LAB RECORD

23CSE111- Object Oriented Programming

Submitted by

CH.SC.U4CSE24143 –S.Lahari Sai

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.U4CSE24143 – S.LAHARI SAI*** 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

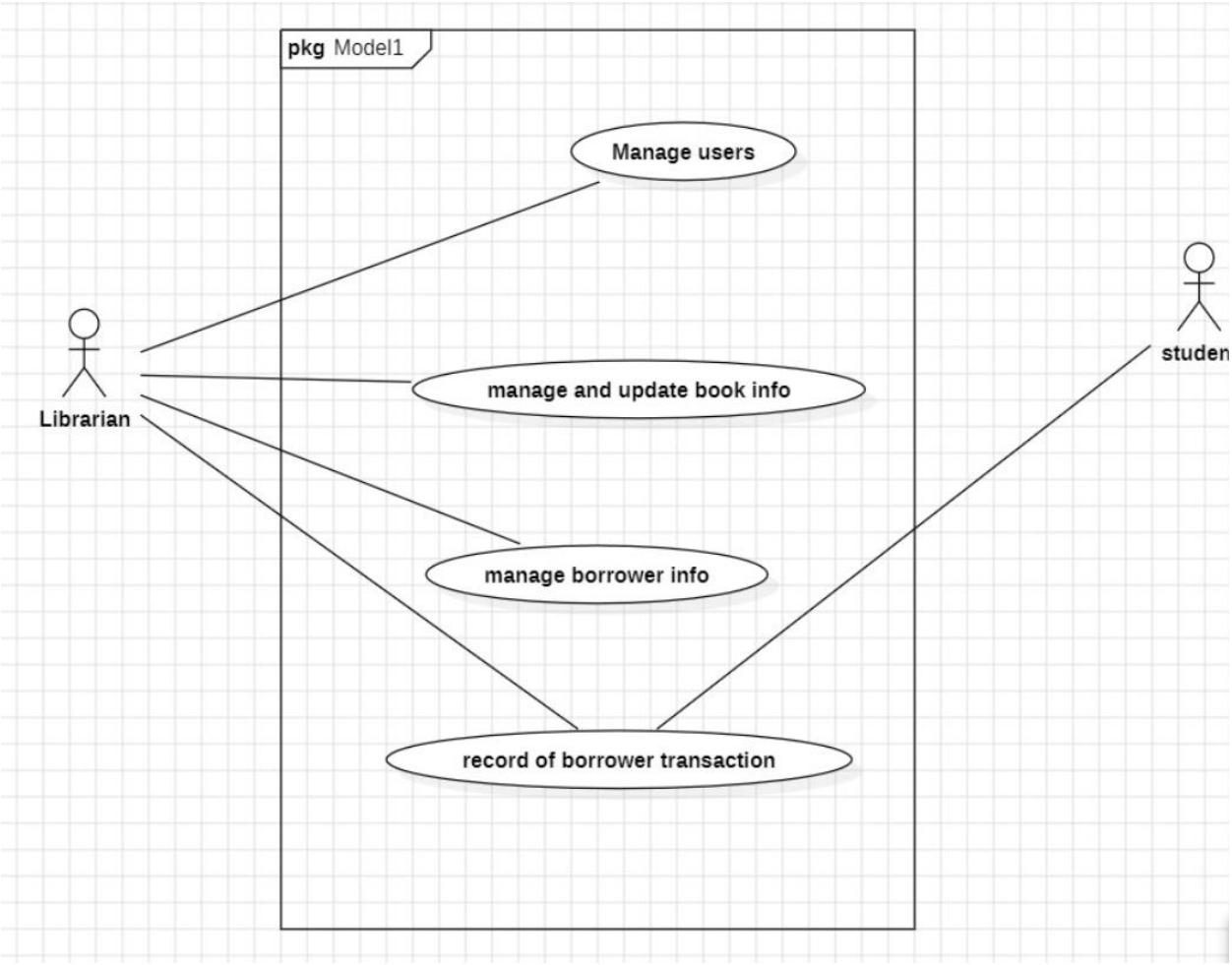
Internal Examiner 1 Internal Examiner 2

INDEX

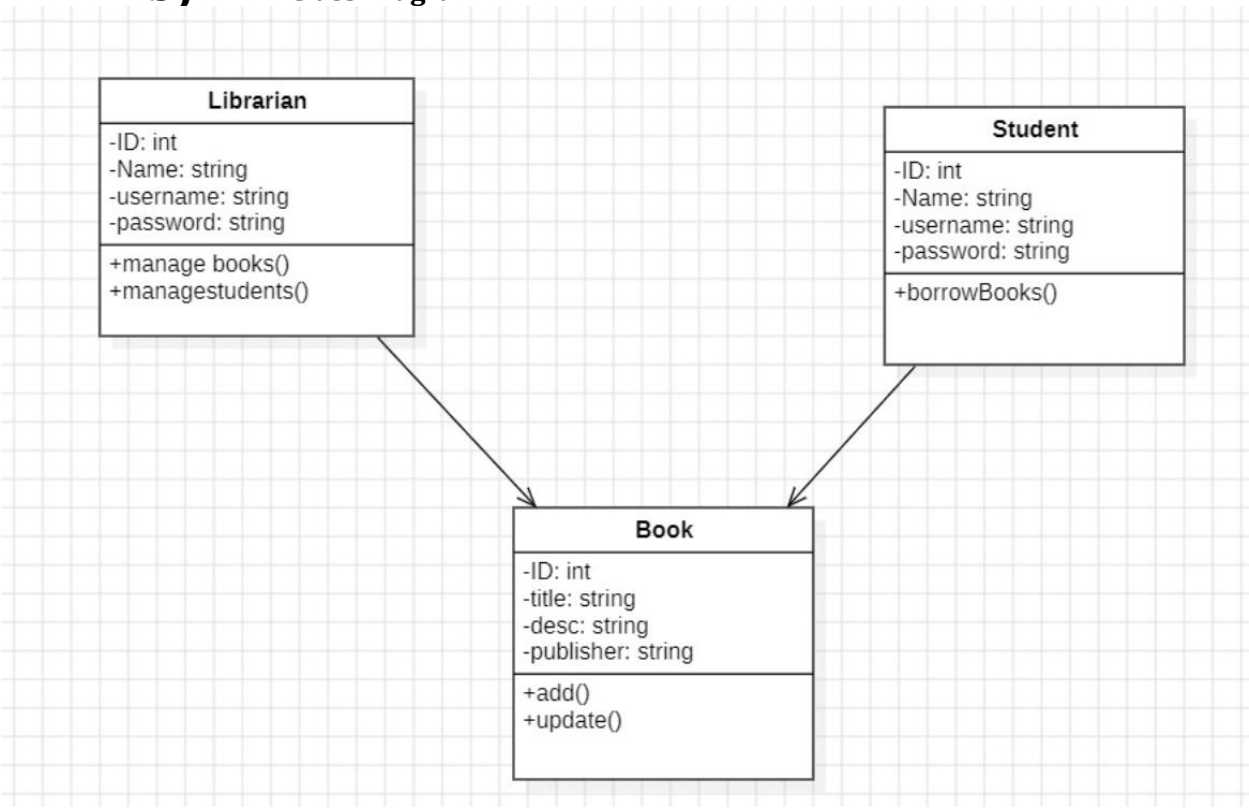
UML DIAGRAMS

1. REFERENCE CENTER

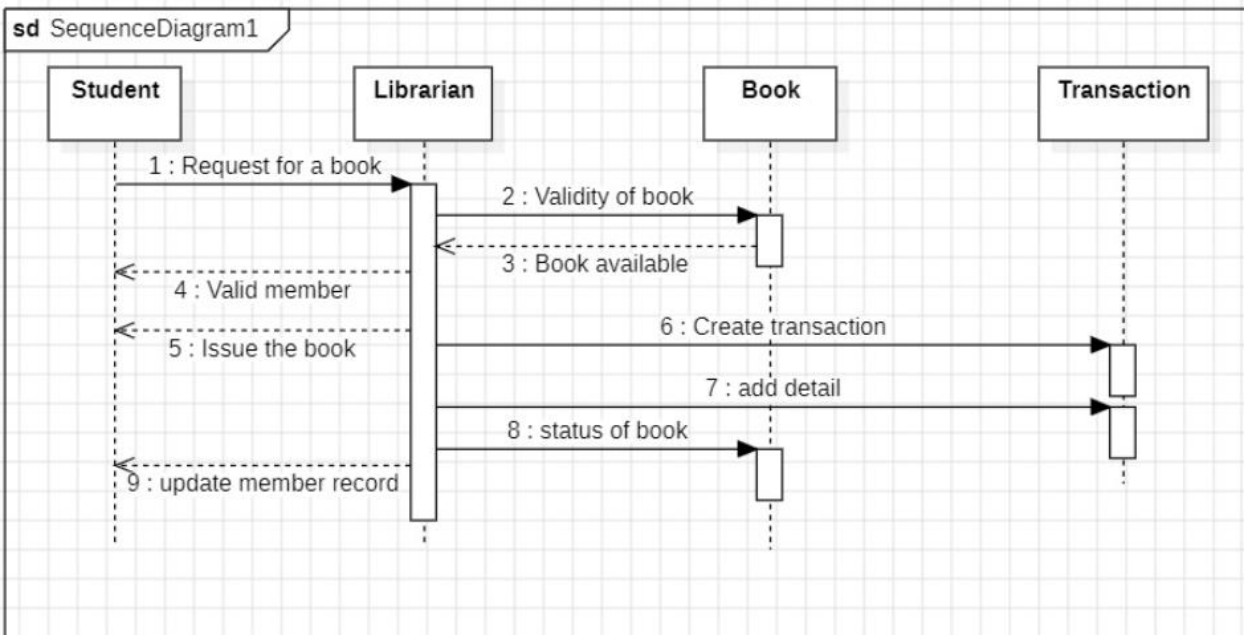
1.a) Use Case Diagram:



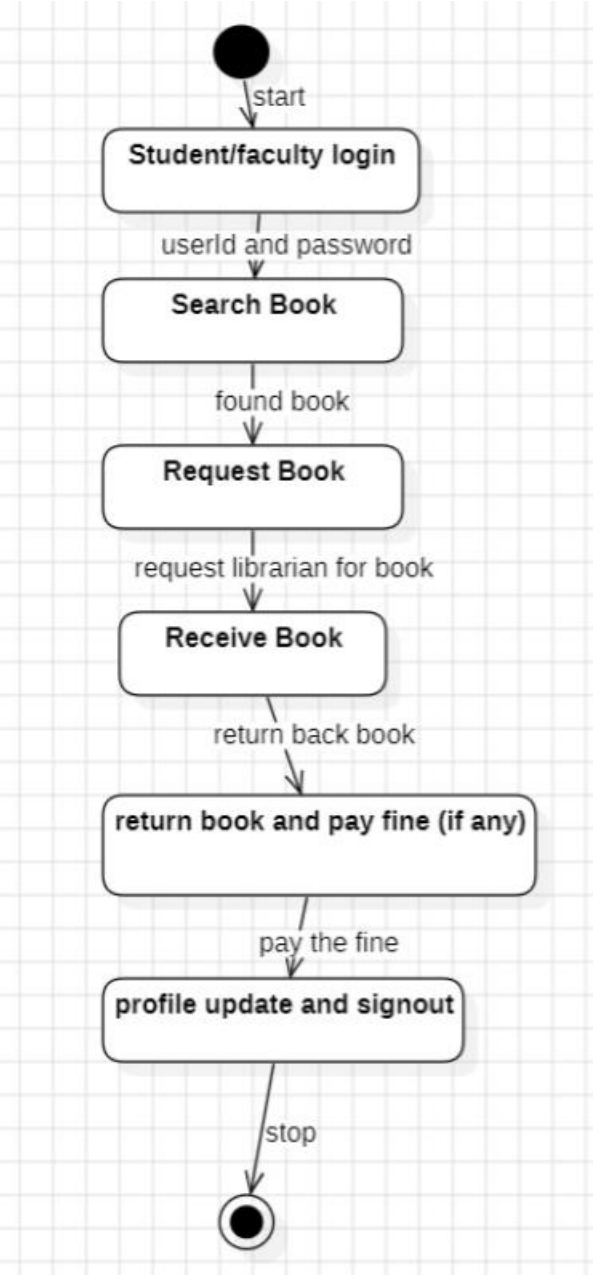
1.b) Class Diagram:



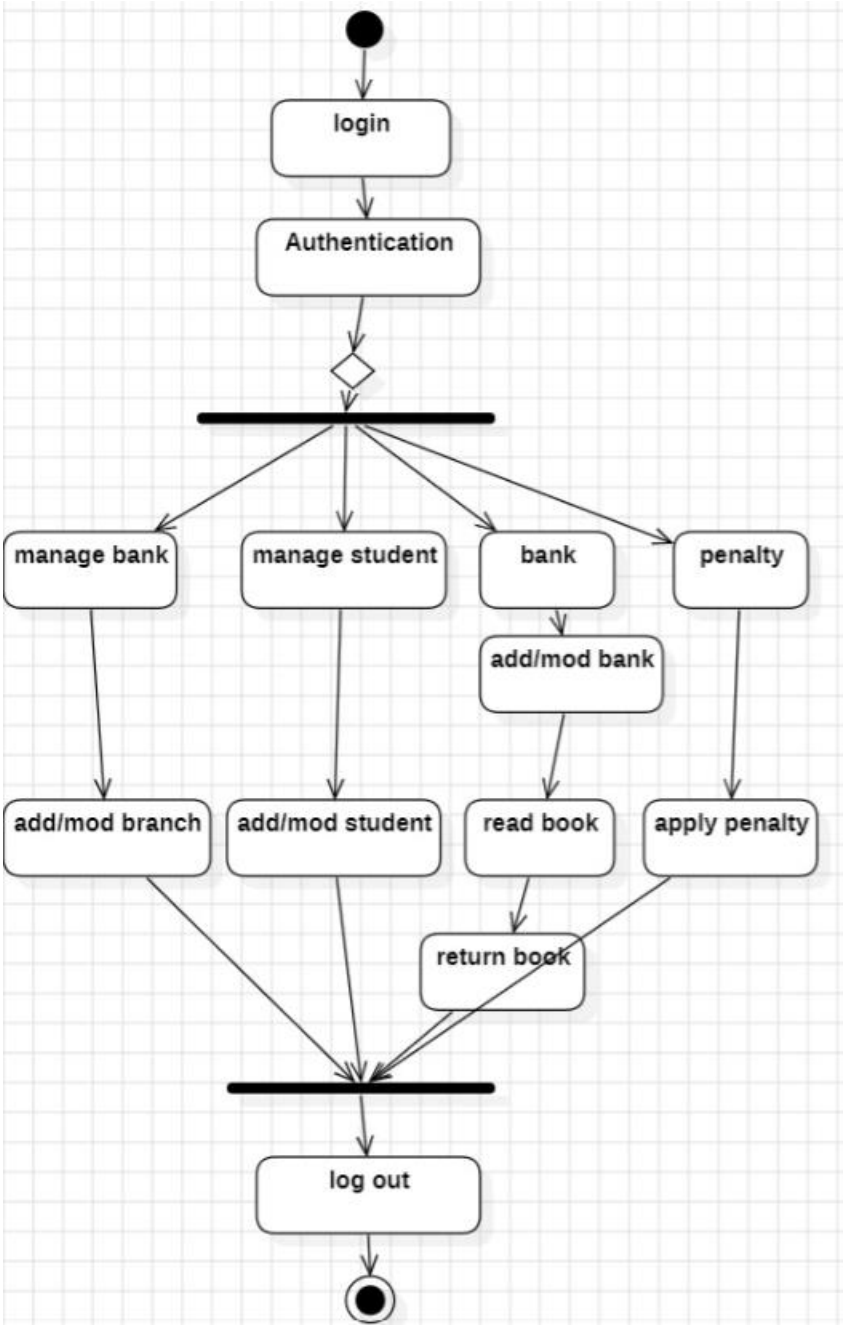
1.c) Sequence Diagram:



1.d) Activity Diagram:

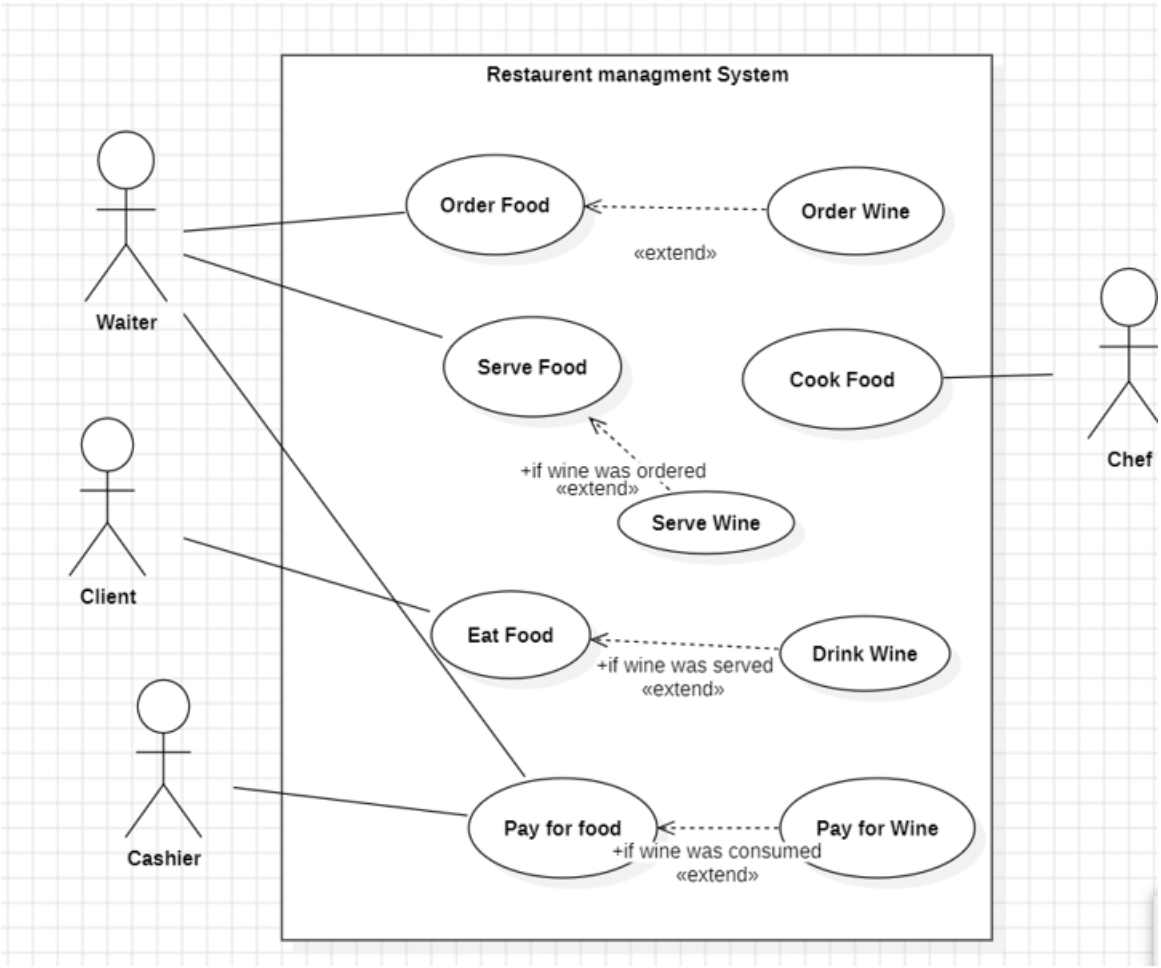


1.e) State-Activity Diagram:

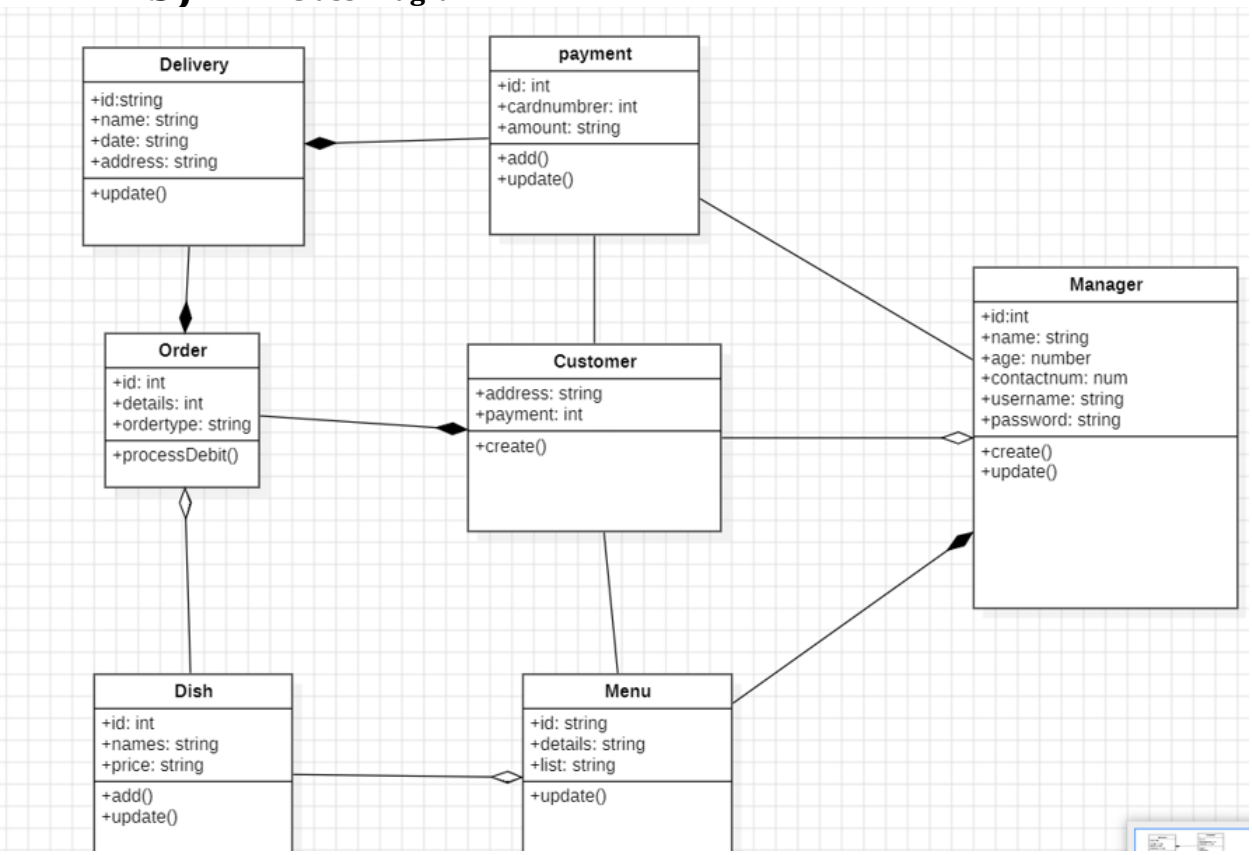


2. CAFE

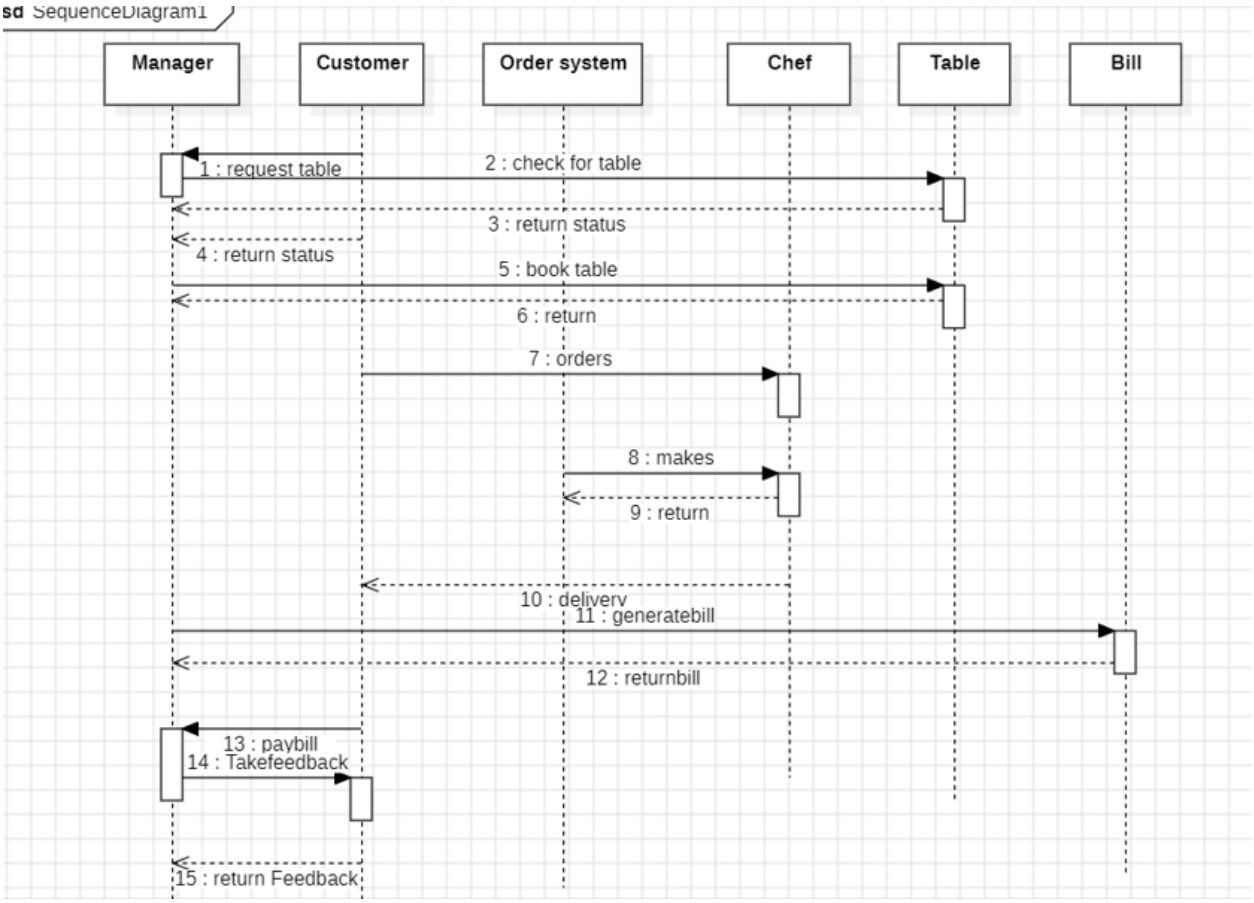
2.a) Use Case Diagram:



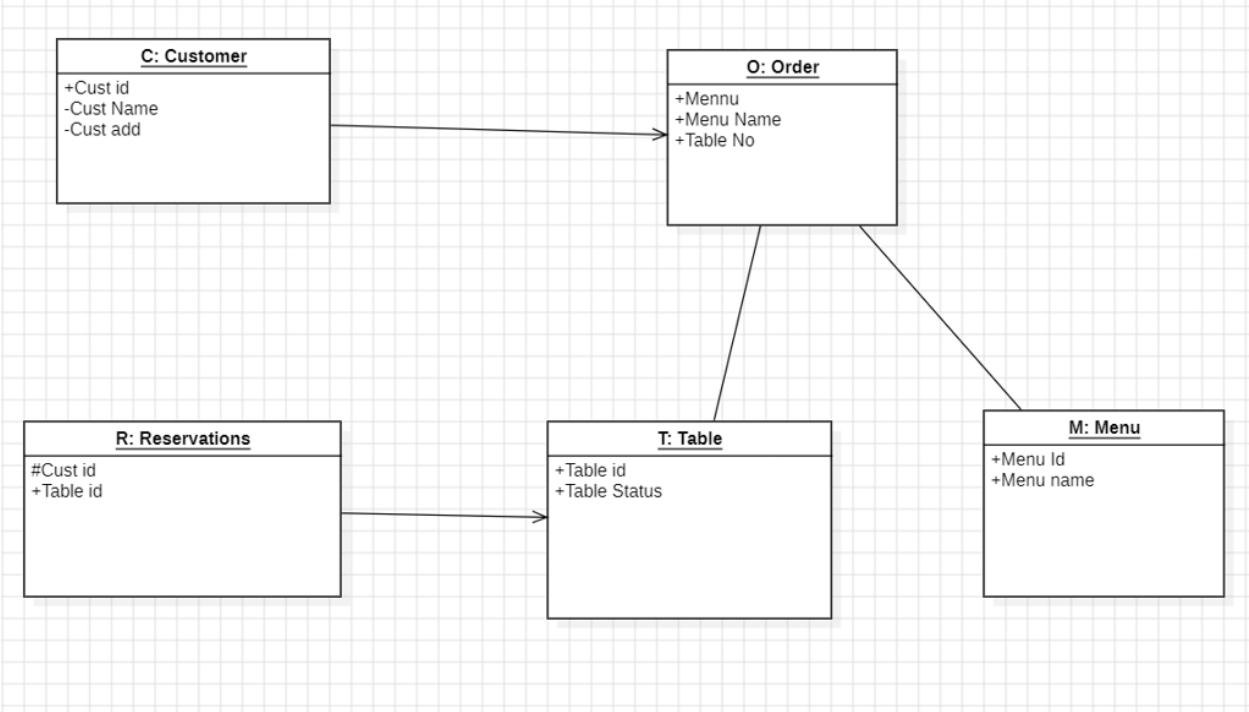
2.b) Class Diagram:



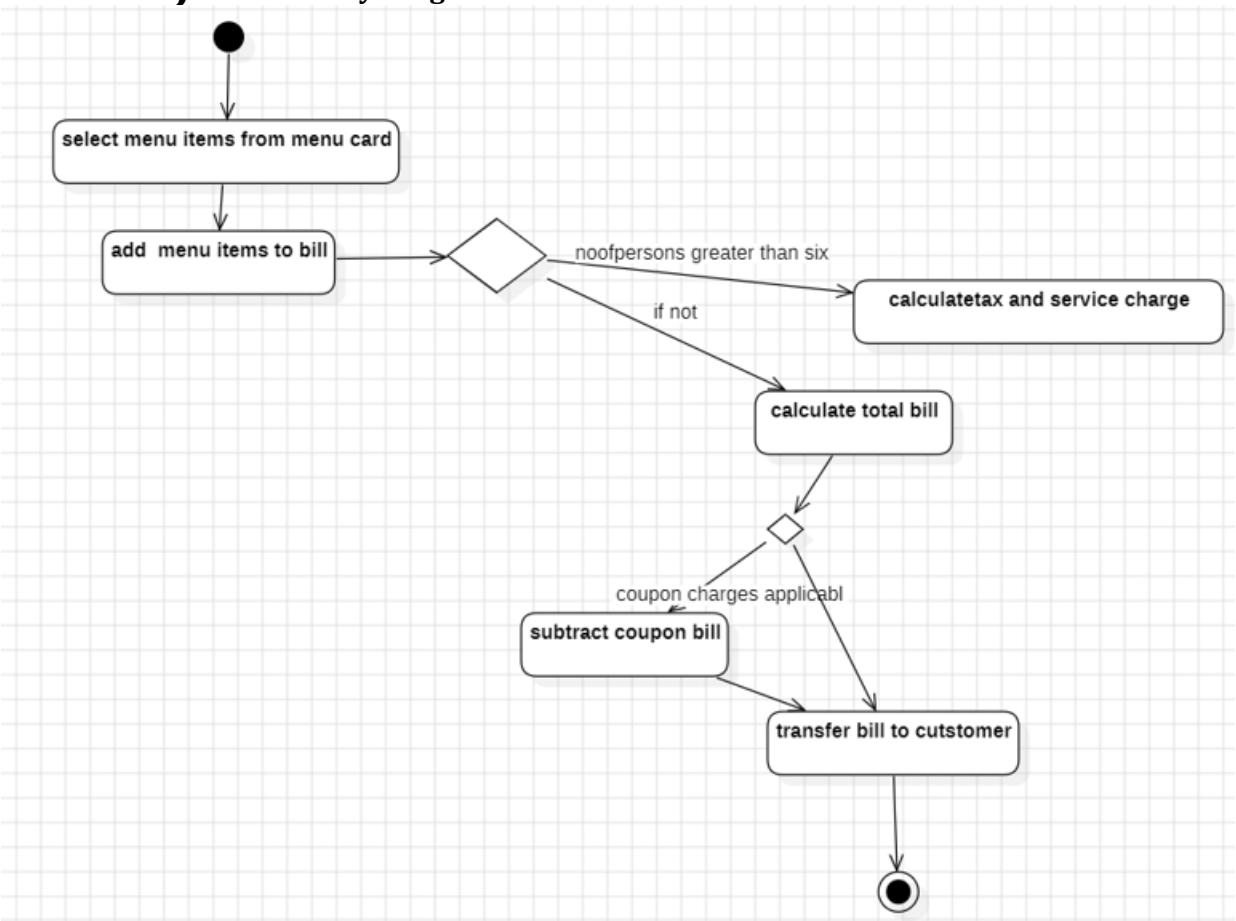
2.c) Sequence Diagram:



2.d) Object Diagram:



2.e) State-Activity Diagram:



3. Basic Java Programs

3.a)Amount:

Code:

```
class Account {
    double balance;

    public Account(double balance) {
        this.balance = balance;
    }

    public void displayBalance() {
        System.out.println("Balance: " + balance);
    }
}
class SavingsAccount extends Account {
    public SavingsAccount(double balance) {
        super(balance);
    }


    public void displayBalance() {
        System.out.println("Savings Account Balance: " + balance);
    }
}

public class Amount {
    public static void main(String[] args) {
        SavingsAccount account = new SavingsAccount(5000);
        account.displayBalance();
    }
}
```

Output:

```
C:\Users\HP\Documents\java programs>javac Amount.java

C:\Users\HP\Documents\java programs>java Amount.java
Savings Account Balance: 5000.0
```

3.b)Transport:

Code:

```
class Vehicle {
    public void type() {
        System.out.println("This is a vehicle");
    }
}
class Car extends Vehicle {
    public void type() {
        System.out.println("This is a car");
    }
}

public class Transport {
    public static void main(String[] args) {
        Car car = new Car();
        car.type();
    }
}
```

Output:

```
C:\Users\HP\Documents\java programs>javac Transport.java

C:\Users\HP\Documents\java programs>java Transport.java
This is a car
```

3.c) Carnivorous:

Code:.

```
class Animal {
        private String name;
        public Animal(String name) {
           this.name = name;
    }
        public void makeSound() {
           System.out.println(name + " makes a sound.");
    }
        public String getName() {
           return name;
    }
}
class Dog extends Animal {
        public Dog(String name) {
           super(name);
    }
        public void makeSound() {
           System.out.println(getName() + " barks.");
    }
}

public class Carnivorous {
        public static void main(String[] args) {
           Dog dog = new Dog("Buddy");
           dog.makeSound();
           Animal animal = new Animal("Generic Animal");
           animal.makeSound();
    }
}
```

Output:

```
C:\Users\HP\Documents\java programs>javac Carnivorous.java

C:\Users\HP\Documents\java programs>java Carnivorous.java
Buddy barks.
Generic Animal makes a sound.
```

3.d)Work

Code:

```
class Employee {
    String name;

    public Employee(String name) {
        this.name = name;
    }

    public void work() {
        System.out.println(name + " is working");
    }
}

class Manager extends Employee {
    public Manager(String name) {
        super(name);
    }

       public void work() {
        System.out.println(name + " is managing the team");
    }
}

public class Work {
    public static void main(String[] args) {
        Manager manager = new Manager("Alice");
        manager.work();
    }
}
```

Output:

```
C:\Users\HP\Documents\java programs>javac Work.java

C:\Users\HP\Documents\java programs>java Work.java
Alice is managing the team
```


Output:

3.e) Study

Code:

```
class Book {
    String title;

    public Book(String title) {
        this.title = title;
    }

    public void display() {
        System.out.println("Book Title: " + title);
    }
}


class EBook extends Book {
    public EBook(String title) {
        super(title);
    }

       public void display() {
        System.out.println("E-Book Title: " + title);
    }
}

public class Study {
    public static void main(String[] args) {
        EBook ebook = new EBook("Java Programming");
        ebook.display();
    }
}
```

Output:

```
C:\Users\HP\Documents\java programs>javac Study.java

C:\Users\HP\Documents\java programs>java Study.java
E-Book Title: Java Programming
```

3.f) Account

Code:

```
class BankAccount {
    double balance;

    public BankAccount(double balance) {
        this.balance = balance;
    }

    public void deposit(double amount) {
        balance += amount;
    }

    public void displayBalance() {
        System.out.println("Bank Account Balance: " + balance);
    }
}


class CheckingAccount extends BankAccount {
    public CheckingAccount(double balance) {
        super(balance);
    }

    public void displayBalance() {
        System.out.println("Checking Account Balance: " + balance);
    }
}

public class Account {
    public static void main(String[] args) {
        CheckingAccount account = new CheckingAccount(1000);
        account.deposit(500);
        account.displayBalance();
    }
}
```

Output:

```
C:\Users\HP\Documents\java programs>javac Account.java

C:\Users\HP\Documents\java programs>java Account.java
Checking Account Balance: 1500.0
```

3.g)Measurements

Code:

```
class Room {

    double length, width;


    public Room(double length, double width) {
        this.length = length;
        this.width = width;
    }


    public double calculateArea() {
        return length * width;
    }
}


class RoomWithHeight extends Room {

    double height;


    public RoomWithHeight(double length, double width, double height) {
        super(length, width); // Call the parent class constructor
        this.height = height;
    }


    public double calculateVolume() {
        return calculateArea() * height;
    }
}

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

        RoomWithHeight room = new RoomWithHeight(5, 4, 3);


        double area = room.calculateArea();
        System.out.println("Room Area: " + area + " square meters");


        double volume = room.calculateVolume();
        System.out.println("Room Volume: " + volume + " cubic meters");
    }
}
```

Output:

```
C:\Users\HP\Documents\java programs>javac Measurements.java

C:\Users\HP\Documents\java programs>java Measurements.java
Room Area: 20.0 square meters
Room Volume: 60.0 cubic meters
```

3.g)AmystrongNumber

Code:

```
import java.util.Scanner;

public class ArmstrongNumber {
    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 result = 0;
        int digits = String.valueOf(num).length();

        while (num != 0) {
            int remainder = num % 10;
            result += Math.pow(remainder, digits);
            num /= 10;
        }

        if (result == originalNum) {
            System.out.println(originalNum + " is an Armstrong number.");
        } else {
            System.out.println(originalNum + " is not an Armstrong number.");
        }

        scanner.close();
    }
}
```

Output:

```
C:\Users\HP\Documents\java programs>javac ArmstrongNumber.java

C:\Users\HP\Documents\java programs>java ArmstrongNumber.java
Enter a number: 20
20 is not an Armstrong number.
```

