell : row) {

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

}

System.out.println();

}

}

static void playerMove() {

Scanner sc = new Scanner(System.in);

System.out.print("Player " + currentPlayer + ", enter your move (1-9): ");

int move = sc.nextInt();

int row = (move - 1) / 3;

int col = (move - 1) % 3;

if (board[row][col] != 'X' && board[row][col] != 'O') {

board[row][col] = currentPlayer;

} else {

System.out.println("Invalid move! Try again.");

playerMove();

}

}

static boolean checkWin() {

for (int i = 0; i < 3; i++) {

if ((board[i][0] == currentPlayer && board[i][1] == currentPlayer && board[i][2] == currentPlayer) ||

(board[0][i] == currentPlayer && board[1][i] == currentPlayer && board[2][i] == currentPlayer)) {

return true;

}

}

return (board[0][0] == currentPlayer && board[1][1] == currentPlayer && board[2][2] == currentPlayer) ||

(board[0][2] == currentPlayer && board[1][1] == currentPlayer && board[2][0] == currentPlayer);

}

static boolean isBoardFull() {

for (char[] row : board) {

for (char cell : row) {

if (cell != 'X' && cell != 'O') {

return false;

}

}

}

return true;

}

}

OUTPUT:

1 2 3

4 5 6

7 8 9

Player X, enter your move (1-9): 5

1 2 3

4 X 6

7 8 9

Player O, enter your move (1-9): 1

O 2 3

4 X 6

7 8 9

... (game continues)

Player X wins

3J) UnitConverter

import java.util.Scanner;

public class UnitConverter {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

while (true) { // Outer loop for multiple conversions

System.out.print("Enter distance in kilometers (or type '-1' to exit): ");

double kilometers = sc.nextDouble();

if (kilometers == -1) break;

for (int i = 1; i <= 1; i++) { // Inner loop for repeated confirmation

double miles = kilometers \* 0.621371;

System.out.println(kilometers + " kilometers is equal to " + miles + " miles.");

}

}

}

}

OUTPUT: Enter distance in kilometers (or type '-1' to exit): 5

5.0 kilometers is equal to 3.106855 miles.

Enter distance in kilometers (or type '-1' to exit): 10

10.0 kilometers is equal to 6.21371 miles.

Enter distance in kilometers (or type '-1' to exit): -1

|  |  |  |
| --- | --- | --- |
| EXP: 4,5,6,7 | INHERITANCE | OOPS |

4a)

code: class Animal {

void sound() {

System.out.println("Animals make different sounds.");

}

}

class Dog extends Animal {

void bark() {

System.out.println("Dog barks: Woof Woof!");

}

}

public class SingleInheritance {

public static void main(String[] args) {

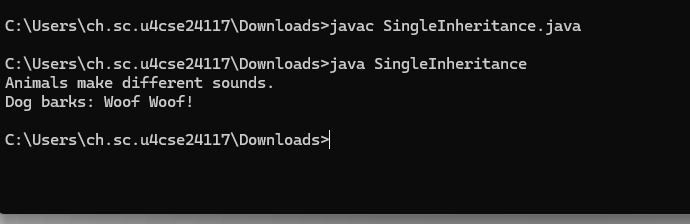
Dog myDog = new Dog();

myDog.sound(); // Calling method from parent class

myDog.bark(); // Calling method from child class

}

}

Output: 

4b)

Code: class Vehicle {

Vehicle() {

System.out.println("Vehicle is created.");

}

}

class Car extends Vehicle {

Car() {

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

}

}

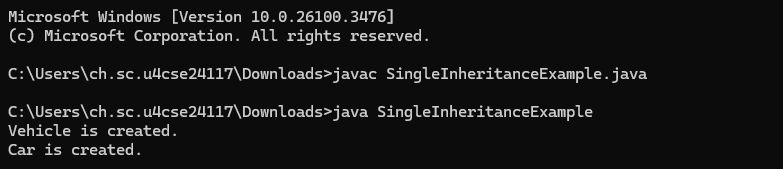
public class SingleInheritanceExample {

public static void main(String[] args) {

Car myCar = new Car(); // Calls both parent and child constructors

}

}

Output:5a)

Code: public class I3{

public static void main(String[]args){

Car c1=new BMW();

BMW b1=new BMW();

Car c2=new Car();

BMW t3=new BMWTruck();

BMWTruck t4=new BMWTruck();

t3.stt();

t3.drive();

t4.stt();

t4.drive();

t4.offroad();

c1.stt();

b1.stt();

b1.drive();

}

}

class Car{

void stt(){

System.out.println("Car Starts");

}

}

class BMW extends Car{

void drive(){

System.out.println("BMW Is Driven");

}

}

class BMWTruck extends BMW{

void offroad(){

System.out.println("Offroad is supported with ease");

}}

Output:

5b)

Code: public class I4{

public static void main(String[]args){

rect r1=new rect();

r1.area(2,3);

square s1=new square();

s1.area(6);

}

}

class Shape{

void area(){

System.out.println("This can Calculate area of rectangle and square");

}

}

class rect extends Shape{

void area(int a,int b){

System.out.println("Area Of rectangle is: "+(a\*b));

}

}

class square extends rect{

void area(int a){

System.out.println("Area Of Square is: "+(a\*a));

}

}

Output:

\

6a))AIM: Create a Java program where Shape has display(), Circle adds area(radius), and Rectangle adds area(length, breadth); call all methods.

CODE: class Shape {

void display() {

System.out.println("This is a shape");

}

}

class Circle extends Shape { void area(double radius) {

System.out.println("Circle Area: " + (Math.PI \* radius \* radius));

}

}

class Rectangle extends Shape { void area(double length, double breadth)

{ System.out.println("Rectangle Area: " + (length \* breadth));

}

}

public class Main {

public static void main(String[] args) {

Circle c = new Circle();

c.display();

c.area(5);

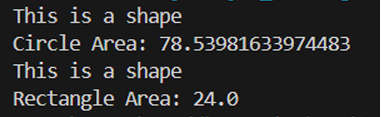
Rectangle r = new Rectangle();

r.display();

r.area(4, 6);

}

}

OUTPUT:

6b)AIM: Write a Java program where Vehicle has start(), Car adds drive(), and Bike adds ride(); call all methods using Car and Bike.

CODE:

class Vehicle {

void start() { System.out.println("Vehicle is starting...");

}

}

class Car extends Vehicle {

void drive() {

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

}

}

class Bike extends Vehicle {

void ride() { System.out.println("Bike is riding...");

}

}

public class Main { public static void main(String[] args)

{ Car car = new Car();

car.start();

car.drive();

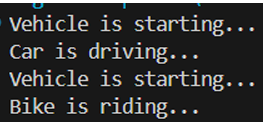
Bike bike = new Bike();

bike.start();

bike.ride();

}

Output:



7a) Person class:

class Person {

String name;

int age;

public Person(String name, int age){

this.name = name;

this.age = age;

}

public void displayPerson() {

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

}

}

class Employee extends Person {

double salary;

public Employee(String name, int age, double salary) {

super(name, age);

this.salary = salary;

}

public void displayEmployee() {

displayPerson();

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

}

}

class Student extends Person {

String grade;

public Student(String name, int age, String grade) {

super(name, age);

this.grade = grade;

}

public void displayStudent() {

displayPerson();

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

}

}

class Intern extends Employee {

String internshipField;

public Intern(String name, int age, double salary, String internshipField) {

super(name, age, salary);

this.internshipField = internshipField;

}

public void displayIntern() {

displayEmployee();

System.out.println("Internship Field: " + internshipField);

}

}

public class HY1 {

public static void main(String[] args) {

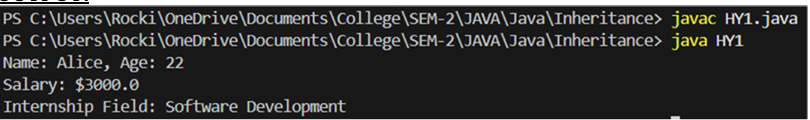
Intern intern = new Intern("Alice", 22, 3000, "Software Development");

intern.displayIntern();

}

}

Output:



7b) Grandparent class:

class Grandparent {

void grandparentMethod() {

System.out.println("This is a grandparent");

}

}

class Parent1 extends Grandparent {

void parent1Method() {

System.out.println("This is from Parent1");

}

}

class Parent2 extends Grandparent {

void parent2Method() {

System.out.println("This is from Parent2");

}

}

class Child extends Parent1 {

Parent2 p2 = new Parent2();

void childMethod() {

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

}

void useParent2Methods() {

p2.parent2Method();

}

}

public class HY2 {

public static void main(String[] args) {

Child obj = new Child();

obj.grandparentMethod();

obj.parent1Method();

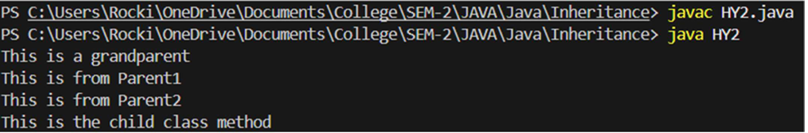
obj.useParent2Methods();

obj.childMethod();

}

}

Output:



POLYMORPHISM

8. CONSTRUCTOR

8.a) Constructor 1: class Student { String name; int rollNumber;

public Student(String name, int rollNumber) {   
 this.name = name;   
 this.rollNumber = rollNumber;   
}   
  
public void display() {   
 System.out.println("Student Name: " + name);   
 System.out.println("Roll Number: " + rollNumber);   
}   
  
public static void main(String[] args) {   
 Student s1 = new Student("Alice", 101);   
 s1.display();   
}

}

OUTPUT:



8.b) Constructor 2: class Rectangle { double length, breadth;

// Constructor   
public Rectangle(double length, double breadth) {   
 this.length = length;   
 this.breadth = breadth;   
}   
  
// Method to calculate and return area   
public double getArea() {   
 return length \* breadth;   
}   
  
public static void main(String[] args) {   
 Rectangle rect = new Rectangle(5.0, 3.0);   
 System.out.println("Area of Rectangle: " + rect.getArea());   
}

}

OUTPUT:



9a) 9a) CODE:

class Student {

String name;

int age;

char grade;

Student(String name) {

this.name = name;

}

Student(String name, int age) {

this.name = name;

this.age = age;

}

Student(String name, int age, char grade) {

this.name = name;

this.age = age;

this.grade = grade;

}

void display() {

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

if (age != 0)

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

if (grade != '\u0000')

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

System.out.println();

}

public static void main(String[] args) {

Student s1 = new Student("Anjali");

Student s2 = new Student("Ravi", 19);

Student s3 = new Student("Sneha", 20, 'A');

s1.display();

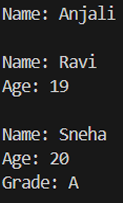
s2.display();

s3.display();

}

}

Screenshot:



10a)

CODE: class AreaCalculator {

double area(double radius) {

return Math.PI \* radius \* radius;

}

int area(int side) { return side \* side;

}

int area(int length, int breadth) { return length \* breadth;

}

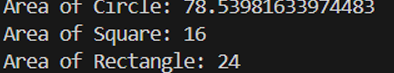
public static void main(String[] args) {

AreaCalculator ac = new AreaCalculator(); System.out.println("Area of Circle: " + ac.area(5.0)); System.out.println("Area of Square: " + ac.area(4)); System.out.println("Area of Rectangle: " + ac.area(4, 6));

}

}

Output:



10.b) Method overloading 2: class Display { public void show(int number) { System.out.println("Integer: " + number); }

public void show(String message) {   
 System.out.println("Message: " + message);   
}   
  
public static void main(String[] args) {   
 Display obj = new Display();   
 obj.show(42);   
 obj.show("Polymorphism");   
}

}

OUTPUT:



11a)

CODE:

class BankAccount {

String accountHolder;

double balance;

BankAccount(String name, double balance) {

this.accountHolder = name;

this.balance = balance;

}

void calculateInterest() {

System.out.println("Generic account: No specific interest calculation.");

}

}

class SavingsAccount extends BankAccount {

SavingsAccount(String name, double balance) {

super(name, balance);