3.i)Art

Code:

```
class Shape {
    public void draw() {
        System.out.println("Drawing a shape");
    }
}


class Circle extends Shape {

    public void draw() {
        System.out.println("Drawing a circle");
    }
}

public class Art {
    public static void main(String[] args) {
        Circle circle = new Circle();
        circle.draw();
    }
}
```

Output:

```
C:\Users\HP\Documents\java programs>javac Art.java

C:\Users\HP\Documents\java programs>java Art.java
Drawing a circle
```

3.j)Sounds

Code:

```
class Animal {
  public void animalSound() {
    System.out.println("The animal makes a sound");
  }
}

class Pig extends Animal {
  public void animalSound() {
    System.out.println("The pig says: wee wee");
  }
}

class Dog extends Animal {
  public void animalSound() {
    System.out.println("The dog says: bow wow");
  }
}

class Sounds {
  public static void main(String[] args) {
    Animal myAnimal = new Animal();
    Animal myPig = new Pig();
    Animal myDog = new Dog();
    myAnimal.animalSound();
    myPig.animalSound();
    myDog.animalSound();
  }
}
```

Output:

```
C:\Users\HP\Documents\java programs>java Sounds.java
The animal makes a sound
The pig says: wee wee
The dog says: bow wow
```

INHERITANCE

4.Single inheritance Programs:

4.a) Vehicle

Code:

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

    void stop() {
        System.out.println("Vehicle is stopping...");
    }
}
class Car extends Vehicle {
    void honk() {
        System.out.println("Car is honking...");
    }
}

public class Main {
    public static void main(String[] args) {
        Car myCar = new Car();
        myCar.start();
        myCar.honk();
        myCar.stop();
    }
}
```

Output:

```
C:\Users\TEMP.CH.000\Documents>java Main
Vehicle is starting...
Car is honking...
Vehicle is stopping...
```

4.b) Employee

Code:

```
class Employee {
    int empId;
    String name;
    Employee(int empId, String name) {
        this.empId = empId;
        this.name = name;
    }

    void displayDetails() {
        System.out.println("Employee ID: " + empId);
        System.out.println("Employee Name: " + name);
    }
}
class Manager extends Employee {
    String department;
    Manager(int empId, String name, String department) {
        super(empId, name);
        this.department = department;
    }

    void displayManagerDetails() {
        displayDetails();
        System.out.println("Department: " + department);
    }
}

public class Work {
    public static void main(String[] args) {
        Manager manager = new Manager(101, "John Doe", "HR");
        manager.displayManagerDetails();
    }
}
```

Output:

```
C:\Users\TEMP.CH.000\Documents>java Work
Employee ID: 101
Employee Name: John Doe
Department: HR
```

5.Multilevel inheritance Programs:

5.a) Plants

Code:

```
class Plant {
    void grow() {
        System.out.println("Plants grow.");
    }
}

class Flower extends Plant {
    void bloom() {
        System.out.println("Flowers bloom beautifully.");
    }
}

class Rose extends Flower {
    void smell() {
        System.out.println("Roses have a pleasant fragrance.");
    }
}

public class Herb {
    public static void main(String[] args) {
        Rose rose = new Rose();
        rose.grow();
        rose.bloom();
        rose.smell();
    }
}
```

Output:

```
C:\Users\TEMP.CH.000\Documents>java Herb
Plants grow.
Flowers bloom beautifully.
Roses have a pleasant fragrance.
```

5.b) Book

Code:

```
class Book {
    void read() {
        System.out.println("Books are read for knowledge or entertainment.");
    }
}

class Fiction extends Book {
    void imagination() {
        System.out.println("Fiction sparks imagination.");
    }
}

class Novel extends Fiction {
    void pages() {
        System.out.println("Novels often have hundreds of pages.");
    }
}

public class Read {
    public static void main(String[] args) {
        Novel novel = new Novel();
        novel.read();
        novel.imagination();
        novel.pages();
    }
}
```

Output:

```
C:\Users\TEMP.CH.000\Documents>java Read
Books are read for knowledge or entertainment.
Fiction sparks imagination.
Novels often have hundreds of pages.
```

6.Hierarchical inheritance Programs:

6.a) Shapes

Code:

```
class Shape {
    void describe() {
        System.out.println("This is a shape.");
    }
}
class Rectangle extends Shape {
    void area() {
        System.out.println("Area of rectangle = length × breadth");
    }
}
class Circle extends Shape {
    void area() {
        System.out.println("Area of circle = π × r × r");
    }
}

public class Polygon {
    public static void main(String[] args) {
        Rectangle rectangle = new Rectangle();
        rectangle.describe();
        rectangle.area();
        Circle circle = new Circle();
        circle.describe();
        circle.area();
    }
}
```

Output:

```
C:\Users\TEMP.CH.000\Documents>java Polygon
This is a shape.
Area of rectangle = length ? breadth
This is a shape.
Area of circle = π ? r ? r
```

6.b) Persons

Code:

```
class Person {
    void introduce() {
        System.out.println("I am a person.");
    }
}
class Student extends Person {
    void study() {
        System.out.println("Students study hard.");
    }
}
class Teacher extends Person {
    void teach() {
        System.out.println("Teachers teach with passion.");
    }
}

public class People {
    public static void main(String[] args) {
        Student student = new Student();
        student.introduce();
        student.study();
        Teacher teacher = new Teacher();
        teacher.introduce();
        teacher.teach();
    }
}
```

Output:

```
C:\Users\TEMP.CH.000\Documents>java People
I am a person.
Students study hard.
I am a person.
Teachers teach with passion.
```

7.Hybrid inheritance Programs:

7.a) Devices

Code:

```
class Device {
    void powerOn() {
        System.out.println("Device is powered on.");
    }
}
interface Printer {
    void print();
}
class Computer extends Device implements Printer {
    void process() {
        System.out.println("Computer is processing tasks.");
    }
    public void print() {
        System.out.println("Computer is printing documents.");
    }
}

public class Operator {
    public static void main(String[] args) {
        Computer computer = new Computer();
        computer.powerOn();
        computer.process();
        computer.print();
    }
}
```

Output:

```
C:\Users\TEMP.CH.000\Documents>java Operator
Device is powered on.
Computer is processing tasks.
Computer is printing documents.
```

7.b) Appliances

Code:

```
class Appliance {
    void switchOn() {
        System.out.println("Appliance is switched on.");
    }
}
interface Dryer {
    void dryClothes();
}
class WashingMachine extends Appliance implements Dryer {
    void washClothes() {
        System.out.println("Washing machine is washing clothes.");
    }
    public void dryClothes() {
        System.out.println("Washing machine is drying clothes.");
    }
}

public class Machine {
    public static void main(String[] args) {
        WashingMachine machine = new WashingMachine();
        machine.switchOn();
        machine.washClothes();
        machine.dryClothes();
    }
}
```

Output:

```
C:\Users\TEMP.CH.000\Documents>java Machine
Appliance is switched on.
Washing machine is washing clothes.
Washing machine is drying clothes.
```

POLYMORPHISM

8.Constructor programs:

8.a) Games

Code:

```
class Game {
    Game() {
        System.out.println("A new game is starting...");
    }
}

class Chess extends Game {
    Chess() {
        super();
        System.out.println("Chess game is ready.");
    }
}

class Puzzle extends Game {
    Puzzle() {
        super(); // Calls the Game constructor
        System.out.println("Puzzle game is ready.");
    }
}

public class Play {
    public static void main(String[] args) {
        Chess chess = new Chess();
        Puzzle puzzle = new Puzzle();
    }
}
```

Output:

```
C:\Users\TEMP.CH.000\Documents>java Play
A new game is starting...
Chess game is ready.
A new game is starting...
Puzzle game is ready.
```

9.Constructor overloading programs:

9.a) Food

Code:

```
class IceCream {
    String flavor;
    int scoops;
    IceCream() {
        flavor = "Vanilla";
        scoops = 1;
    }
    IceCream(String f) {
        flavor = f;
        scoops = 1;
    }
IceCream(String f, int s) {
        flavor = f;
        scoops = s;
    }

    void display() {
        System.out.println("Flavor: " + flavor + ", Scoops: " + scoops);
    }
}

public class Food {
    public static void main(String[] args) {
        IceCream ice1 = new IceCream();
        ice1.display();
        IceCream ice2 = new IceCream("Chocolate");
        ice2.display();
        IceCream ice3 = new IceCream("Strawberry", 3);
        ice3.display();
    }
}
```

Output:

```
C:\Users\HP\Documents\java programs>
C:\Users\HP\Documents\java programs>java  Food.java
Flavor: Vanilla, Scoops: 1
Flavor: Chocolate, Scoops: 1
Flavor: Strawberry, Scoops: 3
```

10.Method overloading programs:

10.a) Account

Code:

```
class Account {
    private String accountType;
    private double balance;
    Account() {
        accountType = "Default Account";
        balance = 0.0;
    }
    Account(String accountType) {
        this.accountType = accountType;
        balance = 0.0;
    }
    Account(String accountType, double balance) {
        this.accountType = accountType;
        this.balance = balance;
    }

    void displayDetails() {
        System.out.println("Account Type: " + accountType);
        System.out.println("Balance: $" + balance);
    }
}

public class Amount {
    public static void main(String[] args) {
        Account defaultAccount = new Account();
        defaultAccount.displayDetails();

        Account savingsAccount = new Account("Savings Account");
        savingsAccount.displayDetails();
        Account premiumAccount = new Account("Premium Account", 1000.0);
        premiumAccount.displayDetails();
    }
}
```

Output:

```
C:\Users\TEMP.CH.000\Documents>java Amount
Account Type: Default Account
Balance: $0.0
Account Type: Savings Account
Balance: $0.0
Account Type: Premium Account
Balance: $1000.0
```

10.b) orders

Code:

```
class Order {
    private int orderId;
    private String itemName;
    private int quantity;
    Order() {
        orderId = 0;
        itemName = "Unknown Item";
        quantity = 0;
    }
    Order(int orderId) {
        this.orderId = orderId;
        itemName = "Unknown Item";
        quantity = 0;
    }
    Order(int orderId, String itemName) {
        this.orderId = orderId;
        this.itemName = itemName;
        quantity = 0;
    }
    Order(int orderId, String itemName, int quantity) {
        this.orderId = orderId;
        this.itemName = itemName;
        this.quantity = quantity;
    }
    void displayOrderDetails() {
        System.out.println("Order ID: " + orderId);
        System.out.println("Item Name: " + itemName);
        System.out.println("Quantity: " + quantity);
    }
}

public class Amazon{
    public static void main(String[] args) {
        Order defaultOrder = new Order();
        defaultOrder.displayOrderDetails();
        Order singleOrder = new Order(101);
        singleOrder.displayOrderDetails();
        Order itemOrder = new Order(102, "Laptop");
        itemOrder.displayOrderDetails();
        Order fullOrder = new Order(103, "Smartphone", 2);
        fullOrder.displayOrderDetails();
```

Output:

```
C:\Users\TEMP.CH.000\Documents>java Amazon
Order ID: 0
Item Name: Unknown Item
Quantity: 0
Order ID: 101
Item Name: Unknown Item
Quantity: 0
Order ID: 102
Item Name: Laptop
Quantity: 0
Order ID: 103
Item Name: Smartphone
Quantity: 2
```

11.Method overriding programs:

11.a) Network

Code:

```
class Network {
    Network() {
        System.out.println("Initializing network...");
    }
}

class WiredNetwork extends Network {
    WiredNetwork() {
        super(); // Calls the Network constructor
        System.out.println("Wired Network is set up.");
    }
}

class WirelessNetwork extends Network {
    WirelessNetwork() {
        super(); // Calls the Network constructor
        System.out.println("Wireless Network is set up.");
    }
}

public class Connection {
    public static void main(String[] args) {
        WiredNetwork wired = new WiredNetwork();
        WirelessNetwork wireless = new WirelessNetwork();
    }
}
```

Output:

```
C:\Users\TEMP.CH.000\Documents>java Connection
Initializing network...
Wired Network is set up.
Initializing network...
Wireless Network is set up.
```

11.b)Payment Systems

Code:

```
class PaymentSystem {
    PaymentSystem() {
        System.out.println("Initializing payment system...");
    }
}

class CreditCardPayment extends PaymentSystem {
    CreditCardPayment() {
        super(); // Calls the PaymentSystem constructor
        System.out.println("Credit Card Payment is being processed.");
    }
}

class PayPalPayment extends PaymentSystem {
    PayPalPayment() {
        super(); // Calls the PaymentSystem constructor
        System.out.println("PayPal Payment is being processed.");
    }
}

public class Payment {
    public static void main(String[] args) {
        CreditCardPayment creditCard = new CreditCardPayment();
        PayPalPayment payPal = new PayPalPayment();
    }
}
```

Output:

```
C:\Users\TEMP.CH.000\Documents>java Payment
Initializing payment system...
Credit Card Payment is being processed.
Initializing payment system...
PayPal Payment is being processed.
```

ABSTRACTION

INTERFACE PROGRAMS:

12.A)Operations

Code:

```
abstract class Operation {
    abstract void performOperation();
}
interface Result {
    void displayResult();
}
class Addition extends Operation implements Result {
    private int a, b, sum;

    Addition(int a, int b) {
        this.a = a;
        this.b = b;
    }
    void performOperation() {
        sum = a + b;
    }
    public void displayResult() {
        System.out.println("Addition Result: " + sum);
    }
}

public class Sings {
    public static void main(String[] args) {
        Addition addition = new Addition(10, 20);
        addition.performOperation();
        addition.displayResult();
    }
}
```

Output:

```
C:\Users\HP\Documents\java programs>javac Sings.java

C:\Users\HP\Documents\java programs>java Sings
Addition Result:30
```

12.b)FileHandler

Code:

```
abstract class FileHandler {
    abstract void openFile(String fileName);
}
interface FileOperations {
    void readFile();
    void closeFile();
}
class TextFileHandler extends FileHandler implements FileOperations {
    private String fileName;
    void openFile(String fileName) {
        this.fileName = fileName;
        System.out.println("Opening file: " + fileName);
    }
    public void readFile() {
        System.out.println("Reading content from file: " + fileName);
    }
    public void closeFile() {
        System.out.println("Closing file: " + fileName);
    }
}

public class Files {
    public static void main(String[] args) {
        TextFileHandler fileHandler = new TextFileHandler();
        fileHandler.openFile("data.txt");
        fileHandler.readFile();
        fileHandler.closeFile();
    }
}
```

output:

```
C:\Users\CH.SC.U4CSE24143\Downloads>java Files
Opening file: data.txt
Reading content from file: data.txt
Closing file: data.txt
```

12.c)Data Base Connection

Code:

```
abstract class DatabaseConnection {
    abstract void connectToDatabase();
}
interface QueryExecutor {
    void executeQuery(String query);
}
class MySQLConnection extends DatabaseConnection implements QueryExecutor {
    void connectToDatabase() {
        System.out.println("Connecting to MySQL database...");
    }
    public void executeQuery(String query) {
        System.out.println("Executing query: " + query);
    }
}

public class Data {
    public static void main(String[] args) {
        MySQLConnection mySQL = new MySQLConnection();
        mySQL.connectToDatabase();
        mySQL.executeQuery("SELECT * FROM users;");
    }
}
```

Output:

```
C:\Users\CH.SC.U4CSE24143\Downloads>java Data
Connecting to MySQL database...
Executing query: SELECT * FROM users;
```

12.d)Shapes

Code:

```
abstract class Shape {
    public double getArea() {
        return 0.0;  // Default area
    }
    public abstract void draw();
}

interface Colorable {
    void color(String color);
}
class Circle extends Shape implements Colorable {
    private double radius;
    public Circle(double radius) {
        this.radius = radius;
    }
    public void draw() {
        System.out.println("Drawing a circle with radius: " + radius);
    }
    public double getArea() {
        return Math.PI * radius * radius;  // Area of the circle
    }
    public void color(String color) {
        System.out.println("Coloring the circle with color: " + color);
    }
}

public class Shapes {
    public static void main(String[] args) {
        Shape myShape = new Circle(5.0);
        myShape.draw();
        System.out.println("Area: " + myShape.getArea());
        Colorable colorableShape = (Colorable) myShape;
        colorableShape.color("Red");
    }
}
```

Output:

```
C:\Users\CH.SC.U4CSE24143\Downloads>java Shapes
Drawing a circle with radius: 5.0
Area: 78.53981633974483
Coloring the circle with color: Red
```

13. ABSTRACT CLASS PROGRAMS

13.a)Tool

Code:

```
abstract class Device {
    public abstract double powerConsumption();
}
class Laptop extends Device {
    private double powerPerHour;

    public Laptop(double powerPerHour) {
        this.powerPerHour = powerPerHour;
    }
    public double powerConsumption() {
        return powerPerHour * 24;
    }
}
class Refrigerator extends Device {
    private double powerPerHour;

    public Refrigerator(double powerPerHour) {
        this.powerPerHour = powerPerHour;
    }
    public double powerConsumption() {
        return powerPerHour * 24 * 30;
    }
}

public class Tool {
    public static void main(String[] args) {
        Device laptop = new Laptop(0.5);
        Device fridge = new Refrigerator(1.2);

        System.out.println("Laptop power consumption in a day: " + laptop.powerConsumption() + " kWh");
        System.out.println("Fridge power consumption in a month: " + fridge.powerConsumption() + " kWh");
    }
}
```

Output:

```
C:\Users\HP\Documents\java programs>java Tool
Laptop power consumption in a day: 12.0 kWh
Fridge power consumption in a month: 863.9999999999999 kWh
```

13.b)Document

Code:

```
abstract class Document {
    public abstract void read();
    public abstract void write(String content);
}
class TextFile extends Document {
    private String content;
    public void read() {
        System.out.println("Reading text content: " + content);
    }
    public void write(String content) {
        this.content = content;
        System.out.println("Writing to text file: " + content);
    }
}

class PDF extends Document {
    private String content;
    public void read() {
        System.out.println("Reading PDF content: " + content);
    }
    public void write(String content) {
        this.content = content;
        System.out.println("Writing to PDF: " + content);
    }
}

public class Doc {
    public static void main(String[] args) {
        Document textFile = new TextFile();
        Document pdfFile = new PDF();

        textFile.write("Hello, this is a text document.");
        pdfFile.write("This is a PDF document.");

        textFile.read();
        pdfFile.read();
    }
}
```

Output:

```
C:\Users\HP\Documents\java programs>java Doc
Writing to text file: Hello, this is a text document.
Writing to PDF: This is a PDF document.
Reading text content: Hello, this is a text document.
Reading PDF content: This is a PDF document.
```

13.c)Pay

Code:

```
abstract class PaymentMethod {
    public abstract void processPayment(double amount);
}
class CreditCardPayment extends PaymentMethod {
    public void processPayment(double amount) {
        System.out.println("Processing credit card payment of $" + amount);
    }
}
class PayPalPayment extends PaymentMethod {
    public void processPayment(double amount) {
        System.out.println("Processing PayPal payment of $" + amount);
    }
}
class BitcoinPayment extends PaymentMethod {
    public void processPayment(double amount) {
        System.out.println("Processing Bitcoin payment of $" + amount);
    }
}

public class Pay {
    public static void main(String[] args) {
        PaymentMethod creditCard = new CreditCardPayment();
        PaymentMethod payPal = new PayPalPayment();
        PaymentMethod bitcoin = new BitcoinPayment();

        creditCard.processPayment(100.0);
        payPal.processPayment(250.5);
        bitcoin.processPayment(450.75);
    }
}
```

Output:

```
C:\Users\HP\Documents\java programs>java Pay
Processing credit card payment of $100.0
Processing PayPal payment of $250.5
Processing Bitcoin payment of $450.75
```

13.d) Data Base

Code:

```
abstract class DatabaseConnection {
    public abstract void connect();
    public abstract void executeQuery(String query);
}
class MySQLConnection extends DatabaseConnection {
    public void connect() {
        System.out.println("Connecting to MySQL Database...");
    }
    public void executeQuery(String query) {
        System.out.println("Executing MySQL query: " + query);
    }
}

class OracleConnection extends DatabaseConnection {
    public void connect() {
        System.out.println("Connecting to Oracle Database...");
    }
    public void executeQuery(String query) {
        System.out.println("Executing Oracle query: " + query);
    }
}

public class Base {
    public static void main(String[] args) {
        DatabaseConnection mysql = new MySQLConnection();
        DatabaseConnection oracle = new OracleConnection();

        mysql.connect();
        mysql.executeQuery("SELECT * FROM users");

        oracle.connect();
        oracle.executeQuery("SELECT * FROM employees");
    }
}
```

Output:

```
C:\Users\HP\Documents\java programs>java Base
Connecting to MySQL Database...
Executing MySQL query: SELECT * FROM users
Connecting to Oracle Database...
Executing Oracle query: SELECT * FROM employees
```

ENCAPSULATION

ENCAPSULATION PROGRAMS:

14.a)Bank Account:

Code:

```
class BankAccount {
    private double balance;
    public BankAccount(double initialBalance) {
        if (initialBalance >= 0) {
            balance = initialBalance;
        } else {
            System.out.println("Invalid initial balance.");
        }
    }
    public double getBalance() {
        return balance;
    }
    public void deposit(double amount) {
        if (amount > 0) {
            balance += amount;
        } else {
            System.out.println("Deposit amount must be positive.");
        }
    }
    public void withdraw(double amount) {
        if (amount > 0 && amount <= balance) {
            balance -= amount;
        } else {
            System.out.println("Invalid withdrawal amount.");
        }
    }
}
public class Bank {
    public static void main(String[] args) {
        BankAccount account = new BankAccount(500);

        account.deposit(200);
        System.out.println("Balance after deposit: $" + account.getBalance());

        account.withdraw(150);
        System.out.println("Balance after withdrawal: $" + account.getBalance());

        account.withdraw(600);
    }
}
```

Output:

```
C:\Users\HP\Documents\java programs>java Bank
Balance after deposit: $700.0
Balance after withdrawal: $550.0
Invalid withdrawal amount.
```

14.b)Products

Code:

```
class Product {
    private String name;
    private double price;
    public Product(String name, double price) {
        this.name = name;
        setPrice(price);
    }
    public String getName() {
        return name;
    }
    public double getPrice() {
        return price;
    }
    public void setPrice(double price) {
        if (price >= 0) {
            this.price = price;
        } else {
            System.out.println("Price cannot be negative.");
        }
    }
    public void applyDiscount(double discountPercent) {
        if (discountPercent > 0 && discountPercent <= 100) {
            price = price - (price * discountPercent / 100);
        } else {
            System.out.println("Invalid discount percentage.");
        }
    }
}
public class Products {
    public static void main(String[] args) {
        Product product = new Product("Laptop", 1000);
        System.out.println("Product Name: " + product.getName());
        System.out.println("Original Price: $" + product.getPrice());
        product.applyDiscount(10);
        System.out.println("Price after 10% discount: $" + product.getPrice());
        product.setPrice(1200); // Setting a new price
        System.out.println("New Price: $" + product.getPrice());
    }
}
```

Output:

```
C:\Users\HP\Documents\java programs>java Products
Product Name: Laptop
Original Price: $1000.0
Price after 10% discount: $900.0
New Price: $1200.0
```

14.c)Motor

Code:

```
class Car {
    private int speed;
    public Car() {
        speed = 0;
    }
    public int getSpeed() {
        return speed;
    }
    public void accelerate(int increment) {
        if (increment > 0) {
            speed += increment;
        } else {
            System.out.println("Speed increment must be positive.");
        }
    }
    public void decelerate(int decrement) {
        if (decrement > 0 && speed - decrement >= 0) {
            speed -= decrement;
        } else {
            System.out.println("Invalid deceleration value.");
        }
    }
}
public class Motor {
    public static void main(String[] args) {
        Car car = new Car();
        System.out.println("Initial Speed: " + car.getSpeed() + " km/h");
        car.accelerate(50);
        System.out.println("Speed after accelerating: " + car.getSpeed() + " km/h");
        car.decelerate(20);
        System.out.println("Speed after decelerating: " + car.getSpeed() + " km/h");
        car.decelerate(50);
    }
}
```

Output:

```
C:\Users\HP\Documents\java programs>java Motor
Initial Speed: 0 km/h
Speed after accelerating: 50 km/h
Speed after decelerating: 30 km/h
Invalid deceleration value.
```

14.d) Temperature

Code:

```
class Temperature {
    private double temperatureInCelsius;
    public Temperature(double temperatureInCelsius) {
        this.temperatureInCelsius = temperatureInCelsius;
    }
    public double getTemperatureInCelsius() {
        return temperatureInCelsius;
    }
    public double toFahrenheit() {
        return (temperatureInCelsius * 9/5) + 32;
    }
    public double toKelvin() {
        return temperatureInCelsius + 273.15;
    }
}
public class Temp {
    public static void main(String[] args) {
        Temperature temp = new Temperature(25);
        System.out.println("Temperature in Celsius: " + temp.getTemperatureInCelsius());
        System.out.println("Temperature in Fahrenheit: " + temp.toFahrenheit());
        System.out.println("Temperature in Kelvin: " + temp.toKelvin());
    }
}
```

Output:

```
C:\Users\HP\Documents\java programs>java Temp
Temperature in Celsius: 25.0
Temperature in Fahrenheit: 77.0
Temperature in Kelvin: 298.15
```

PACKAGES PROGRAMS:

15.a) Library(user-interface)

Code:

```
import java.util.ArrayList;
class Book {
    private String title;
    private String author;
    private boolean isCheckedOut;
    public Book(String title, String author) {
        this.title = title;
        this.author = author;
        this.isCheckedOut = false;
    }
    public String getTitle() {
        return title;
    }
    public String getAuthor() {
        return author;
    }
    public boolean isCheckedOut() {
        return isCheckedOut;
    }
    public void checkOut() {
        if (!isCheckedOut) {
            isCheckedOut = true;
            System.out.println("The book '" + title + "' has been checked out.");
        } else {
            System.out.println("Sorry, the book '" + title + "' is already checked out.");
        }
    }
    public void returnBook() {
        if (isCheckedOut) {
            isCheckedOut = false;
            System.out.println("The book '" + title + "' has been returned.");
        } else {
            System.out.println("This book was not checked out.");
        }
    }
}
class Library {
    private ArrayList<Book> books;
    public Library() {
        books = new ArrayList<>();
    }
    public void addBook(Book book) {
        books.add(book);
        System.out.println("The book '" + book.getTitle() + "' by " + book.getAuthor() + " has been added to the library.");
    }
    public void listBooks() {
        if (books.isEmpty()) {
            System.out.println("The library is empty.");
        } else {
            System.out.println("Books in the library:");
            for (Book book : books) {
                System.out.println("- " + book.getTitle() + " by " + book.getAuthor() + (book.isCheckedOut() ? " (Checked Out)" : " (Available)"));
            }
        }
    }
}

public class Refcen {
    public static void main(String[] args) {
        Library library = new Library();
        Book book1 = new Book("1984", "George Orwell");
        Book book2 = new Book("To Kill a Mockingbird", "Harper Lee");
        library.addBook(book1);
        library.addBook(book2);
        library.listBooks();
        book1.checkOut();
        book1.checkOut();
        book1.returnBook();
        library.listBooks();
    }
}
```

output:

```
C:\Users\HP\Documents\java programs>java Refcen
The book '1984' by George Orwell has been added to the library.
The book 'To Kill a Mockingbird' by Harper Lee has been added to the library.
Books in the library:
- 1984 by George Orwell (Available)
- To Kill a Mockingbird by Harper Lee (Available)
The book '1984' has been checked out.
Sorry, the book '1984' is already checked out.
The book '1984' has been returned.
Books in the library:
- 1984 by George Orwell (Available)
- To Kill a Mockingbird by Harper Lee (Available)
```

15.b)Maps(user-interface)

Code:

```
import java.util.HashMap;
import java.util.Map;
class Product {
    private String name;
    private double price;
    private int quantity;
    public Product(String name, double price, int quantity) {
        this.name = name;
        this.price = price;
        this.quantity = quantity;
    }
    public String getName() {
        return name;
    }
    public double getPrice() {
        return price;
    }
    public int getQuantity() {
        return quantity;
    }
    public void updateQuantity(int quantity) {
        if (this.quantity + quantity >= 0) {
            this.quantity += quantity;
            System.out.println("Updated quantity of '" + name + "' to " + this.quantity);
        } else {
            System.out.println("Not enough stock of '" + name + "' to reduce quantity by " + quantity);
        }
    }
}

class Inventory {
    private Map<String, Product> products;
    public Inventory() {
        products = new HashMap<>();
    }
    public void addProduct(Product product) {
        products.put(product.getName(), product);
        System.out.println("Product '" + product.getName() + "' has been added to the inventory.");
    }
    public void checkStock(String productName) {
        Product product = products.get(productName);
        if (product != null) {
            System.out.println("Product: " + product.getName() + ", Price: $" + product.getPrice() + ", Quantity: " + product.getQuantity());
        } else {
            System.out.println("Product '" + productName + "' not found in inventory.");
        }
    }
    public void updateProductQuantity(String productName, int quantity) {
        Product product = products.get(productName);
        if (product != null) {
            product.updateQuantity(quantity);
        } else {
            System.out.println("Product '" + productName + "' not found in inventory.");
        }
    }
}

public class Maps {
    public static void main(String[] args) {
        Inventory inventory = new Inventory();
        Product product1 = new Product("Laptop", 1200.0, 10);
        Product product2 = new Product("Smartphone", 800.0, 50);
        inventory.addProduct(product1);
        inventory.addProduct(product2);
        inventory.checkStock("Laptop");
        inventory.checkStock("Smartphone");
        inventory.updateProductQuantity("Laptop", -5);  // Sell 5 laptops
        inventory.updateProductQuantity("Smartphone", 20);  // Add 20 smartphones
        inventory.checkStock("Laptop");
        inventory.checkStock("Smartphone");
    }
}
```

Output:

```
C:\Users\HP\Documents\java programs>java Maps
Product 'Laptop' has been added to the inventory.
Product 'Smartphone' has been added to the inventory.
Product: Laptop, Price: $1200.0, Quantity: 10
Product: Smartphone, Price: $800.0, Quantity: 50
Updated quantity of 'Laptop' to 5
Updated quantity of 'Smartphone' to 70
Product: Laptop, Price: $1200.0, Quantity: 5
Product: Smartphone, Price: $800.0, Quantity: 70
```

15.c)Files(Built-in package)

Code:

```
import java.io.File;
import java.io.FileWriter;
import java.io.FileReader;
import java.io.BufferedReader;
import java.io.IOException;

public class Files {
    public static void main(String[] args) {
        File file = new File("example.txt");
        try (FileWriter writer = new FileWriter(file)) {
            writer.write("Hello, this is a test file for file handling in Java!\n");
            writer.write("We are using the java.io package for this task.");
            System.out.println("Data written to file successfully.");
        } catch (IOException e) {
            System.out.println("An error occurred while writing to the file.");
            e.printStackTrace();
        }
        try (FileReader reader = new FileReader(file);
             BufferedReader bufferedReader = new BufferedReader(reader)) {
            String line;
            System.out.println("\nReading from the file:");
            while ((line = bufferedReader.readLine()) != null) {
                System.out.println(line);
            }
        } catch (IOException e) {
            System.out.println("An error occurred while reading from the file.");
            e.printStackTrace();
        }
    }
}
```

Output:

```
C:\Users\HP\Documents\java programs>java Files
Data written to file successfully.

Reading from the file:
Hello, this is a test file for file handling in Java!
We are using the java.io package for this task.
```

15.d)DateTime(bulid-in packages)

Code:

```
import java.time.LocalDate;
import java.time.LocalTime;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;

public class DT{
    public static void main(String[] args) {
        LocalDate currentDate = LocalDate.now();
        System.out.println("Current Date: " + currentDate);
        LocalTime currentTime = LocalTime.now();
        System.out.println("Current Time: " + currentTime);
        LocalDateTime currentDateTime = LocalDateTime.now();
        System.out.println("Current Date and Time: " + currentDateTime);
        LocalDate futureDate = currentDate.plusDays(10);
        System.out.println("Date after 10 days: " + futureDate);
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("dd/MM/yyyy HH:mm:ss");
        String formattedDate = currentDateTime.format(formatter);
        System.out.println("Formatted Date and Time: " + formattedDate);
    }
}
```

Output:

```
C:\Users\HP\Documents\java programs>java DT
Current Date: 2025-04-03
Current Time: 19:14:12.096746800
Current Date and Time: 2025-04-03T19:14:12.096746800
Date after 10 days: 2025-04-13
Formatted Date and Time: 03/04/2025 19:14:12
```

16) EXCEPTION HANDLING PROGRAMS:

16.a)DivisionByZero

Code:

```
import java.util.Scanner;

public class DivisionByZero {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        try {
            System.out.print("Enter the numerator: ");
            int numerator = scanner.nextInt();
            System.out.print("Enter the denominator: ");
            int denominator = scanner.nextInt();

            int result = numerator / denominator;
            System.out.println("The result is: " + result);
        } catch (ArithmeticException e) {
            System.out.println("Error: Cannot divide by zero.");
        } catch (java.util.InputMismatchException e) {
            System.out.println("Error: Please enter valid integers.");
        } catch (Exception e) {
            System.out.println("An unexpected error occurred: " + e.getMessage());
        } finally {
            scanner.close();
        }
    }
}
```

Output:

```
C:\Users\HP\Documents\java programs>javac DivisionByZero.java

C:\Users\HP\Documents\java programs>java DivisionByZero
Enter the numerator: 12
Enter the denominator: 4
The result is: 3
```

16.b)ArrayIndexHandling:

Code:

```
import java.util.Scanner;

public class ArrayIndexHandling {
    public static void main(String[] args) {
        int[] numbers = {10, 20, 30, 40, 50};
        Scanner scanner = new Scanner(System.in);

        try {
            System.out.print("Enter an index (0-4) to access the array element: ");
            int index = scanner.nextInt();
            System.out.println("Element at index " + index + ": " + numbers[index]);
        } catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("Error: Index is out of bounds. Valid index range is 0 to 4.");
        } catch (java.util.InputMismatchException e) {
            System.out.println("Error: Invalid input. Please enter an integer.");
        } catch (Exception e) {
            System.out.println("An unexpected error occurred: " + e.getMessage());
        } finally {
            scanner.close();
        }
    }
}
```

Output:

```
C:\Users\HP\Documents\java programs>javac ArrayIndexHandling.java

C:\Users\HP\Documents\java programs>java ArrayIndexHandling
Enter an index (0-4) to access the array element: 3
Element at index 3: 40
```

16.c)FileExceptionHandling:

Code:

```
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.util.Scanner;

public class FileExceptionHandling {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        try {
            System.out.print("Enter the file path: ");
            String filePath = scanner.nextLine();
            File file = new File(filePath);
            FileReader fileReader = new FileReader(file);
            int character;

            while ((character = fileReader.read()) != -1) {
                System.out.print((char) character);
            }
            fileReader.close();
        } catch (FileNotFoundException e) {
            System.out.println("Error: The file was not found.");
        } catch (IOException e) {
            System.out.println("Error: An IO error occurred while reading the file.");
        } catch (Exception e) {
            System.out.println("An unexpected error occurred: " + e.getMessage());
        } finally {
            scanner.close();
        }
    }
}
```

Output:

```
C:\Users\HP\Documents\java programs>javac FileExceptionHandling.java

C:\Users\HP\Documents\java programs>java FileExceptionHandling
Enter the file path: 60
Error: The file was not found.
```

16.d)NumberFormatHandling:

Code:

```
import java.util.Scanner;

public class NumberFormatHandling {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        try {
            System.out.print("Enter a number: ");
            String input = scanner.nextLine();
            int number = Integer.parseInt(input);  // This could throw NumberFormatException

            System.out.println("You entered: " + number);
        } catch (NumberFormatException e) {
            System.out.println("Error: Invalid number format. Please enter a valid integer.");
        } catch (Exception e) {
            System.out.println("An unexpected error occurred: " + e.getMessage());
        } finally {
            scanner.close();
        }
    }
}
```

Output:

```
C:\Users\HP\Documents\java programs>javac NumberFormatHandling.java

C:\Users\HP\Documents\java programs>java NumberFormatHandling
Enter a number: 5
You entered: 5
```

17.FILE HANDLING PROGRAMS:

17.a)WriteToFile

Code:

```
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;

public class WriteToFile {
    public static void main(String[] args) {
        try {
            // Create a new file
            File file = new File("output.txt");

            // Check if the file already exists
            if (!file.exists()) {
                file.createNewFile();  // Create the file if it doesn't exist
            }

            // Create FileWriter object to write data to the file
            FileWriter writer = new FileWriter(file);
            writer.write("This is a test message written to the file.\n");
            writer.write("File handling in Java is easy to implement.");
            writer.close();  // Close the writer

            System.out.println("Data has been written to the file successfully.");
        } catch (IOException e) {
            System.out.println("An error occurred during file handling: " + e.getMessage());
        }
    }
}
```

Output:

```
C:\Users\HP\Documents\java programs>javac WriteToFile.java

C:\Users\HP\Documents\java programs>java WriteToFile
Data has been written to the file successfully.
```

17.b)ReadFromFile:

Code:

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;

public class ReadFromFile {
    public static void main(String[] args) {
        try {
            // Create BufferedReader to read from the file
            BufferedReader reader = new BufferedReader(new FileReader("output.txt"));
            String line;
            while ((line = reader.readLine()) != null) {
                System.out.println(line);
            }

            reader.close();  // Close the reader
        } catch (IOException e) {
            System.out.println("An error occurred while reading the file: " + e.getMessage());
        }
    }
}
```

Output:

```
C:\Users\HP\Documents\java programs>java ReadFromFile
This is a test message written to the file.
File handling in Java is easy to implement.
```

17.c) AppendToFile

Code:

```
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;

public class AppendToFile {
    public static void main(String[] args) {
        try {
            FileWriter writer = new FileWriter("output.txt", true);
            writer.write("\nThis line is appended to the file.");
            writer.close();
            System.out.println("Data has been appended to the file successfully.");
        } catch (IOException e) {
            System.out.println("An error occurred during file handling: " + e.getMessage());
        }
    }
}
```

Output:

```
C:\Users\HP\Documents\java programs>javac AppendToFile.java

C:\Users\HP\Documents\java programs>java AppendToFile
Data has been appended to the file successfully.
```

17.d)DeleteFile:

Code:

```
import java.io.File;

public class DeleteFile {
    public static void main(String[] args) {
        try {
            File file = new File("output.txt");
            if (file.exists()) {
                if (file.delete()) {
                    System.out.println("File deleted successfully.");
                } else {
                    System.out.println("Failed to delete the file.");
                }
            } else {
                System.out.println("File does not exist.");
            }
        } catch (Exception e) {
            System.out.println("An error occurred during file handling: " + e.getMessage());
        }
    }
}
```

Output:

```
C:\Users\HP\Documents\java programs>javaDeleteFile
'javaDeleteFile' is not recognized as an internal or external command,
operable program or batch file.
```

59