}

@Override

void calculateInterest() {

double interest = balance \* 0.04;

System.out.println("Savings Account Interest for " + accountHolder + ": ₹"

+ interest);

}

}

class FixedDepositAccount extends BankAccount {

FixedDepositAccount(String name, double balance) {

super(name, balance);

}

@Override

void calculateInterest() {

double interest = balance \* 0.07;

System.out.println("Fixed Deposit Interest for " + accountHolder + ": ₹" +

interest);

}

}

public class BankTest {

public static void main(String[] args) {

BankAccount acc1 = new SavingsAccount("Sheshank", 10000);

BankAccount acc2 = new FixedDepositAccount("Sheshank", 20000);

acc1.calculateInterest();

acc2.calculateInterest();

}

}

Output: 

11.b) Method overriding 2: class Bank { public double getInterestRate() { return 5.0;

}

}

class SBI extends Bank { @Override public double getInterestRate() { return 6.5;

}

}

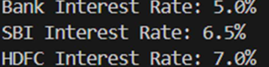
class HDFC extends Bank { @Override public double getInterestRate() { return 7.0; } }

public class Main { public static void main(String[] args) { Bank b = new Bank(); SBI sbi = new SBI(); HDFC hdfc = new HDFC();

System.out.println("Bank Interest Rate: " + b.getInterestRate() + "%");   
 System.out.println("SBI Interest Rate: " + sbi.getInterestRate() + "%");   
 System.out.println("HDFC Interest Rate: " + hdfc.getInterestRate() + "%");   
}

}

OUTPUT:



ABSTRACTION

12. INTERFACE

12a) Interface 1:

interface Printable {

void print();

}

class Document implements Printable {

public void print() {

System.out.println("Printing the document");

}

public static void main(String[] args) {

Document doc = new Document(); doc.print();

}

}

OUTPUT:



12b)AIM: Transport Booking System.

CODE:

package Abstraction;

interface Transport {

void bookRide(String source, String destination);

}

class Cab implements Transport {

public void bookRide(String source, String destination) {

System.out.println("Cab booked from " + source + " to " + destination + ".");

}

}

class Auto implements Transport {

public void bookRide(String source, String destination) {

System.out.println("Auto booked from " + source + " to " + destination + ".");

}

}

class Bike implements Transport {

public void bookRide(String source, String destination) {

System.out.println("Bike booked from " + source + " to " + destination + ".");

}

}

public class TransportBooking {

public static void main(String[] args) {

Transport t1 = new Cab();

Transport t2 = new Auto();

Transport t3 = new Bike();

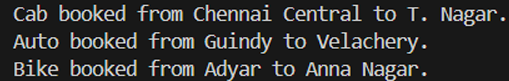
t1.bookRide("Chennai Central", "T. Nagar");

t2.bookRide("Guindy", "Velachery");

t3.bookRide("Adyar", "Anna Nagar");

}

}

OUTPUT: 

12c)AIM: Online Payment System.

CODE:

package Abstraction;

interface OnlinePayment {

void pay(double amount);

}

class Paytm implements OnlinePayment {

public void pay(double amount) {

System.out.println("Paid ₹" + amount + " using Paytm.");

}

}

class GooglePay implements OnlinePayment {

public void pay(double amount) {

System.out.println("Paid ₹" + amount + " using Google Pay.");

}

}

public class PaymentApp {

public static void main(String[] args) {

OnlinePayment p1 = new Paytm();

OnlinePayment p2 = new GooglePay();

p1.pay(999);

p2.pay(499);

}

}

OUTPUT: 

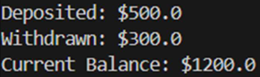
12.d) Interface 4: interface BankAccount { void deposit(double amount); void withdraw(double amount); double getBalance(); }

class SavingsAccount implements BankAccount { private double balance;

public SavingsAccount(double initialBalance) {   
 this.balance = initialBalance;   
}   
  
public void deposit(double amount) {   
 balance += amount;   
 System.out.println("Deposited: $" + amount);   
}   
  
public void withdraw(double amount) {   
 if (amount <= balance) {   
 balance -= amount;   
 System.out.println("Withdrawn: $" + amount);   
 } else {   
 System.out.println("Insufficient balance!");   
 }   
}   
  
public double getBalance() {   
 return balance;   
}   
  
public static void main(String[] args) {   
 SavingsAccount account = new SavingsAccount(1000);   
 account.deposit(500);   
 account.withdraw(300);   
 System.out.println("Current Balance: $" + account.getBalance());   
}

}

Output:



13. ABSTRACT CLASS

13.a) Abstract class 1: abstract class Animal { abstract void makeSound(); }

class Dog extends Animal { public void makeSound() { System.out.println("Dog barks"); } }

class Cat extends Animal { public void makeSound() { System.out.println("Cat meows"); } }

public class AnimalTest { public static void main(String[] args) { Animal dog = new Dog(); Animal cat = new Cat();

dog.makeSound();   
 cat.makeSound();   
}

}

OUTPUT:



13.b) Abstract class 2: abstract class Shape { abstract double calculateArea(); }

class Circle extends Shape { double radius;

public Circle(double radius) {   
 this.radius = radius;   
}   
  
public double calculateArea() {   
 return Math.PI \* radius \* radius;   
}

}

class Rectangle extends Shape { double length, breadth;

public Rectangle(double length, double breadth) {   
 this.length = length;   
 this.breadth = breadth;   
}   
  
public double calculateArea() {   
 return length \* breadth;   
}

}

public class ShapeTest { public static void main(String[] args) { Shape circle = new Circle(5); Shape rectangle = new Rectangle(4, 6);

System.out.println("Circle Area: " + circle.calculateArea());   
 System.out.println("Rectangle Area: " + rectangle.calculateArea());   
}

}

OUTPUT:



13c)CODE:

package Abstraction;

abstract class Order {

String product;

Order(String product) {

this.product = product;

}

abstract void calculateShipping();

void confirmOrder() {

System.out.println("Order placed for: " + product);

}

}

class LocalOrder extends Order {

LocalOrder(String product) {

super(product);

}

@Override

void calculateShipping() {

System.out.println("Shipping charges: ₹50 (Local delivery)");

}

}

class InternationalOrder extends Order {

InternationalOrder(String product) {

super(product);

}

@Override

void calculateShipping() {

System.out.println("Shipping charges: ₹500 (International

delivery)");

}

}

public class OrderSystem {

public static void main(String[] args) {

Order o1 = new LocalOrder("Book");

Order o2 = new InternationalOrder("Laptop");

o1.confirmOrder();

o1.calculateShipping();

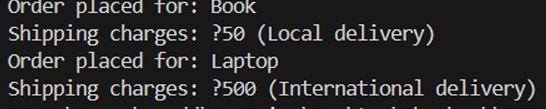
o2.confirmOrder();

o2.calculateShipping();

}

}

OUTPUT:



13d)

CODE:

abstract class ElectricityBill {

String customerName;

int units;

ElectricityBill(String customerName, int units) {

this.customerName = customerName;

this.units = units;

}

abstract double calculateBill();

void printBill() {

System.out.println(customerName + "'s Bill Amount: ₹" +

calculateBill());

}

}

class DomesticConnection extends ElectricityBill {

DomesticConnection(String customerName, int units) {

super(customerName, units);

}

@Override

double calculateBill() {

return units \* 3.5;

}

}

class CommercialConnection extends ElectricityBill {

CommercialConnection(String customerName, int units) {

super(customerName, units);

}

double calculateBill() {

return units \* 7.0;

}

}

public class ElectricityMain {

public static void main(String[] args) {

ElectricityBill d = new DomesticConnection("Sheshank", 100);

ElectricityBill c = new CommercialConnection("Amrita Labs",

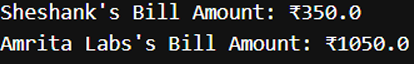
150);

d.printBill();

c.printBill();

}

}

OUTPUT: 

ENCAPSULATION

14. ENCAPSULATION PROGRAMS

14a)

CODE:

package Encapsulation;

class BankAccount {

private String accountHolder;

private double balance;

public void setAccountHolder(String name) {

this.accountHolder = name;

}

public String getAccountHolder() {

return accountHolder;

}

public double getBalance() {

return balance;

}

public void deposit(double amount) {

if (amount > 0)

balance += amount;

}

public void withdraw(double amount) {

if (amount > 0 && amount <= balance)

balance -= amount;

else

System.out.println("Insufficient funds or invalid amount.");

}

}

public class BankTest {

public static void main(String[] args) {

BankAccount acc = new BankAccount();

acc.setAccountHolder("Sheshank");

acc.deposit(5000);

acc.withdraw(1500);

System.out.println("Account Holder: " + acc.getAccountHolder());

System.out.println("Balance: ₹" + acc.getBalance());

}

}

OUTPUT:



14.b) Encapsulation 2: class Student { private String name; private int rollNumber; private int marks;

public Student(String name, int rollNumber, int marks) {   
 this.name = name;   
 this.rollNumber = rollNumber;   
 setMarks(marks);   
}   
  
public void setMarks(int marks) {   
 if (marks >= 0 && marks <= 100) {   
 this.marks = marks;   
 } else {   
 System.out.println("Invalid marks! Setting marks to 0.");   
 this.marks = 0;   
 }   
}   
  
public int getMarks() {   
 return marks;   
}   
  
public void display() {   
 System.out.println("Name: " + name + ", Roll Number: " + rollNumber + ", Marks: "

* marks); }

public static void main(String[] args) { Student student = new Student("Alice", 101, 95); student.display(); }

}

OUTPUT: 

14c)AIM: Movie Ticket Booking.

CODE:

package Encapsulation;

class MovieTicket {

private String movieName;

private int numberOfTickets;

private double ticketPrice;

public void setMovieName(String name) {

this.movieName = name;

}

public void setNumberOfTickets(int count) {

if (count > 0) {

this.numberOfTickets = count;

} else {

System.out.println("Ticket count must be positive.");

}

}

public void setTicketPrice(double price) {

if (price > 0) {

this.ticketPrice = price;

}

}

public double getTotalAmount() {

return numberOfTickets \* ticketPrice;

}

public void printBill() {

System.out.println("Movie: " + movieName);

System.out.println("Tickets: " + numberOfTickets);

System.out.println("Total Amount: ₹" + getTotalAmount());

}

}

public class MovieTest {

public static void main(String[] args) {

MovieTicket ticket = new MovieTicket();

ticket.setMovieName("RRR");

ticket.setNumberOfTickets(3);

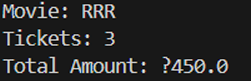
ticket.setTicketPrice(150);

ticket.printBill();

}

}

OUTPUT:



14d)Encapsulation 4:

class Car {

private String brand;

private String model;

private double price;

public Car(String brand, String model, double price) {

this.brand = brand;

this.model = model;

setPrice(price);

}

public void setPrice(double price) {

if (price > 0) {

this.price = price;

} else {

System.out.println("Price cannot be zero or negative. Setting to default

$10,000.");

this.price = 10000;

}

}

public double getPrice() {

return price;

}

public void display() {

System.out.println("Car Brand: " + brand + ", Model: " + model + ", Price: $" +

price);

}

public static void main(String[] args) {

Car car = new Car("Tesla", "Model S", 79999);

car.display();

}

}

OUTPUT: 

PACKAGES

15.User-Defined Packages

15.a) User-defined 1: CODE-1: package package1;

public class Calculator { public int add(int a, int b) { return a + b; }

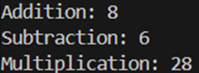
public int subtract(int a, int b) {   
 return a - b;   
}   
  
public int multiply(int a, int b) {   
 return a \* b;   
}

}

CODE-2: import package1.Calculator;

public class PackageTest { public static void main(String[] args) { Calculator calc = new Calculator(); System.out.println("Addition: " + calc.add(5, 3)); System.out.println("Subtraction: " + calc.subtract(8, 2)); System.out.println("Multiplication: " + calc.multiply(4, 7)); } }

OUTPUT:



15.b) User-defined 2: CODE-1: package package2;

public class EmployeeDetails { private String name; private int id; private double salary;

public EmployeeDetails(String name, int id, double salary) {   
 this.name = name;   
 this.id = id;   
 this.salary = salary;   
}   
  
public void display() {   
 System.out.println("Employee: " + name + ", ID: " + id + ", Salary: $" + salary);   
}

}

CODE-2: import package2.EmployeeDetails;

public class EmployeeTest { public static void main(String[] args) { EmployeeDetails emp = new EmployeeDetails("John Doe", 101, 6000); emp.display(); } }

OUTPUT:



15c)AIM: Password Generator

CODE:

import java.time.Duration;

import java.time.Instant;

import java.util.Random;

import java.util.Scanner;

public class PasswordGenerator {

public static String generatePassword(int length) {

String chars =

"ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvw

xyz0123456789@#$%&\*!";

Random random = new Random();

StringBuilder password = new StringBuilder();

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

int index = random.nextInt(chars.length());

password.append(chars.charAt(index));

}

return password.toString();

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter password length: ");

int length = scanner.nextInt();

Instant start = Instant.now();

String password = generatePassword(length);

Instant end = Instant.now();

Duration timeElapsed = Duration.between(start, end);

System.out.println("Generated Password: " + password);

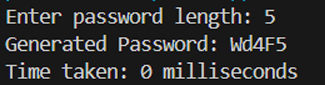
System.out.println("Time taken: " + timeElapsed.toMillis() + "

milliseconds");

scanner.close();

}

}

OUTPUT: 

15d)

CODE:

import java.time.LocalDate;

import java.time.LocalTime;

import java.util.Random;

import java.util.Date;

import java.io.FileWriter;

import java.io.IOException;

public class Builtin {

public static void main(String[] args) {

LocalDate today = LocalDate.now();

LocalTime now = LocalTime.now();

System.out.println("Today's Date: " + today);

System.out.println("Current Time: " + now);

Random random = new Random();

int randNumber = random.nextInt(100);

System.out.println("Random Number: " + randNumber);

// File

try {

FileWriter writer = new FileWriter("output.txt");

writer.write("Date: " + today + "\n");

writer.write("Time: " + now + "\n");

writer.write("Random Number: " + randNumber + "\n");

writer.close();

System.out.println("Data successfully written to file.");

} catch (IOException e) {

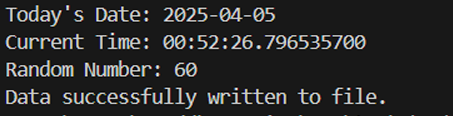
e.printStackTrace();

}

}

}

OUTPUT:



16.a) Array Exception :

Code :

public class ArrayException{ public static void main(String[] args) { int[] numbers = {1, 2, 3};

try {   
 System.out.println(numbers[5]); // Accessing invalid index   
 } catch (ArrayIndexOutOfBoundsException e) {   
 System.out.println("Error: Index out of bounds!");   
 }   
  
 System.out.println("Program continues...");   
}

}

Output :



16.b) Divide By Zero Exception :

Code :

public class DivideByZero {

public static void main(String[] args)

{

try { int result = 10 / 0; // This will cause ArithmeticException System.out.println("Result: " + result);

}

catch (ArithmeticException e)

{ System.out.println("Error: Cannot divide by zero!");

}

System.out.println("Program continues...");

}

}

Output: 

16c)

CODE:

import java.io.BufferedReader;

import java.io.FileReader;

import java.io.IOException;

public class WritingFile {

public static void main(String[] args) {

try {

FileReader fileReader = new

FileReader("C:\\Users\\prodd\\OneDrive\\Desktop\\Sheshank\\OOPS\\File

Handling\\config.txt");

BufferedReader bufferedReader = new BufferedReader(fileReader);

String line;

System.out.println("Reading configuration:");

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

System.out.println(line);

}

bufferedReader.close();

} catch (IOException e) {

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

}

}

}

OUTPUT: 

16d)

CODE:

import java.util.\*;

public class user {

public static void main(String[] args) {

Scanner n = new Scanner(System.in);

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

String nam = n.nextLine();

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

String pass = n.nextLine();

Login p = new Login();

p.signup(nam, pass);

}

}

class SyntaxException extends Exception {

SyntaxException(String message) {

super(message);

}

}

class Login {

void signup(String nam, String pass) {

System.out.println("Thanks for signing up! Please log in.");

try {

Scanner nn = new Scanner(System.in);

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

String nm = nn.nextLine();

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

String pss = nn.nextLine();

if (!nm.equals(nam) || !pss.equals(pass)) {

throw new SyntaxException("Wrong username or

password");

}

System.out.println("Login successful!");

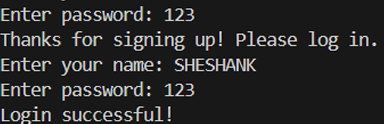
} catch (SyntaxException e) {

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

}

}

}

OUTPUT:

File Handling Programs

17.a) Replace a word in a Text File :

Code :

import java.io.*; import java.nio.file.*;

public class ReplaceWord { public static void main(String[] args) { String filePath = "test1.txt"; String oldWord = "Java"; String newWord = "Python";

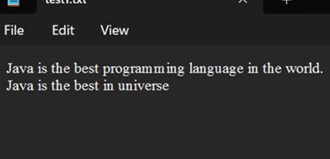
try {   
 String content = new String(Files.readAllBytes(Paths.get(filePath)));   
 content = content.replaceAll(oldWord, newWord);   
 Files.write(Paths.get(filePath), content.getBytes());   
 System.out.println("Replaced all occurrences of '" + oldWord + "' with '" +

newWord + "'."); } catch (IOException e) { e.printStackTrace(); } } }

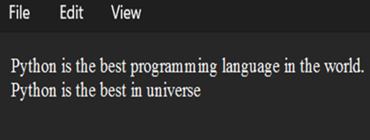
Output :



File before :



File after :



17.b) Read a Text File :

Code :

import java.io.File; import java.io.FileNotFoundException; import java.util.Scanner;

public class ReadFile { public static void main(String[] args) { try (Scanner scanner = new Scanner(new File("test.txt"))) { while (scanner.hasNextLine()) { System.out.println(scanner.nextLine()); } } catch (FileNotFoundException e) { e.printStackTrace(); } } }

Output :



17c)

CODE:

import java.io.BufferedReader;

import java.io.FileReader;

import java.io.IOException;

public class WritingFile {

public static void main(String[] args) {

try {

FileReader fileReader = new

FileReader("C:\\Users\\prodd\\OneDrive\\Desktop\\Sheshank\\OOPS\\File

Handling\\config.txt");

BufferedReader bufferedReader = new BufferedReader(fileReader);

String line;

System.out.println("Reading configuration:");

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

System.out.println(line);

}

bufferedReader.close();

} catch (IOException e) {

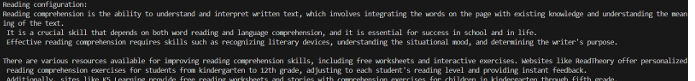
System.out.println("Error reading file: " + e.getMessage());

}

}

}

OUTPUT:



17d)

CODE:

import java.io.\*;

public class FileHandlingExample {

public static void main(String[] args) {

String fileName = "example.txt";

try (FileWriter writer = new FileWriter(fileName)) {

writer.write("Hello, this is a test file!\n");

writer.write("This is line 2.");

System.out.println("Successfully wrote to the file.");

} catch (IOException e) {

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

}

try (BufferedReader reader = new BufferedReader(new

FileReader(fileName))) {

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

String line;

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

System.out.println(line);

}

} catch (IOException e) {

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

}

try (FileWriter writer = new FileWriter(fileName, true)) {

writer.write("\nThis is an appended line.");

System.out.println("\nSuccessfully appended to the file.");

} catch (IOException e) {

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

}

try (FileInputStream fis = new FileInputStream(fileName)) {

System.out.println("\nReading using FileInputStream:");

int content;

while ((content = fis.read()) != -1) {

System.out.print((char) content);

}

} catch (IOException e) {

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

}

}

}

OUTPUT:

