LAB RECORD

23CSE111- Object Oriented Programming

Submitted by

CH.SC.U4CSE24142 -Santhosh

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.U4CSE24142 – Santhosh A*** 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 08 /04/2025

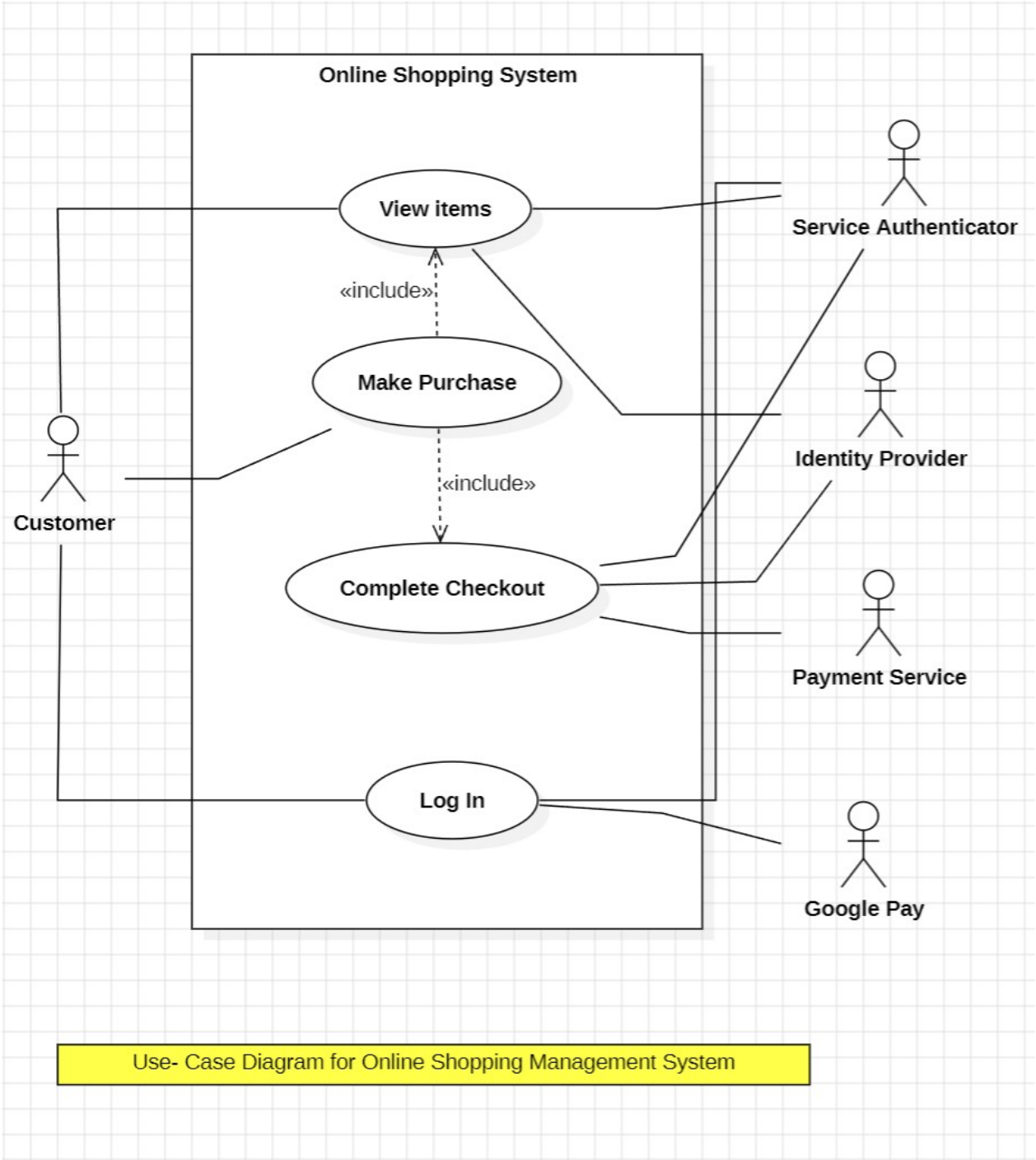
Internal Examiner 1 Internal Examiner 2

INDEX

UML DIAGRAMS

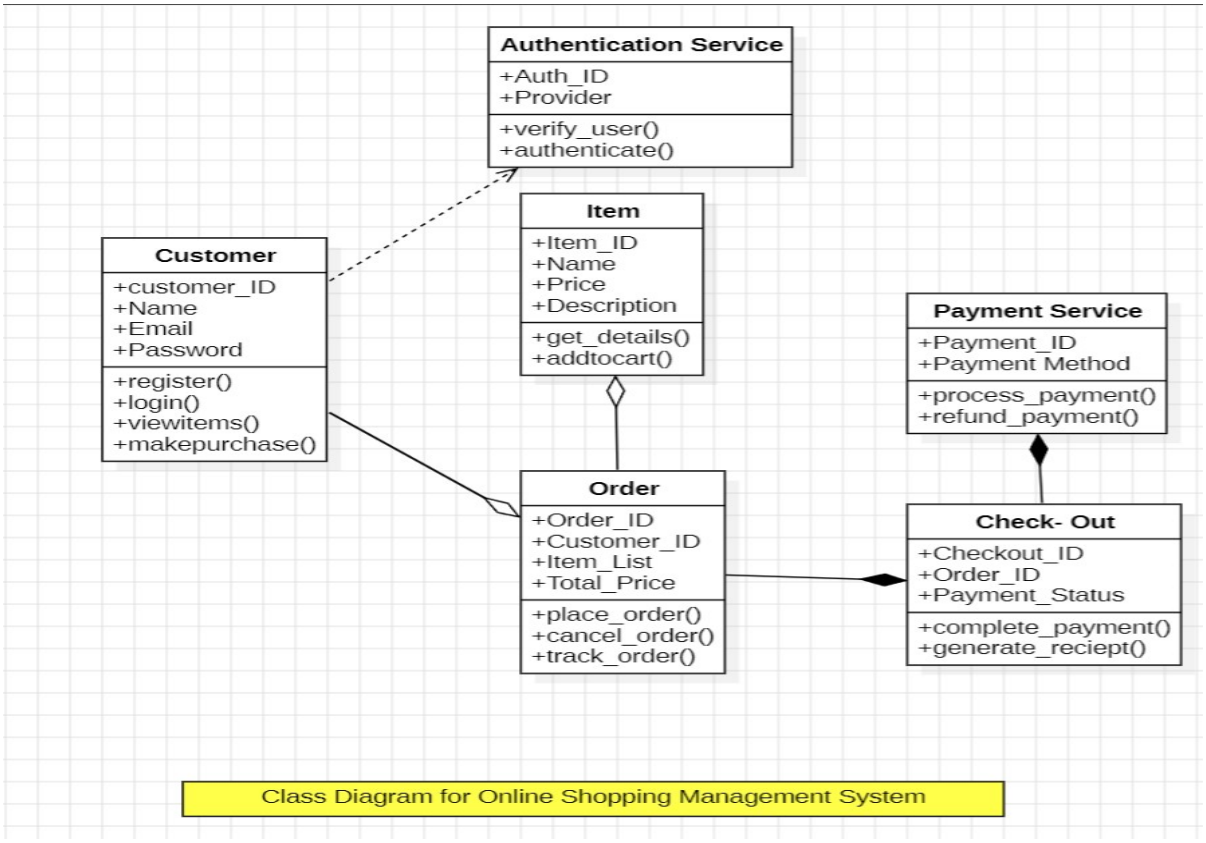
1. ONLINE SHOPPING MANAGEMENT SYSTEM

1.a) USE CASE DIAGRAM:



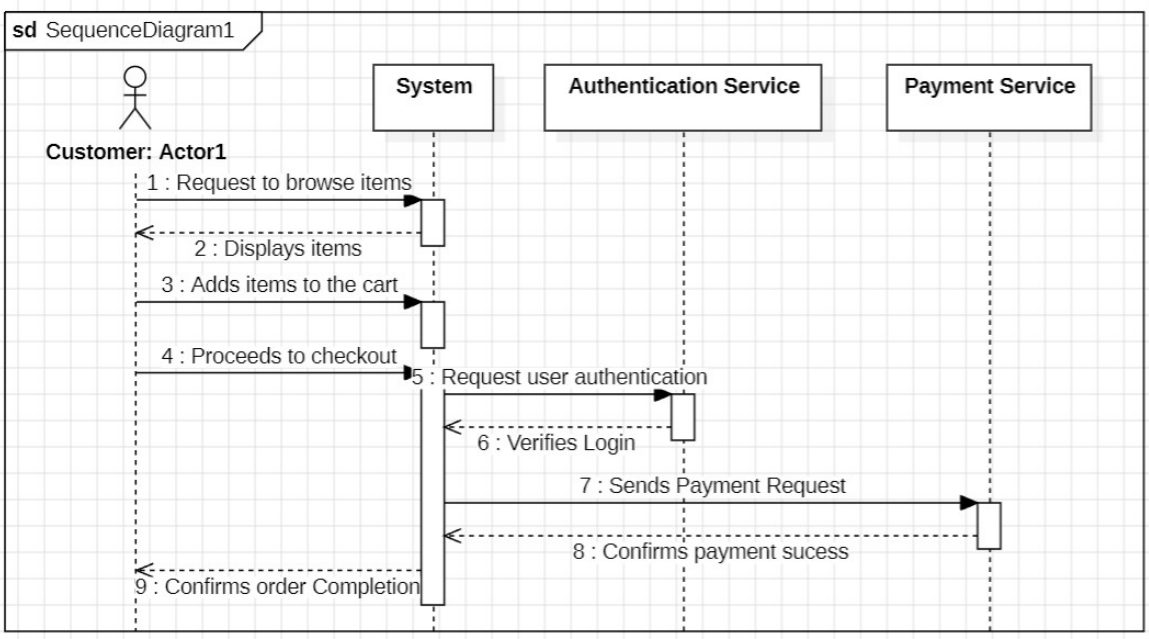
Use- Case Diagram for Online Shopping Management System

1.b) CLASS DIAGRAM:



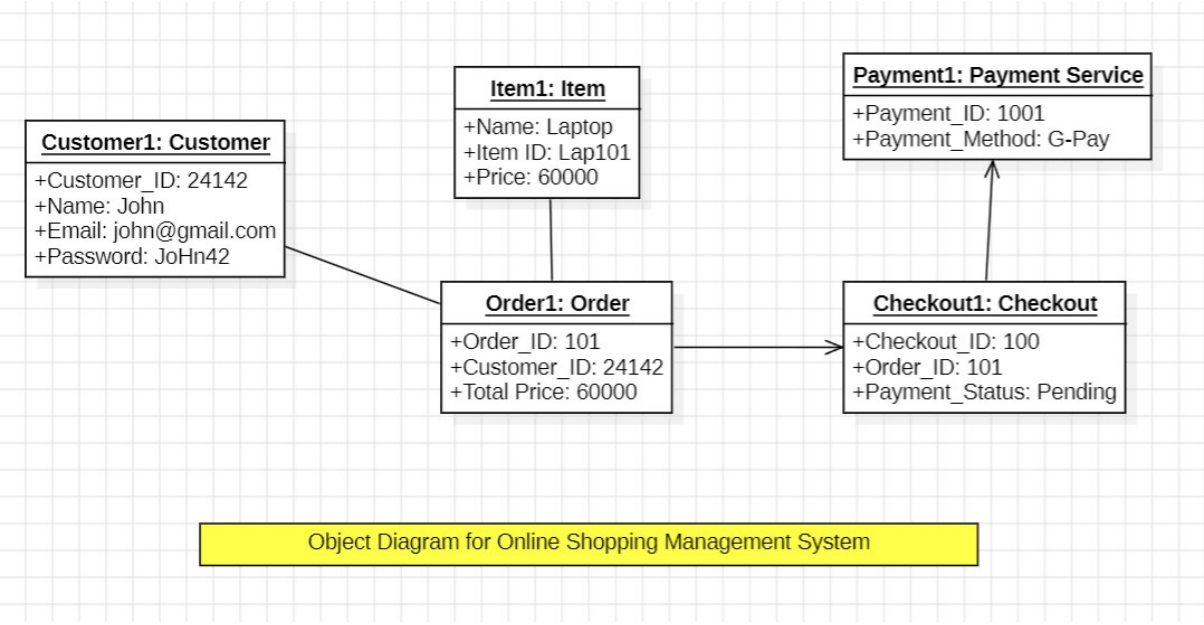
Class Diagram for Online Shopping Management System

1. c) SEQUENCE DIAGRAM:

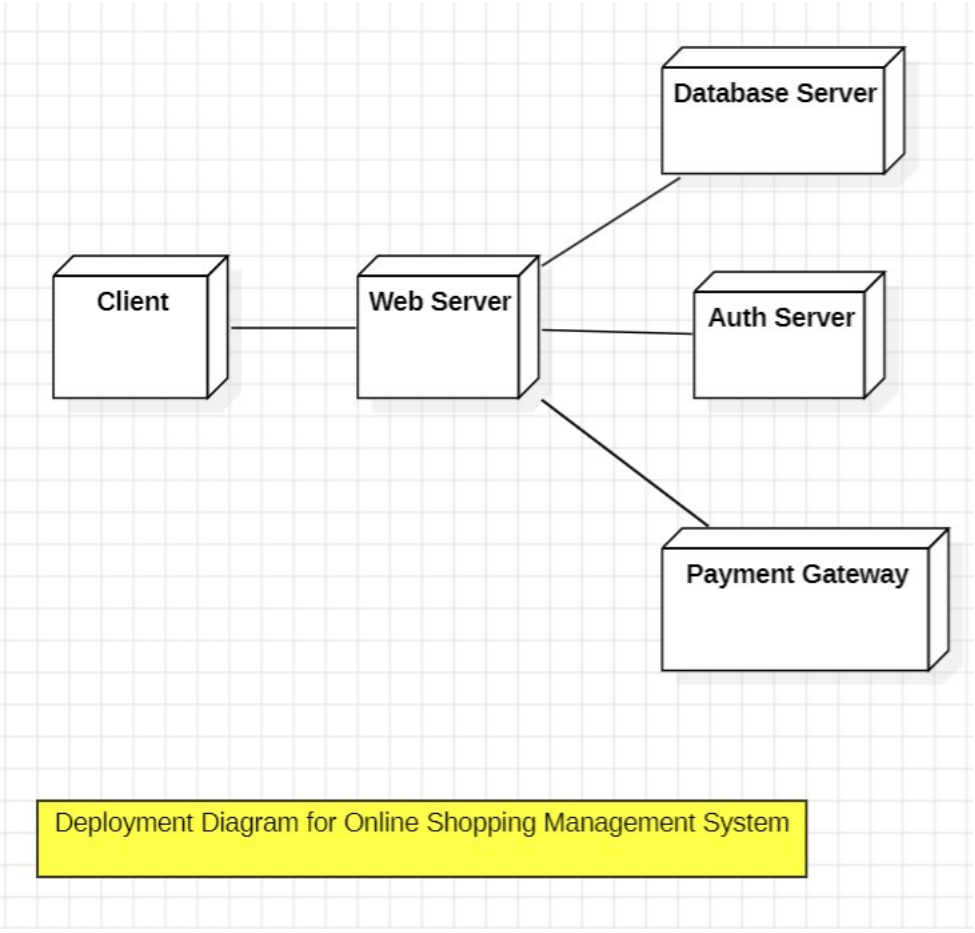


Sequence Diagram for Online Shopping Management System

1.d) OBJECT DIAGRAM:

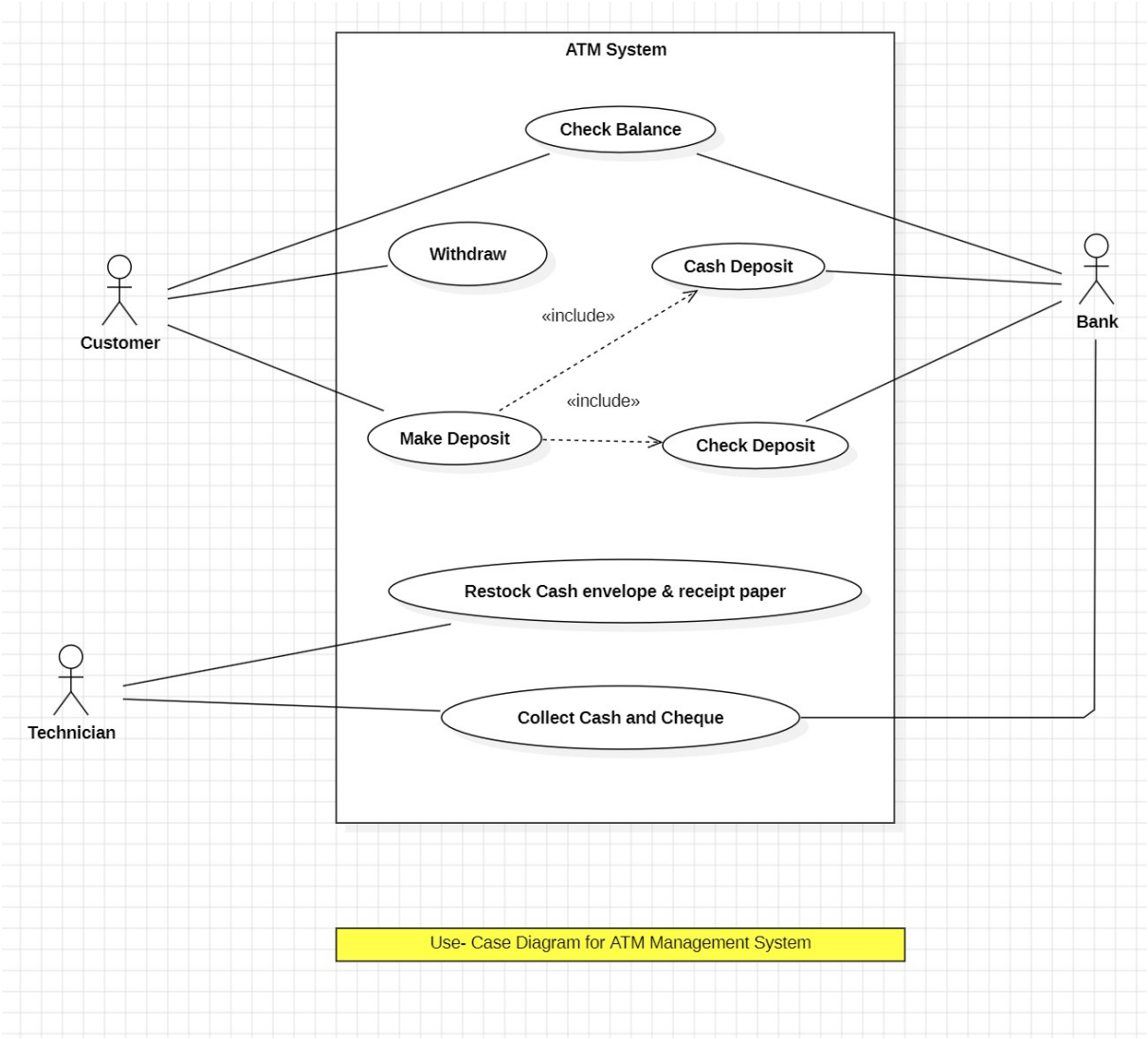


1.e) DEPLOYMENT DIAGRAM:



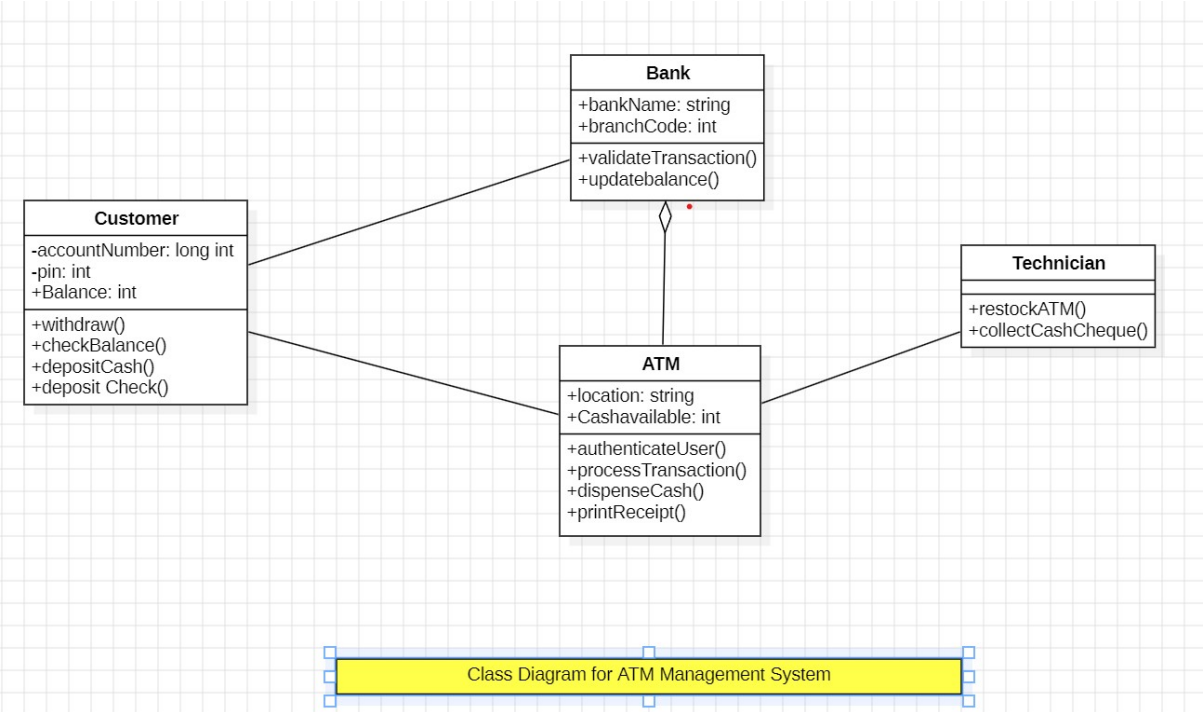
2. ATM MANAGEMENT SYSTEM

2.a) USE CASE DIAGRAM:

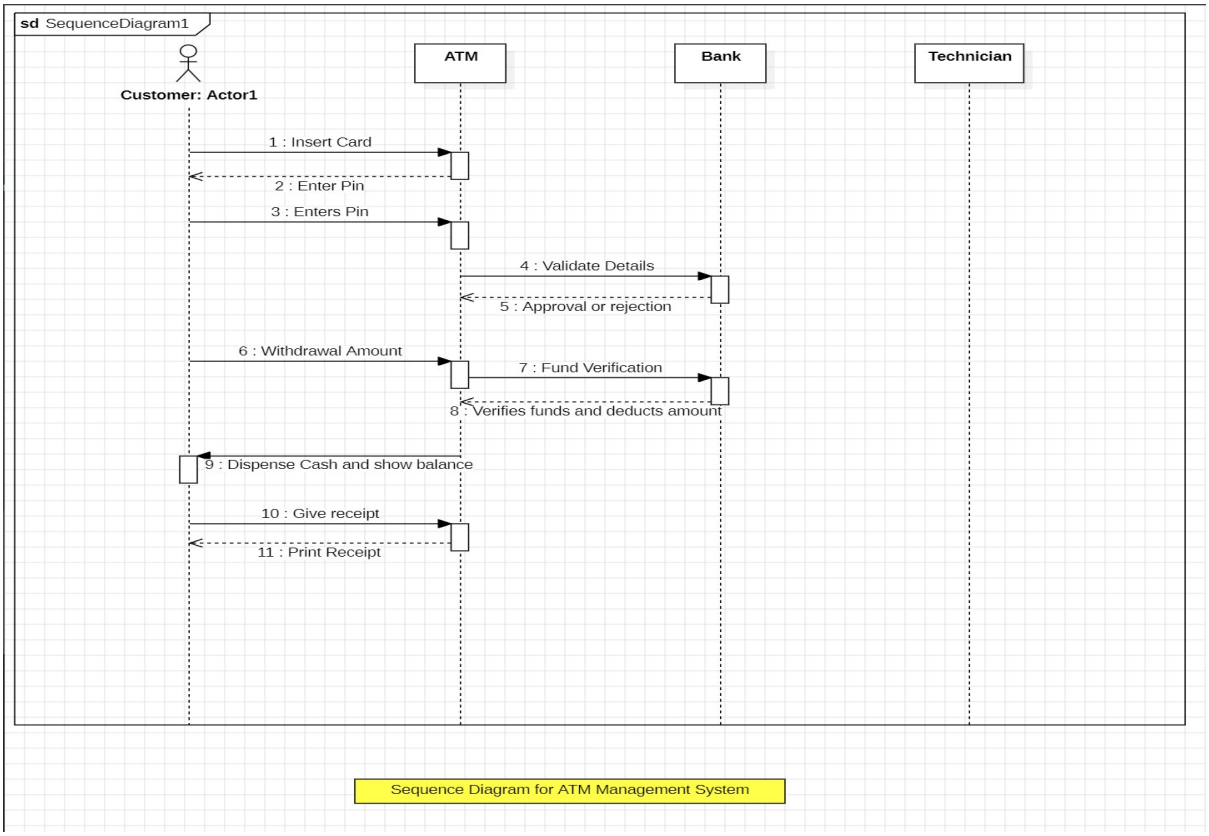


Use- Case Diagram for ATM Management System

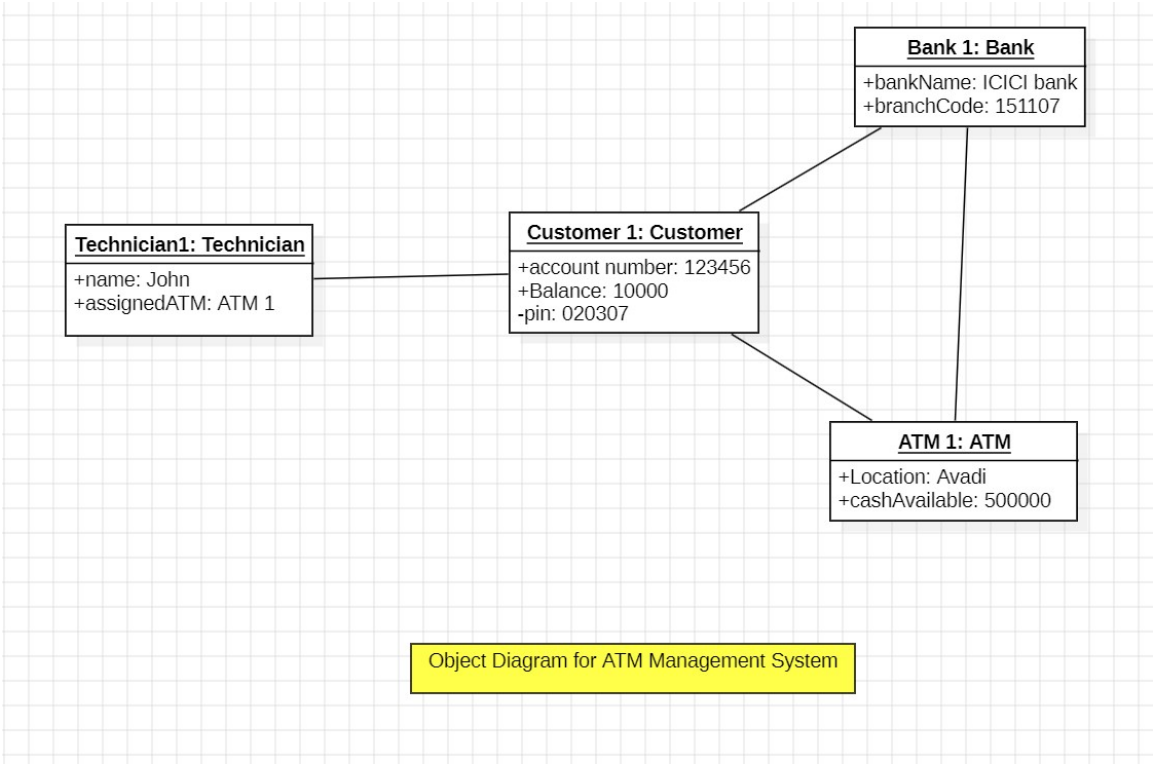
2.b) CLASS DIAGRAM:



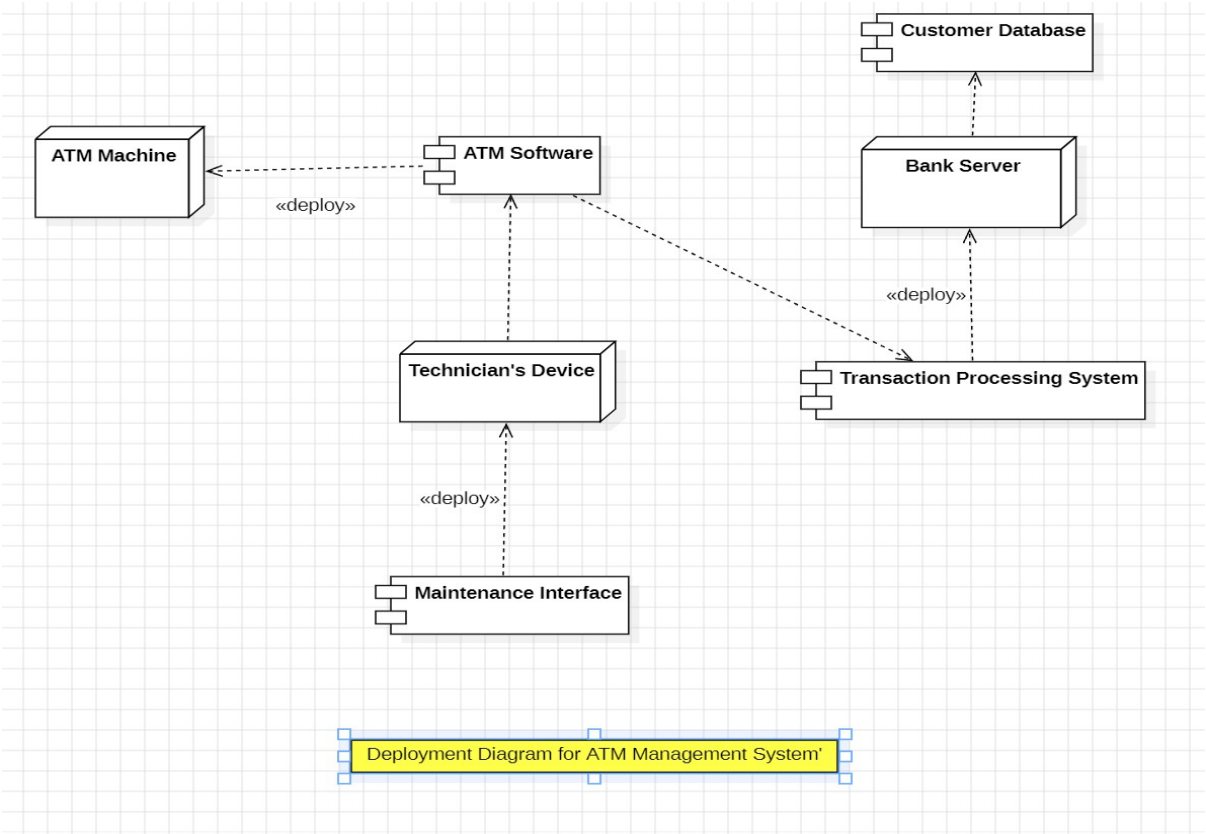
2. c) SEQUENCE DIAGRAM:



3. d) OBJECT DIAGRAM:



2. e) DEPLOYMENT DIAGRAM:



JAVA PROGRAMS

3.Basic Java Programs

3.a) Cash- Withdrawal System:

CODE:

```java
import java.util.Scanner;

public class ATMWithdrawal {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter balance: ");
        double balance = scanner.nextDouble();

        System.out.print("Enter withdrawal amount: ");
        double amount = scanner.nextDouble();

        if (amount > balance) {
            System.out.println("Insufficient balance!");
        } else if (amount <= 0) {
            System.out.println("Invalid amount entered!");
        } else {
            balance -= amount;
            System.out.println("Withdrawal successful! Remaining balance: " + balance);
        }
        scanner.close();
    }
}
```

OUTPUT:

```
D:\AVV CHENNAI\Object Oriented Programming\JAVA>javac ATMWithdrawal.java

D:\AVV CHENNAI\Object Oriented Programming\JAVA>java ATMWithdrawal
Enter balance: 10000
Enter withdrawal amount: 1000
Withdrawal successful! Remaining balance: 9000.0
```

3.b) Odd or Even:

CODE:

```java
import java.util.Scanner;

public class EvenOdd {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a number: ");
        int num = scanner.nextInt();

        if (num % 2 == 0) {
            System.out.println(num + " is Even.");
        } else {
            System.out.println(num + " is Odd.");
        }
        scanner.close();
    }
}
```

OUTPUT:

```
D:\AVV CHENNAI\Object Oriented Programming\JAVA>java EvenOdd
Enter a number: 32
32 is Even.
```

3.c) Largest Number:

CODE:

```java
import java.util.Scanner;

public class LargestNumber {
    Run | Debug
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print(s:"Enter three numbers: ");
        int a = scanner.nextInt();
        int b = scanner.nextInt();
        int c = scanner.nextInt();

        if (a >= b && a >= c) {
            System.out.println(a + " is the largest.");
        } else if (b >= a && b >= c) {
            System.out.println(b + " is the largest.");
        } else {
            System.out.println(c + " is the largest.");
        }
        scanner.close();
    }
}
```

OUTPUT:

```
D:\AVV CHENNAI\Object Oriented Programming\JAVA>java LargestNumber
Enter three numbers: 32
45
18
45 is the largest.
```

3.d) Leap Year Checker:

CODE:

```java
import java.util.Scanner;

public class LeapYearCheck {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a year: ");
        int year = scanner.nextInt();

        if ((year % 4 == 0 && year % 100 != 0) || (year % 400 == 0)) {
            System.out.println(year + " is a Leap Year.");
        } else {
            System.out.println(year + " is not a Leap Year.");
        }
        scanner.close();
    }
}
```

OUTPUT:

```
D:\AVV CHENNAI\Object Oriented Programming\JAVA>java LeapYearCheck
Enter a year: 2024
2024 is a Leap Year.
```

3.e) Number Checker:

CODE:

```java
import java.util.Scanner;

public class NumberCheck {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a number: ");
        int num = scanner.nextInt();

        if (num > 0) {
            System.out.println("The number is Positive.");
        } else if (num < 0) {
            System.out.println("The number is Negative.");
        } else {
            System.out.println("The number is Zero.");
        }
        scanner.close();
    }
}
```

OUTPUT:

```
D:\AVV CHENNAI\Object Oriented Programming\JAVA>java NumberCheck
Enter a number: -45
The number is Negative.
```

3.f) Quadratic Equation:

CODE:

```java
import java.util.Scanner;

public class QuadraticEquation {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter coefficients a, b, and c: ");
        double a = scanner.nextDouble();
        double b = scanner.nextDouble();
        double c = scanner.nextDouble();

        double discriminant = b * b - 4 * a * c;

        if (discriminant > 0) {
            double root1 = (-b + Math.sqrt(discriminant)) / (2 * a);
            double root2 = (-b - Math.sqrt(discriminant)) / (2 * a);
            System.out.println("Roots are real and different: " + root1 + ", " + root2);
        } else if (discriminant == 0) {
            double root = -b / (2 * a);
            System.out.println("Roots are real and equal: " + root);
        } else {
            System.out.println("Roots are imaginary.");
        }
        scanner.close();
    }
}
```

OUTPUT:

```
D:\AVV CHENNAI\Object Oriented Programming\JAVA>javac QuadraticEquation.java

D:\AVV CHENNAI\Object Oriented Programming\JAVA>java QuadraticEquation
Enter coefficients a, b, and c: 1
5
6
Roots are real and different: -2.0, -3.0
```

3.g) Student Grades:

CODE:

```java
import java.util.Scanner;

public class StudentGrade {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter marks: ");
        int marks = scanner.nextInt();

        if (marks >= 90) {
            System.out.println("Grade: A");
        } else if (marks >= 80) {
            System.out.println("Grade: B");
        } else if (marks >= 70) {
            System.out.println("Grade: C");
        } else if (marks >= 60) {
            System.out.println("Grade: D");
        } else {
            System.out.println("Grade: F (Fail)");
        }
        scanner.close();
    }
}
```

OUTPUT:

```
D:\AVV CHENNAI\Object Oriented Programming\JAVA>javac StudentGrade.java

D:\AVV CHENNAI\Object Oriented Programming\JAVA>java StudentGrade
Enter marks: 97
Grade: A
```

3.h) Triangle Type:

CODE:

```java
import java.util.Scanner;

public class TriangleType {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter three sides of the triangle: ");
        int a = scanner.nextInt();
        int b = scanner.nextInt();
        int c = scanner.nextInt();

        if (a == b && b == c) {
            System.out.println("It is an Equilateral Triangle.");
        } else if (a == b || b == c || a == c) {
            System.out.println("It is an Isosceles Triangle.");
        } else {
            System.out.println("It is a Scalene Triangle.");
        }
        scanner.close();
    }
}
```

OUTPUT:

```
D:\AVV CHENNAI\Object Oriented Programming\JAVA>javac TriangleType.java

D:\AVV CHENNAI\Object Oriented Programming\JAVA>java TriangleType
Enter three sides of the triangle: 5
5
5
It is an Equilateral Triangle.
```

3.i) Voting Eligibility:

CODE:

```java
import java.util.Scanner;

public class VotingEligibility {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter your age: ");
        int age = scanner.nextInt();

        if (age >= 18) {
            System.out.println("You are eligible to vote.");
        } else {
            System.out.println("You are not eligible to vote.");
        }
        scanner.close();
    }
}
```

OUTPUT:

```
D:\AVV CHENNAI\Object Oriented Programming\JAVA>javac VotingEligibility.java

D:\AVV CHENNAI\Object Oriented Programming\JAVA>java VotingEligibility
Enter your age: 21
You are eligible to vote.
```

3.j) Vowels and Consonants Classifier:

CODE:

```java
import java.util.Scanner;

public class VowelConsonant {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a character: ");
        char ch = scanner.next().toLowerCase().charAt(0);

        if (ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u') {
            System.out.println(ch + " is a Vowel.");
        } else if ((ch >= 'a' && ch <= 'z')) {
            System.out.println(ch + " is a Consonant.");
        } else {
            System.out.println("Invalid input! Please enter a letter.");
        }
        scanner.close();
    }
}
```

OUTPUT:

```
D:\AVV CHENNAI\Object Oriented Programming\JAVA>javac VowelConsonant.java

D:\AVV CHENNAI\Object Oriented Programming\JAVA>java VowelConsonant
Enter a character: u
u is a Vowel.
```

4.SINGLE INHERITANCE PROGRAMS

4.a) Book Library:

CODE:

```
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.ArrayList;

class Book {
    protected String title;
    protected String author;
    protected String isbn;
    protected Date publishDate;
    protected boolean isAvailable;
    protected ArrayList<String> borrowerHistory;

    public Book(String title, String author, String isbn, Date publishDate) {
        this.title = title;
        this.author = author;
        this.isbn = isbn;
        this.publishDate = publishDate;
        this.isAvailable = true;
        this.borrowerHistory = new ArrayList<>();
    }

    public void checkOut(String borrowerName) {
        if (isAvailable) {
            isAvailable = false;
            borrowerHistory.add(borrowerName + " (" + new Date() + ")");
            System.out.println(title + " has been checked out by " + borrowerName);
        } else {
            System.out.println(title + " is not available.");
        }
    }

    public void returnBook() {
        isAvailable = true;
        System.out.println(title + " has been returned.");
    }

    public void displayDetails() {
        System.out.println("\n=== Book Details ===");
        System.out.println("Title: " + title);
        System.out.println("Author: " + author);
        System.out.println("ISBN: " + isbn);
        System.out.println("Published: " + publishDate);
        System.out.println("Status: " + (isAvailable ? "Available" : "Checked Out"));
    }

    public void showBorrowerHistory() {
        System.out.println("\nBorrower History for " + title + ":");
        if (borrowerHistory.isEmpty()) {
            System.out.println("No borrowing history available.");
        } else {
            for (String record : borrowerHistory) {
                System.out.println("- " + record);
            }
        }
    }
    public String getTitle() { return title; }
    public String getAuthor() { return author; }
    public String getIsbn() { return isbn; }
}
class Textbook extends Book {
    private String subject;
```

```
    private int edition;
    private boolean hasSolutionsManual;
    private boolean isReserved;

    public Textbook(String title, String author, String isbn, Date publishDate,
                    String subject, int edition, boolean hasSolutionsManual) {
        super(title, author, isbn, publishDate);
        this.subject = subject;
        this.edition = edition;
        this.hasSolutionsManual = hasSolutionsManual;
        this.isReserved = false;
    }

    public void requestSolutionsManual() {
        if (hasSolutionsManual) {
            System.out.println("Solutions manual for " + title + " is available.");
        } else {
            System.out.println("No solutions manual available for this textbook.");
        }
    }

    public void reserveForClass(String className) {
        isReserved = true;
        System.out.println(title + " reserved for " + className);
    }

    @Override
    public void checkOut(String borrowerName) {
        if (isReserved) {
            System.out.println(title + " is reserved and cannot be checked out.");
            return;
        }
        super.checkOut(borrowerName);
    }

    @Override
    public void displayDetails() {
        super.displayDetails();
        System.out.println("Type: Textbook");
        System.out.println("Subject: " + subject);
        System.out.println("Edition: " + edition);
        System.out.println("Solutions Manual: " + (hasSolutionsManual ? "Yes" : "No"));
        System.out.println("Reserved: " + (isReserved ? "Yes" : "No"));
    }

    public void updateEdition(int newEdition) {
        this.edition = newEdition;
        System.out.println(title + " updated to edition " + newEdition);
    }
}
public class BookLibrary {
    public static void main(String[] args) throws Exception {
        SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
        Date pubDate = sdf.parse("2020-01-15");
        Textbook calculus = new Textbook("Calculus: Early Transcendentals",
                                         "James Stewart",
                                         "978-1305270339",
                                         pubDate,
                                         "Mathematics",
                                         8,
                                         true);
        calculus.displayDetails();
        calculus.checkOut("John Smith");
        calculus.checkOut("Alice Johnson");
        calculus.returnBook();
        calculus.requestSolutionsManual();
        calculus.updateEdition(9);
        calculus.reserveForClass("Math 101");
        calculus.checkOut("Bob Brown");
        calculus.showBorrowerHistory();
        Date novelDate = sdf.parse("2019-05-21");
        Book novel = new Book("The Testaments", "Margaret Atwood", "978-0385543781", novelDate);
```

```
        novel.displayDetails();
        novel.checkOut("Emma Wilson");
        novel.checkOut("David Lee");
        novel.returnBook();
        novel.checkOut("David Lee");
        novel.showBorrowerHistory();
    }
}
```

OUTPUT:

```
D:\AVV CHENNAI\Object Oriented Programming\JAVA\Inheritance\Single Inheritance>javac BookLibrary.java

D:\AVV CHENNAI\Object Oriented Programming\JAVA\Inheritance\Single Inheritance>java BookLibrary.java

=== Book Details ===
Title: Calculus: Early Transcendentals
Author: James Stewart
ISBN: 978-1305270339
Published: Wed Jan 15 00:00:00 IST 2020
Status: Available
Type: Textbook
Subject: Mathematics
Edition: 8
Solutions Manual: Yes
Reserved: No
Calculus: Early Transcendentals has been checked out by John Smith
Calculus: Early Transcendentals is not available.
Calculus: Early Transcendentals has been returned.
Solutions manual for Calculus: Early Transcendentals is available.
Calculus: Early Transcendentals updated to edition 9
Calculus: Early Transcendentals reserved for Math 101
Calculus: Early Transcendentals is reserved and cannot be checked out.

Borrower History for Calculus: Early Transcendentals:
- John Smith (Fri Apr 04 22:41:39 IST 2025)

=== Book Details ===
Title: The Testaments
Author: Margaret Atwood
ISBN: 978-0385543781
Published: Tue May 21 00:00:00 IST 2019
Status: Available
The Testaments has been checked out by Emma Wilson
The Testaments is not available.
The Testaments has been returned.
The Testaments has been checked out by David Lee

Borrower History for The Testaments:
- Emma Wilson (Fri Apr 04 22:41:39 IST 2025)
- David Lee (Fri Apr 04 22:41:39 IST 2025)
```

4.b) Electronic Devices:

CODE:

```
import java.util.ArrayList;
import java.util.Date;

class Device {
    protected String model;
    protected String manufacturer;
    protected String serialNumber;
```

```
    protected Date productionDate;
    protected boolean isPoweredOn;
    protected ArrayList<String> operationLog;

    public Device(String model, String manufacturer, String serialNumber, Date productionDate) {
        this.model = model;
        this.manufacturer = manufacturer;
        this.serialNumber = serialNumber;
        this.productionDate = productionDate;
        this.isPoweredOn = false;
        this.operationLog = new ArrayList<>();
        logOperation("Device manufactured");
    }

    protected void logOperation(String operation) {
        String logEntry = new Date() + " - " + operation;
        operationLog.add(logEntry);
    }

    public void powerOn() {
        if (!isPoweredOn) {
            isPoweredOn = true;
            logOperation("Device powered on");
            System.out.println(model + " powered on");
        } else {
            System.out.println(model + " is already on");
        }
    }

    public void powerOff() {
        if (isPoweredOn) {
            isPoweredOn = false;
            logOperation("Device powered off");
            System.out.println(model + " powered off");
        } else {
            System.out.println(model + " is already off");
        }
    }

    public void displayDeviceInfo() {
        System.out.println("\n=== Device Information ===");
        System.out.println("Model: " + model);
        System.out.println("Manufacturer: " + manufacturer);
        System.out.println("Serial: " + serialNumber);
        System.out.println("Production Date: " + productionDate);
        System.out.println("Status: " + (isPoweredOn ? "ON" : "OFF"));
    }

    public void showOperationHistory() {
        System.out.println("\nOperation History for " + model + ":");
        for (String log : operationLog) {
            System.out.println("- " + log);
        }
    }
}

class Smartphone extends Device {
    private String osVersion;
    private String imei;
    private boolean isLocked;
    private ArrayList<String> installedApps;

    public Smartphone(String model, String manufacturer, String serialNumber,
                      Date productionDate, String osVersion, String imei) {
        super(model, manufacturer, serialNumber, productionDate);
        this.osVersion = osVersion;
        this.imei = imei;
        this.isLocked = true;
        this.installedApps = new ArrayList<>();
        logOperation("Smartphone initialized");
    }

    public void unlock(String method) {
```

```
        if (isPoweredOn) {
            isLocked = false;
            logOperation("Unlocked via " + method);
            System.out.println(model + " unlocked via " + method);
        } else {
            System.out.println("Cannot unlock - device is powered off");
        }
    }

    public void lock() {
        isLocked = true;
        logOperation("Device locked");
        System.out.println(model + " locked");
    }

    public void installApp(String appName) {
        if (isPoweredOn) {
            installedApps.add(appName);
            logOperation("App installed: " + appName);
            System.out.println(appName + " installed successfully");
        } else {
            System.out.println("Cannot install app - device is powered off");
        }
    }

    public void uninstallApp(String appName) {
        if (installedApps.remove(appName)) {
            logOperation("App uninstalled: " + appName);
            System.out.println(appName + " uninstalled successfully");
        } else {
            System.out.println(appName + " not found in installed apps");
        }
    }

    public void listInstalledApps() {
        System.out.println("\nInstalled Apps on " + model + ":");
        if (installedApps.isEmpty()) {
            System.out.println("No apps installed");
        } else {
            for (String app : installedApps) {
                System.out.println("- " + app);
            }
        }
    }

    public void makeCall(String number) {
        if (isPoweredOn && !isLocked) {
            logOperation("Call made to: " + number);
            System.out.println("Calling " + number + "...");
        } else if (!isPoweredOn) {
            System.out.println("Cannot make call - device is powered off");
        } else {
            System.out.println("Cannot make call - device is locked");
        }
    }

    @Override
    public void displayDeviceInfo() {
        super.displayDeviceInfo();
        System.out.println("Type: Smartphone");
        System.out.println("OS Version: " + osVersion);
        System.out.println("IMEI: " + imei);
        System.out.println("Lock Status: " + (isLocked ? "LOCKED" : "UNLOCKED"));
        System.out.println("Installed Apps: " + installedApps.size());
    }
}

public class Electronic_Devices {
    public static void main(String[] args) {
        Smartphone phone = new Smartphone("Galaxy S23", "Samsung", "SN123456",
                                          new Date(), "Android 13", "IMEI123456789");

        phone.displayDeviceInfo();
```

```
        phone.powerOn();
        phone.unlock("fingerprint");
        phone.installApp("WhatsApp");
        phone.installApp("Google Maps");
        phone.makeCall("1234567890");
        phone.listInstalledApps();
        phone.uninstallApp("Google Maps");
        phone.lock();
        phone.powerOff();
        phone.showOperationHistory();
    }
}
```

OUTPUT:

```
D:\AVV CHENNAI\Object Oriented Programming\JAVA\Inheritance\Single Inheritance>javac Electronic_Devices.java

D:\AVV CHENNAI\Object Oriented Programming\JAVA\Inheritance\Single Inheritance>java Electronic_Devices.java

=== Device Information ===
Model: Galaxy S23
Manufacturer: Samsung
Serial: SN123456
Production Date: Fri Apr 04 22:51:37 IST 2025
Status: OFF
Type: Smartphone
OS Version: Android 13
IMEI: IMEI123456789
Lock Status: LOCKED
Installed Apps: 0
Galaxy S23 powered on
Galaxy S23 unlocked via fingerprint
WhatsApp installed successfully
Google Maps installed successfully
Calling 1234567890...

Installed Apps on Galaxy S23:
- WhatsApp
- Google Maps
Google Maps uninstalled successfully
Galaxy S23 locked
Galaxy S23 powered off

Operation History for Galaxy S23:
- Fri Apr 04 22:51:37 IST 2025 - Device manufactured
- Fri Apr 04 22:51:37 IST 2025 - Smartphone initialized
- Fri Apr 04 22:51:37 IST 2025 - Device powered on
- Fri Apr 04 22:51:37 IST 2025 - Unlocked via fingerprint
- Fri Apr 04 22:51:37 IST 2025 - App installed: WhatsApp
- Fri Apr 04 22:51:37 IST 2025 - App installed: Google Maps
- Fri Apr 04 22:51:37 IST 2025 - Call made to: 1234567890
- Fri Apr 04 22:51:37 IST 2025 - App uninstalled: Google Maps
- Fri Apr 04 22:51:37 IST 2025 - Device locked
- Fri Apr 04 22:51:37 IST 2025 - Device powered off
```

5. MULTILEVEL INHERITANCE PROGRAMS

5. a) Hospital Management:

CODE:

```
class Person {
    String name = "Dr. Smith";

    void showName() {
        System.out.println("Doctor Name: " + name);
    }
}
class Doctor extends Person {
    void diagnose() {
        System.out.println(name + " is diagnosing a patient.");
    }
}
class Surgeon extends Doctor {
    void performSurgery() {
        System.out.println(name + " is performing surgery.");
    }
}
public class HospitalManagement {
    public static void main(String[] args) {
        Surgeon surgeon = new Surgeon();

        surgeon.showName();
        surgeon.diagnose();
        surgeon.performSurgery();
    }
}
```

OUTPUT:

```
D:\AVV CHENNAI\Object Oriented Programming\JAVA\Inheritance\Multilevel Inheritance>javac HospitalManagement.java

D:\AVV CHENNAI\Object Oriented Programming\JAVA\Inheritance\Multilevel Inheritance>java HospitalManagement.java
Doctor Name: Dr. Smith
Dr. Smith is diagnosing a patient.
Dr. Smith is performing surgery.
```

5.b) School Management:

CODE:

```
class Person {
    String name = "Alice";

    void showName() {
        System.out.println("Name: " + name);
    }
}
class Teacher extends Person {
    String subject = "Mathematics";
    void teach() {
        System.out.println(name + " teaches " + subject);
    }
}
class Principal extends Teacher {
    void manageSchool() {
        System.out.println(name + " is managing the school.");
    }
}
public class SchoolManagement {
    public static void main(String[] args) {
        Principal principal = new Principal();

        principal.showName();
        principal.teach();
        principal.manageSchool();
    }
}
```

OUTPUT:

```
D:\AVV CHENNAI\Object Oriented Programming\JAVA\Inheritance\Multilevel Inheritance>javac SchoolManagement.java

D:\AVV CHENNAI\Object Oriented Programming\JAVA\Inheritance\Multilevel Inheritance>java SchoolManagement.java
Name: Alice
Alice teaches Mathematics
Alice is managing the school.
```

6. HIERARCHICAL INHERITANCE PROGRAMS

6.a) Employee Payroll:

CODE:

```
class Employee {
    String name = "John Doe";
    double salary = 40000;

    void displayInfo() {
        System.out.println("Employee Name: " + name);
        System.out.println("Base Salary: $" + salary);
    }
}
class FullTimeEmployee extends Employee {
    void calculateBonus() {
        double bonus = salary * 0.10;
        System.out.println("Full-Time Employee Bonus: $" + bonus);
    }
}
class PartTimeEmployee extends Employee {
    void calculateHourlyPay(int hours) {
        double hourlyRate = salary / 160; // Assuming 160 working hours in a month
        System.out.println("Part-Time Employee Pay for " + hours + " hours: $" + (hourlyRate * hours));
    }
}
public class EmployeePayroll {
    public static void main(String[] args) {
        FullTimeEmployee fullTime = new FullTimeEmployee();
        fullTime.displayInfo();
        fullTime.calculateBonus();
        PartTimeEmployee partTime = new PartTimeEmployee();
        partTime.displayInfo();
        partTime.calculateHourlyPay(20);
    }
}
```

OUTPUT:

```
D:\AVV CHENNAI\Object Oriented Programming\JAVA\Inheritance\Hierarchial Inheritance>javac EmployeePayroll.java

D:\AVV CHENNAI\Object Oriented Programming\JAVA\Inheritance\Hierarchial Inheritance>java EmployeePayroll.java
Employee Name: John Doe
Base Salary: $40000.0
Full-Time Employee Bonus: $4000.0
Employee Name: John Doe
Base Salary: $40000.0
Part-Time Employee Pay for 20 hours: $5000.0
```

6. b) Food Delivery System:

CODE:

```
class Food {
    String name = "Generic Food";

    void prepare() {
        System.out.println(name + " is being prepared.");
    }
}

class Pizza extends Food {
    void addToppings() {
        System.out.println("Adding cheese and pepperoni.");
    }
}
class Burger extends Food {
    void addSauce() {
        System.out.println("Adding ketchup and mustard.");
    }
}
public class FoodDelivery {
    public static void main(String[] args) {
        Pizza pizza = new Pizza();
        pizza.name = "Pepperoni Pizza";
        pizza.prepare();
        pizza.addToppings();
        Burger burger = new Burger();
        burger.name = "Cheese Burger";
        burger.prepare();
        burger.addSauce();
    }
}
```

OUTPUT:

```
D:\AVV CHENNAI\Object Oriented Programming\JAVA\Inheritance\Hierarchial Inheritance>javac FoodDelivery.java

D:\AVV CHENNAI\Object Oriented Programming\JAVA\Inheritance\Hierarchial Inheritance>java FoodDelivery.java
Pepperoni Pizza is being prepared.
Adding cheese and pepperoni.
Cheese Burger is being prepared.
Adding ketchup and mustard.
```

7. HYBRID INHERITANCE PROGRAMS

7.a) Shape System:

CODE:

```
import java.util.Scanner;

public class ShapeSystem {
    public static void main(String[] args) {
        Circle circle = new Circle();
        circle.getColor();
        circle.getArea();
        circle.getRadius();
    }
}
class Shape {
    String color;
    void getColor() {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter shape color: ");
        color = scanner.nextLine();
    }
}
class ColorInfo {
    String hexCode;
    void getHexCode() {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter color hex code: ");
        hexCode = scanner.nextLine();
    }
}
class TwoDShape extends Shape {
    double area;
    void getArea() {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter area: ");
        area = scanner.nextDouble();
    }
}
class Circle extends TwoDShape {
    double radius;
    void getRadius() {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter radius: ");
        radius = scanner.nextDouble();
    }
}
```

OUTPUT:

```
D:\AVV CHENNAI\Object Oriented Programming\JAVA\Inheritance\Hybrid Inheritance>javac ShapeSystem.java

D:\AVV CHENNAI\Object Oriented Programming\JAVA\Inheritance\Hybrid Inheritance>java ShapeSystem.java
Enter shape color: Blue
Enter area: 314.00
Enter radius: 10
```

7. b) University System:

CODE:

```
import java.util.Scanner;

public class UniversitySystem {
    public static void main(String[] args) {
        Undergraduate undergrad = new Undergraduate();
        undergrad.getName();
        undergrad.getStudentID();
        undergrad.getMajor();
    }
}
class Person {
    String name;
    void getName() {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter person name: ");
        name = scanner.nextLine();
    }
}
class Course {
    String courseName;
    void getCourseName() {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter course name: ");
        courseName = scanner.nextLine();
    }
}
class Student extends Person {
    int studentID;
    void getStudentID() {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter student ID: ");
        studentID = scanner.nextInt();
    }
}
class Undergraduate extends Student {
    String major;
    void getMajor() {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter major: ");
        major = scanner.nextLine();
    }
}
```

OUTPUT:

```
D:\AVV CHENNAI\Object Oriented Programming\JAVA\Inheritance\Hybrid Inheritance>javac UniversitySystem.java

D:\AVV CHENNAI\Object Oriented Programming\JAVA\Inheritance\Hybrid Inheritance>java UniversitySystem.java
Enter person name: Santhosh
Enter student ID: 24142
Enter major: Computer Science
```

8. CONSTRUCTOR PROGRAMS

8.a) Car Information:

CODE:

```
class Car {
    private String model;
    private int year;

    public Car(String model, int year) {
        this.model = model;
        this.year = year;
    }

    public void displayInfo() {
        System.out.println("Model: " + model + ", Year: " + year);
    }

    public static void main(String[] args) {
        Car myCar = new Car("Toyota Camry", 2022);
        myCar.displayInfo();
    }
}
public class CarInfo {
    private String model;
    private int year;

    public void Car(String model, int year) {
        this.model = model;
        this.year = year;
    }

    public void displayInfo() {
        System.out.println("Model: " + model + ", Year: " + year);
    }

    public static void main(String[] args) {
        Car myCar = new Car("Toyota Camry", 2022);
        myCar.displayInfo();
    }
}
```

OUTPUT:

```
D:\AVV CHENNAI\Object Oriented Programming\JAVA\Polymorphism\Constructors>javac CarInfo.java

D:\AVV CHENNAI\Object Oriented Programming\JAVA\Polymorphism\Constructors>java CarInfo.java
Model: Toyota Camry, Year: 2022
```

9. CONSTRUCTOR OVERLOADING PROGRAMS

9.a) Car Specifications:

CODE:

```
class Car {
    private String make;
    private String model;
    private int year;
    private double price;

    public Car(String make, String model) {
        this.make = make;
        this.model = model;
        this.year = 2023;
        this.price = 30000.0;
    }
    public Car(String make, String model, int year) {
        this.make = make;
        this.model = model;
        this.year = year;
        this.price = 30000.0;
    }
    public Car(String make, String model, int year, double price) {
        this.make = make;
        this.model = model;
        this.year = year;
        this.price = price;
    }
    public void display() {
        System.out.println("Car: " + make + " " + model + ", Year: " + year + ", Price: " + price);
    }
    public static void main(String[] args) {
        Car car1 = new Car("Toyota", "Corolla");
        car1.display();
        Car car2 = new Car("Honda", "Civic", 2022);
        car2.display();
        Car car3 = new Car("BMW", "X5", 2021, 50000.0);
        car3.display();
    }
}
```

OUTPUT:

```
D:\AVV CHENNAI\Object Oriented Programming\JAVA\Polymorphism\Constructor Overloading>javac Car.java

D:\AVV CHENNAI\Object Oriented Programming\JAVA\Polymorphism\Constructor Overloading>java Car.java
Car: Toyota Corolla, Year: 2023, Price: 30000.0
Car: Honda Civic, Year: 2022, Price: 30000.0
Car: BMW X5, Year: 2021, Price: 50000.0
```

10. METHOD OVERLOADING PROGRAMS

10. a) Banking App

CODE:

```
class Bank {
    void checkBalance(int accNumber) {
        System.out.println("Checking balance for Account: " + accNumber);
    }

    void checkBalance(int accNumber, int pin) {
        System.out.println("Checking balance for Account: " + accNumber + " with PIN verification.");
    }
    void checkBalance(int accNumber, int pin, int otp) {
        System.out.println("Checking balance for Account: " + accNumber + " with PIN and OTP verification.");
    }
}
public class BankingApp {
    public static void main(String[] args) {
        Bank bank = new Bank();
        bank.checkBalance(123456);
        bank.checkBalance(123456, 1234);
        bank.checkBalance(123456, 1234, 567890);
    }
}
```

OUTPUT:

```
D:\AVV CHENNAI\Object Oriented Programming\JAVA\Polymorphism\Method Overloading>javac BankingApp.java

D:\AVV CHENNAI\Object Oriented Programming\JAVA\Polymorphism\Method Overloading>java BankingApp.java
Checking balance for Account: 123456
Checking balance for Account: 123456 with PIN verification.
Checking balance for Account: 123456 with PIN and OTP verification.
```

10.b) Ride App:

CODE:

```
class RideBooking {
    void bookRide(String destination) {
        System.out.println("Booking ride to " + destination);
    }

    void bookRide(String pickup, String destination) {
        System.out.println("Booking ride from " + pickup + " to " + destination);
    }
    void bookRide(String pickup, String destination, String type) {
        System.out.println("Booking " + type + " ride from " + pickup + " to " + destination);
    }
}
public class RideApp {
    public static void main(String[] args) {
        RideBooking ride = new RideBooking();
        ride.bookRide("Downtown");
        ride.bookRide("Home", "Downtown");
        ride.bookRide("Home", "Downtown", "Luxury");
    }
}
```

OUTPUT:

```
D:\AVV CHENNAI\Object Oriented Programming\JAVA\Polymorphism\Method Overloading>javac RideApp.java

D:\AVV CHENNAI\Object Oriented Programming\JAVA\Polymorphism\Method Overloading>java RideApp.java
Booking ride to Downtown
Booking ride from Home to Downtown
Booking Luxury ride from Home to Downtown
```

11. METHOD OVERRIDING PROGRAMS

11.a) ECommerce App:

CODE:

```
class Product {
    protected String name;
    protected double price;

    public Product(String name, double price) {
        this.name = name;
        this.price = price;
    }

    public void displayDetails() {
        System.out.println(name + " - $" + price);
    }
}

class Electronics extends Product {
    public Electronics(String name, double price) {
        super(name, price);
    }

    @Override
    public void displayDetails() {
        super.displayDetails();
        System.out.println("Type: Electronic");
    }
}
class Grocery extends Product {
    public Grocery(String name, double price) {
        super(name, price);
    }

    @Override
    public void displayDetails() {
        super.displayDetails();
        System.out.println("Type: Grocery");
    }
}
public class ECommerce {
    public static void main(String[] args) {
        Product laptop = new Electronics("Laptop", 999.99);
        Product milk = new Grocery("Milk", 3.99);

        System.out.println("=== Products ===");
        laptop.displayDetails();
        System.out.println();
        milk.displayDetails();
    }
}
```

OUTPUT:

```
D:\AVV CHENNAI\Object Oriented Programming\JAVA\Polymorphism\Method Overriding>javac ECommerce.java

D:\AVV CHENNAI\Object Oriented Programming\JAVA\Polymorphism\Method Overriding>java ECommerce.java
=== Products ===
Laptop - $999.99
Type: Electronic

Milk - $3.99
Type: Grocery
```

11. b) Restaurant System

CODE:

```
class MenuItem {
    protected String name;
    protected double price;

    public MenuItem(String name, double price) {
        this.name = name;
        this.price = price;
    }

    public void prepare() {
        System.out.println("Preparing " + name);
    }

    public void serve() {
        System.out.println("Serving " + name + " ($" + price + ")");
    }
}

class Pizza extends MenuItem {
    private String crustType;

    public Pizza(String name, double price, String crustType) {
        super(name, price);
        this.crustType = crustType;
    }

    @Override
    public void prepare() {
        System.out.println("Making " + crustType + " crust pizza: " + name);
        addToppings();
        bake();
    }

    private void bake() {
        System.out.println("Baking at 400°F for 15 minutes");
    }

    private void addToppings() {
        System.out.println("Adding toppings...");
    }
}
class Drink extends MenuItem {
    private boolean isAlcoholic;

    public Drink(String name, double price, boolean isAlcoholic) {
        super(name, price);
        this.isAlcoholic = isAlcoholic;
    }

    @Override
    public void serve() {
        super.serve();
        if (isAlcoholic) {
            System.out.println("Checking ID...");
        }
        System.out.println("Adding straw");
    }
}
public class RestaurantSystem {
    public static void main(String[] args) {
```

```
        MenuItem margherita = new Pizza("Margherita", 12.99, "thin");
        MenuItem cola = new Drink("Cola", 2.99, false);
        MenuItem beer = new Drink("Craft Beer", 6.99, true);

        System.out.println("=== Order Processing ===");
        processOrder(margherita);
        processOrder(cola);
        processOrder(beer);
    }

    public static void processOrder(MenuItem item) {
        item.prepare();
        item.serve();
        System.out.println("------");
    }
}
```

OUTPUT:

```
D:\AVV CHENNAI\Object Oriented Programming\JAVA\Polymorphism\Method Overriding>javac RestaurantSystem.java

D:\AVV CHENNAI\Object Oriented Programming\JAVA\Polymorphism\Method Overriding>java RestaurantSystem.java
=== Order Processing ===
Making thin crust pizza: Margherita
Adding toppings...
Baking at 400°F for 15 minutes
Serving Margherita ($12.99)
------
Preparing Cola
Serving Cola ($2.99)
Adding straw
------
Preparing Craft Beer
Serving Craft Beer ($6.99)
Checking ID...
Adding straw
------
```

12. INTERFACE PROGRAMS

12.a) Flight Reservation System

CODE:

```
interface FlightBooking {
    void bookTicket(String destination);
}

class DomesticFlight implements FlightBooking {
    public void bookTicket(String destination) {
        System.out.println("Domestic flight booked to: " + destination);
    }
}
class InternationalFlight implements FlightBooking {
    public void bookTicket(String destination) {
        System.out.println("International flight booked to: " + destination);
    }
}
public class FlightReservationSystem {
    public static void main(String[] args) {
        FlightBooking domestic = new DomesticFlight();
        FlightBooking international = new InternationalFlight();
        domestic.bookTicket("New York");
        international.bookTicket("London");
    }
}
```

OUTPUT:

```
D:\AVV CHENNAI\Object Oriented Programming\JAVA\Abstraction\Interface Programs>javac FlightReservationSystem.java

D:\AVV CHENNAI\Object Oriented Programming\JAVA\Abstraction\Interface Programs>java FlightReservationSystem.java
Domestic flight booked to: New York
International flight booked to: London
```

12. b) Notification System

CODE:

```
interface Notification {
    void sendNotification(String message);
}

class EmailNotification implements Notification {
```

```
    public void sendNotification(String message) {
        System.out.println("Email sent: " + message);
    }
}
class SMSNotification implements Notification {
    public void sendNotification(String message) {
        System.out.println("SMS sent: " + message);
    }
}
public class NotificationSystem {
    public static void main(String[] args) {
        Notification email = new EmailNotification();
        Notification sms = new SMSNotification();
        email.sendNotification("Your order has been shipped.");
        sms.sendNotification("Your OTP is 123456.");
    }
}
```

OUTPUT:

```
D:\AVV CHENNAI\Object Oriented Programming\JAVA\Abstraction\Interface Programs>javac NotificationSystem.java

D:\AVV CHENNAI\Object Oriented Programming\JAVA\Abstraction\Interface Programs>java NotificationSystem.java
Email sent: Your order has been shipped.
SMS sent: Your OTP is 123456.
```

12.c) Online Exam System

CODE:

```
interface Exam {
    void startExam();
}

class MCQExam implements Exam {
    public void startExam() {
        System.out.println("Starting Multiple Choice Questions exam...");
    }
}
class CodingExam implements Exam {
    public void startExam() {
        System.out.println("Starting Coding Exam...");
    }
}
public class OnlineExamSystem {
    public static void main(String[] args) {
        Exam mcq = new MCQExam();
        Exam coding = new CodingExam();
        mcq.startExam();
        coding.startExam();
    }
}
```

OUTPUT:

```
D:\AVV CHENNAI\Object Oriented Programming\JAVA\Abstraction\Interface Programs>javac OnlineExamSystem.java

D:\AVV CHENNAI\Object Oriented Programming\JAVA\Abstraction\Interface Programs>java OnlineExamSystem.java
Starting Multiple Choice Questions exam...
Starting Coding Exam...
```

12.d) Tax Calculating System

CODE:

```java
interface Tax {
    void calculateTax();
}

class IndividualTax implements Tax {
    public void calculateTax() {
        System.out.println("Calculating tax for an individual...");
    }
}
class BusinessTax implements Tax {
    public void calculateTax() {
        System.out.println("Calculating tax for a business...");
    }
}
public class TaxCalculationSystem {
    public static void main(String[] args) {
        Tax individual = new IndividualTax();
        Tax business = new BusinessTax();
        individual.calculateTax();
        business.calculateTax();
    }
}
```

OUTPUT:

```
D:\AVV CHENNAI\Object Oriented Programming\JAVA\Abstraction\Interface Programs>javac TaxCalculationSystem.java

D:\AVV CHENNAI\Object Oriented Programming\JAVA\Abstraction\Interface Programs>java TaxCalculationSystem.java
Calculating tax for an individual...
Calculating tax for a business...
```

13. ABSTRACT CLASS PROGRAMS

13.a) Employee Salary:

CODE:

```
abstract class Employee {
    String name;
    double salary;

    Employee(String name, double salary) {
        this.name = name;
        this.salary = salary;
    }
    abstract double getPay();

    void showInfo() {
        System.out.println(name + ": $" + getPay());
    }
}

class Manager extends Employee {
    double bonus;

    Manager(String name, double salary, double bonus) {
        super(name, salary);
        this.bonus = bonus;
    }

    @Override
    double getPay() {
        return salary + bonus;
    }
}
class Developer extends Employee {
    Developer(String name, double salary) {
        super(name, salary);
    }
    @Override
    double getPay() {
        return salary;
    }
}
public class EmployeeSalary {
    public static void main(String[] args) {
        Employee emp1 = new Manager("John", 5000, 1000);
        Employee emp2 = new Developer("Alice", 4000);

        emp1.showInfo(); // John: $6000.0
        emp2.showInfo(); // Alice: $4000.0
    }
}
```

OUTPUT:

```
D:\AVV CHENNAI\Object Oriented Programming\JAVA\Abstraction\Abstract Class>javac EmployeeSalary.java

D:\AVV CHENNAI\Object Oriented Programming\JAVA\Abstraction\Abstract Class>java EmployeeSalary.java
John: $6000.0
Alice: $4000.0
```

13.b) File Format Converter

CODE:

```
abstract class FileConverter {
    abstract void convert(String inputFile, String outputFile);
}

class TextToUppercase extends FileConverter {
    @Override
    void convert(String inputFile, String outputFile) {
        System.out.println("Converting " + inputFile + " to uppercase in " + outputFile);
        // Actual implementation would read/write files here
    }
}
class ImageResizer extends FileConverter {
    @Override
    void convert(String inputFile, String outputFile) {
        System.out.println("Resizing image " + inputFile + " to " + outputFile);
        // Actual implementation would resize image here
    }
}
public class FileFormatConverter {
    public static void main(String[] args) {
        FileConverter textConverter = new TextToUppercase();
        FileConverter imageConverter = new ImageResizer();

        textConverter.convert("input.txt", "output.txt");
        imageConverter.convert("photo.jpg", "small_photo.jpg");
    }
}
```

OUTPUT:

```
D:\AVV CHENNAI\Object Oriented Programming\JAVA\Abstraction\Abstract Class>javac FileFormatConverter.java

D:\AVV CHENNAI\Object Oriented Programming\JAVA\Abstraction\Abstract Class>java FileFormatConverter.java
Converting input.txt to uppercase in output.txt
Resizing image photo.jpg to small_photo.jpg
```

13.c) Notification System

CODE:

```
abstract class Notification {
    abstract void send(String to, String message);
}

class Email extends Notification {
    @Override
    void send(String to, String message) {
        System.out.println("Sending email to " + to + ": " + message);
    }
}
class SMS extends Notification {
    @Override
    void send(String to, String message) {
        System.out.println("Sending SMS to " + to + ": " + message);
    }
}
public class NotificationSystem {
    public static void main(String[] args) {
        Notification email = new Email();
        Notification sms = new SMS();

        email.send("user@example.com", "Hello via Email");
        sms.send("+123456789", "Hello via SMS");
    }
}
```

OUTPUT:

```
D:\AVV CHENNAI\Object Oriented Programming\JAVA\Abstraction\Abstract Class>javac NotificationSystem.java

D:\AVV CHENNAI\Object Oriented Programming\JAVA\Abstraction\Abstract Class>java NotificationSystem.java
Sending email to user@example.com: Hello via Email
Sending SMS to +123456789: Hello via SMS
```

13.d) Simple UI

CODE:

```
abstract class UIElement {
    abstract void draw();
    abstract void onClick();
}

class Button extends UIElement {
    String label;

    Button(String label) {
        this.label = label;
    }
```

```
    @Override
    void draw() {
        System.out.println("[ " + label + " ]");
    }

    @Override
    void onClick() {
        System.out.println(label + " clicked!");
    }
}
class TextBox extends UIElement {
    String text = "";

    @Override
    void draw() {
        System.out.println("|" + (text.isEmpty() ? "______" : text) + "|");
    }

    @Override
    void onClick() {
        System.out.println("TextBox selected - ready for typing");
    }
}
public class SimpleUI {
    public static void main(String[] args) {
        UIElement loginBtn = new Button("Login");
        UIElement searchBox = new TextBox();

        // Draw UI
        loginBtn.draw();
        searchBox.draw();
        loginBtn.onClick();
        searchBox.onClick();
    }
}
```

OUTPUT:

```
D:\AVV CHENNAI\Object Oriented Programming\JAVA\Abstraction\Abstract Class>java SimpleUI.java
[ Login ]
|______|
Login clicked!
TextBox selected - ready for typing
```

14. ENCAPSULATION PROGRAMS

14.a) Mobile System:

CODE:

```
class MobilePhone {
    private String brand;
    private int batteryLevel;
    private boolean isOn;

    public MobilePhone(String brand, int batteryLevel) {
        this.brand = brand;
        this.batteryLevel = batteryLevel;
        this.isOn = false;
    }
    public String getBrand() {
        return brand;
    }
    public int getBatteryLevel() {
        return batteryLevel;
    }
    public boolean isPhoneOn() {
        return isOn;
    }
    public void powerOn() {
        if (!isOn && batteryLevel > 0) {
            isOn = true;
            System.out.println(brand + " phone is now ON.");
        } else if (batteryLevel == 0) {
            System.out.println("Battery is dead. Please charge your phone.");
        } else {
            System.out.println("Phone is already ON.");
        }
    }
    public void powerOff() {
        if (isOn) {
            isOn = false;
            System.out.println(brand + " phone is now OFF.");
        } else {
            System.out.println("Phone is already OFF.");
        }
    }
}
public class MobileSystem {
    public static void main(String[] args) {
        MobilePhone phone = new MobilePhone("Samsung", 80);
        System.out.println("Phone Brand: " + phone.getBrand());
        System.out.println("Battery Level: " + phone.getBatteryLevel() + "%");
        System.out.println("Phone Status: " + (phone.isPhoneOn() ? "ON" : "OFF"));
        phone.powerOn();

        phone.powerOff();
    }
}
```

OUTPUT:

```
D:\AVV CHENNAI\Object Oriented Programming\JAVA\Encapsulation\Encapsulation Programs>javac MobileSystem.java

D:\AVV CHENNAI\Object Oriented Programming\JAVA\Encapsulation\Encapsulation Programs>java MobileSystem.java
Phone Brand: Samsung
Battery Level: 80%
Phone Status: OFF
Samsung phone is now ON.
Samsung phone is now OFF.
```

14.b) Student Management System:

CODE:

```java
class Student {
    private String studentID;
    private String name;
    private double GPA;

    public Student(String studentID, String name, double GPA) {
        this.studentID = studentID;
        this.name = name;
        this.GPA = GPA;
    }
    public String getStudentID() {
        return studentID;
    }
    public String getName() {
        return name;
    }
    public double getGPA() {
        return GPA;
    }
    public void updateGPA(double newGPA) {
        if (newGPA >= 0.0 && newGPA <= 4.0) {
            this.GPA = newGPA;
            System.out.println("GPA updated successfully.");
        } else {
            System.out.println("Invalid GPA value.");
        }
    }
}
public class StudentManagement {
    public static void main(String[] args) {
        Student student = new Student("S12345", "Emma Watson", 3.8);
        System.out.println("Student: " + student.getName());
        System.out.println("GPA: " + student.getGPA());
        student.updateGPA(3.9);
        System.out.println("Updated GPA: " + student.getGPA());
    }
}
```

OUTPUT:

```
D:\AVV CHENNAI\Object Oriented Programming\JAVA\Encapsulation\Encapsulation Programs>javac StudentManagement.java

D:\AVV CHENNAI\Object Oriented Programming\JAVA\Encapsulation\Encapsulation Programs>java StudentManagement.java
Student: Emma Watson
GPA: 3.8
GPA updated successfully.
Updated GPA: 3.9
```

14.c) Train Reservation System

CODE:

```
class TrainTicket {
    private String trainNumber;
    private int availableSeats;

    public TrainTicket(String trainNumber, int availableSeats) {
        this.trainNumber = trainNumber;
        this.availableSeats = availableSeats;
    }
    public String getTrainNumber() {
        return trainNumber;
    }
    public int getAvailableSeats() {
        return availableSeats;
    }
    public void bookTicket(int seats) {
        if (seats > 0 && seats <= availableSeats) {
            availableSeats -= seats;
            System.out.println(seats + " ticket(s) booked successfully.");
        } else {
            System.out.println("Not enough seats available.");
        }
    }
}
public class TrainReservation {
    public static void main(String[] args) {
        TrainTicket ticket = new TrainTicket("TR123", 50);
        System.out.println("Train: " + ticket.getTrainNumber());
        System.out.println("Available Seats: " + ticket.getAvailableSeats());
        ticket.bookTicket(5);
        System.out.println("Updated Available Seats: " + ticket.getAvailableSeats());
    }
}
```

OUTPUT:

```
D:\AVV CHENNAI\Object Oriented Programming\JAVA\Encapsulation\Encapsulation Programs>javac TrainReservation.java

D:\AVV CHENNAI\Object Oriented Programming\JAVA\Encapsulation\Encapsulation Programs>java TrainReservation.java
Train: TR123
Available Seats: 50
5 ticket(s) booked successfully.
Updated Available Seats: 45
```

14.d) Weather App:

CODE:

```
class Weather {
    private double temperature;
    private String condition;

    public Weather(double temperature, String condition) {
        this.temperature = temperature;
        this.condition = condition;
    }
    public double getTemperature() {
        return temperature;
    }
    public String getCondition() {
        return condition;
    }
    public void updateWeather(double temp, String newCondition) {
        this.temperature = temp;
        this.condition = newCondition;
        System.out.println("Weather updated successfully.");
    }
}
public class WeatherApp {
    public static void main(String[] args) {
        Weather todayWeather = new Weather(28.5, "Sunny");
        System.out.println("Temperature: " + todayWeather.getTemperature() + "°C");
        System.out.println("Condition: " + todayWeather.getCondition());
        todayWeather.updateWeather(30.0, "Cloudy");
        System.out.println("Updated Condition: " + todayWeather.getCondition());
    }
}
```

OUTPUT:

```
D:\AVV CHENNAI\Object Oriented Programming\JAVA\Encapsulation\Encapsulation Programs>javac WeatherApp.java

D:\AVV CHENNAI\Object Oriented Programming\JAVA\Encapsulation\Encapsulation Programs>java WeatherApp.java
Temperature: 28.5°C
Condition: Sunny
Weather updated successfully.
Updated Condition: Cloudy
```

15) PACKAGES PROGRAMS

15.a)Prime Number Finder:

PACKAGE CODE(User - Defined Packages):

MathOperations.java:

```
package math.utils;
public class MathOperations {

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

    public static int subtract(int a, int b) {
        return a - b;
    }

    public static int multiply(int a, int b) {
        return a * b;
    }

    public static double divide(int a, int b) {
        if (b == 0) {
            throw new IllegalArgumentException("Cannot divide by zero");
        }
        return (double) a / b;
    }

      public static int factorial(int n) {
        if (n < 0) {
            throw new IllegalArgumentException("Factorial is not defined for negative numbers");
        }
        int result = 1;
        for (int i = 2; i <= n; i++) {
            result *= i;
        }
        return result;
    }

    public static boolean isPrime(int number) {
        if (number <= 1) return false;
        if (number == 2) return true;
        if (number % 2 == 0) return false;

        for (int i = 3; i * i <= number; i += 2) {
            if (number % i == 0) {
                return false;
            }
        }
        return true;
    }

    public static int gcd(int a, int b) {
        while (b != 0) {
            int temp = b;
```

```
            b = a % b;
            a = temp;
        }
        return a;
    }
}
```

File Operations:

```
package file.utils;

import java.io.*;
import java.nio.file.*;
import java.util.*;
public class FileOperations {

    public static void createFile(String filePath) throws IOException {
        File file = new File(filePath);
        if (file.createNewFile()) {
            System.out.println("File created: " + file.getName());
        } else {
            System.out.println("File already exists.");
        }
    }

    public static void writeToFile(String filePath, String content) throws IOException {
        Files.write(Paths.get(filePath), content.getBytes(), StandardOpenOption.CREATE);
    }

    public static String readFile(String filePath) throws IOException {
        return new String(Files.readAllBytes(Paths.get(filePath)));
    }

   public static void appendToFile(String filePath, String content) throws IOException {
        Files.write(Paths.get(filePath), content.getBytes(), StandardOpenOption.APPEND);
    }

    public static int countLines(String filePath) throws IOException {
        try (BufferedReader reader = new BufferedReader(new FileReader(filePath))) {
            int lines = 0;
            while (reader.readLine() != null) lines++;
            return lines;
        }
    }

    public static List<String> findFilesWithExtension(String directory, String extension) throws IOException {
        List<String> result = new ArrayList<>();
        try (DirectoryStream<Path> stream = Files.newDirectoryStream(Paths.get(directory), "*." + extension)) {
            for (Path path : stream) {
                if (!Files.isDirectory(path)) {
                    result.add(path.getFileName().toString());
                }
            }
        }
        return result;
    }

    public static void copyFile(String sourcePath, String destPath) throws IOException {
        Files.copy(Paths.get(sourcePath), Paths.get(destPath), StandardCopyOption.REPLACE_EXISTING);
    }
}
```

PROGRAM CODE:

```java
import math.utils.MathOperations;
import file.utils.FileOperations;
import java.io.IOException;

public class PrimeNumberFinder {
    public static void main(String[] args) {
        int limit = 100;
        StringBuilder primes = new StringBuilder();

        System.out.println("Finding prime numbers up to " + limit);
        for (int i = 2; i <= limit; i++) {
            if (MathOperations.isPrime(i)) {
                primes.append(i).append(" ");
                System.out.print(i + " ");
            }
        }

        try {
            FileOperations.writeToFile("primes.txt", primes.toString());
            System.out.println("\n\nPrime numbers saved to primes.txt");
        } catch (IOException e) {
            System.out.println("Error saving primes: " + e.getMessage());
        }
    }
}
```

OUTPUT:

```
D:\AVV CHENNAI\Object Oriented Programming\JAVA\Encapsulation\Packages Programs\User Defined Packages>javac -d . MathOpe
rations.java

D:\AVV CHENNAI\Object Oriented Programming\JAVA\Encapsulation\Packages Programs\User Defined Packages>javac -d . FileOpe
rations.java

D:\AVV CHENNAI\Object Oriented Programming\JAVA\Encapsulation\Packages Programs\User Defined Packages>javac PrimeNumberF
inder.java

D:\AVV CHENNAI\Object Oriented Programming\JAVA\Encapsulation\Packages Programs\User Defined Packages>java PrimeNumberFi
nder
Finding prime numbers up to 100
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97

Prime numbers saved to primes.txt
```

15.b) Text File Processor(User - Defined Programmes):

PACKAGE CODE:

File Operations:

```java
package file.utils;
```

```java
import java.io.*;
import java.nio.file.*;
import java.util.*;
public class FileOperations {

    public static void createFile(String filePath) throws IOException {
        File file = new File(filePath);
        if (file.createNewFile()) {
            System.out.println("File created: " + file.getName());
        } else {
            System.out.println("File already exists.");
        }
    }

    public static void writeToFile(String filePath, String content) throws IOException {
        Files.write(Paths.get(filePath), content.getBytes(), StandardOpenOption.CREATE);
    }

    public static String readFile(String filePath) throws IOException {
        return new String(Files.readAllBytes(Paths.get(filePath)));
    }

   public static void appendToFile(String filePath, String content) throws IOException {
        Files.write(Paths.get(filePath), content.getBytes(), StandardOpenOption.APPEND);
    }

    public static int countLines(String filePath) throws IOException {
        try (BufferedReader reader = new BufferedReader(new FileReader(filePath))) {
            int lines = 0;
            while (reader.readLine() != null) lines++;
            return lines;
        }
    }

    public static List<String> findFilesWithExtension(String directory, String extension) throws IOException {
        List<String> result = new ArrayList<>();
        try (DirectoryStream<Path> stream = Files.newDirectoryStream(Paths.get(directory), "*." + extension)) {
            for (Path path : stream) {
                if (!Files.isDirectory(path)) {
                    result.add(path.getFileName().toString());
                }
            }
        }
        return result;
    }

    public static void copyFile(String sourcePath, String destPath) throws IOException {
        Files.copy(Paths.get(sourcePath), Paths.get(destPath), StandardCopyOption.REPLACE_EXISTING);
    }
}
```

String Operations:

```java
package string.utils;

import java.util.Arrays;
public class StringOperations {

    public static String reverse(String input) {
        return new StringBuilder(input).reverse().toString();
    }

    public static int countVowels(String input) {
        int count = 0;
        String vowels = "aeiouAEIOU";
        for (char c : input.toCharArray()) {
```

```
            if (vowels.indexOf(c) != -1) {
                count++;
            }
        }
        return count;
    }

    public static boolean isPalindrome(String input) {
        String clean = input.replaceAll("[^a-zA-Z0-9]", "").toLowerCase();
        return clean.equals(reverse(clean));
    }

    public static String[] splitIntoWords(String input) {
        return input.trim().split("\\s+");
    }

    public static String removeDuplicates(String input) {
        StringBuilder result = new StringBuilder();
        for (int i = 0; i < input.length(); i++) {
            if (result.indexOf(String.valueOf(input.charAt(i))) == -1) {
                result.append(input.charAt(i));
            }
        }
        return result.toString();
    }

    public static String capitalizeWords(String input) {
        String[] words = splitIntoWords(input);
        for (int i = 0; i < words.length; i++) {
            if (words[i].length() > 0) {
                words[i] = words[i].substring(0, 1).toUpperCase() +
                           words[i].substring(1).toLowerCase();
            }
        }
        return String.join(" ", words);
    }
}
```

PROGRAMME CODE:

```
import file.utils.FileOperations;
import string.utils.StringOperations;
import java.io.IOException;
import java.util.List;

public class TextFileProcessor {
    public static void main(String[] args) {
        String inputFile = "input.txt";
        String outputFile = "output.txt";

        try {
            FileOperations.createFile(inputFile);
            FileOperations.writeToFile(inputFile, "This is the first line.\n");
            FileOperations.appendToFile(inputFile, "This is the second line.\n");
            FileOperations.appendToFile(inputFile, "Madam Arora teaches malayalam\n");

            String content = FileOperations.readFile(inputFile);
            String processedContent = processContent(content);

            FileOperations.writeToFile(outputFile, processedContent);

            System.out.println("File processing complete.");
            System.out.println("Original content:\n" + content);
            System.out.println("\nProcessed content:\n" + processedContent);

        } catch (IOException e) {
            System.out.println("Error: " + e.getMessage());
```

```
        }
    }

    private static String processContent(String content) {
        String[] lines = content.split("\n");
        StringBuilder result = new StringBuilder();

        for (String line : lines) {
            String processedLine = StringOperations.capitalizeWords(line);
            if (StringOperations.isPalindrome(line.replaceAll("[^a-zA-Z]", ""))) {
                processedLine += " [PALINDROME]";
            }
            result.append(processedLine).append("\n");
        }

        return result.toString();
    }
}
```

15.c) Date Time Example (Built - In Packages)

CODE:

```
import java.time.*;
import java.time.format.*;
import java.util.*;
import java.text.*;
public class DateTimeExample {
    public static void main(String[] args) {
        LocalDateTime now = LocalDateTime.now();

        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("MMMM dd, yyyy hh:mm a");
        System.out.println("Current DateTime: " + now.format(formatter));

        LocalDate birthday = LocalDate.of(1990, Month.JUNE, 15);
        System.out.println("Birthday: " + birthday.format(DateTimeFormatter.ISO_DATE));

        LocalTime start = LocalTime.of(9, 0);
        LocalTime end = LocalTime.of(17, 30);
        Duration duration = Duration.between(start, end);

        NumberFormat nf = NumberFormat.getInstance();
        nf.setMaximumFractionDigits(1);
        System.out.println("Work hours: " + nf.format(duration.toHours()) + " hours");

        Scanner scanner = new Scanner(System.in);
        System.out.print("\nEnter your birth year: ");
        int year = scanner.nextInt();

        Period age = Period.between(LocalDate.of(year, Month.JANUARY, 1), LocalDate.now());
        System.out.println("You're approximately " + age.getYears() + " years old");

        Random random = new Random();
        int randomDay = random.nextInt(28) + 1;
        LocalDate randomDate = LocalDate.of(2023, Month.of(random.nextInt(12) + 1), randomDay);
        System.out.println("Random date this year: " + randomDate);

        scanner.close();
    }
}
```

OUTPUT:

```
D:\AVV CHENNAI\Object Oriented Programming\JAVA\Encapsulation\Packages Programs\Built-in Packages>javac DateTimeExample.
java

D:\AVV CHENNAI\Object Oriented Programming\JAVA\Encapsulation\Packages Programs\Built-in Packages>java DateTimeExample.j
ava
Current DateTime: April 05, 2025 09:58 am
Birthday: 1990-06-15
Work hours: 8 hours
```

15.d) Math Example (Built - In Packages):

CODE:

```java
import java.util.*;
import java.time.*;
import java.text.*;

public class MathExample {
    public static void main(String[] args) {
        System.out.println("Square root of 25: " + Math.sqrt(25));
        System.out.println("Power: " + Math.pow(2, 3));

        Random random = new Random();
        System.out.println("Random number (0-100): " + random.nextInt(101));

        System.out.println("Sin 30°: " + Math.sin(Math.toRadians(30)));

        System.out.println("PI: " + Math.PI);
        System.out.println("E: " + Math.E);

        LocalDate today = LocalDate.now();
        LocalDate futureDate = today.plusDays(100);
        System.out.println("\n100 days from today: " + futureDate);

        DecimalFormat df = new DecimalFormat("#.##");
        System.out.println("Formatted PI: " + df.format(Math.PI));

        Scanner scanner = new Scanner(System.in);
        System.out.print("\nEnter a number to square: ");
        double num = scanner.nextDouble();
        System.out.println("Square of " + num + ": " + Math.pow(num, 2));
        scanner.close();
    }
}
```

OUTPUT:

```
D:\AVV CHENNAI\Object Oriented Programming\JAVA\Encapsulation\Packages Programs\Built-in Packages>javac MathExample.java


D:\AVV CHENNAI\Object Oriented Programming\JAVA\Encapsulation\Packages Programs\Built-in Packages>java MathExample.java
Square root of 25: 5.0
Power: 8.0
Random number (0-100): 4
Sin 30°: 0.49999999999999994
PI: 3.141592653589793
E: 2.718281828459045

100 days from today: 2025-07-14
Formatted PI: 3.14

Enter a number to square: 5
Square of 5.0: 25.0
```

16. EXCEPTION HANDLING PROGRAMS

16.a) Custom Exception Example:

CODE:

```
class InvalidAgeException extends Exception {
    public InvalidAgeException(String message) {
        super(message);
    }
}

public class CustomExceptionExample {
    static void checkAge(int age) throws InvalidAgeException {
        if (age < 18) {
            throw new InvalidAgeException("You must be at least 18 years old.");
        }
        System.out.println("Age verified.");
    }
    public static void main(String[] args) {
        try {
            checkAge(15);
        } catch (InvalidAgeException e) {
            System.out.println("Exception: " + e.getMessage());
        }
    }
}
```

OUTPUT:

```
D:\AVV CHENNAI\Object Oriented Programming\JAVA\Exception Handling>javac CustomExceptionExample.java

D:\AVV CHENNAI\Object Oriented Programming\JAVA\Exception Handling>java CustomExceptionExample.java
Exception: You must be at least 18 years old.
```

16.b) Finally Example:

CODE:

```
public class FinallyExample {
    public static void main(String[] args) {
        try {
            String text = null;
            System.out.println(text.length());
        } catch (NullPointerException e) {
            System.out.println("Error: Null value detected!");
        } finally {
            System.out.println("This code runs no matter what.");
```

```
        }
    }
}
```

OUTPUT:

```
D:\AVV CHENNAI\Object Oriented Programming\JAVA\Exception Handling>javac FinallyExample.java

D:\AVV CHENNAI\Object Oriented Programming\JAVA\Exception Handling>java FinallyExample.java
Error: Null value detected!
This code runs no matter what.
```

16.c) Input Mismatch Example:

CODE:

```
import java.util.Scanner;
import java.util.InputMismatchException;

public class InputMismatchExample {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        try {
            System.out.print("Enter a number: ");
            int num = scanner.nextInt();
            System.out.println("You entered: " + num);
        } catch (InputMismatchException e) {
            System.out.println("Error: Please enter a valid number.");
        }
    }
}
```

OUTPUT:

```
D:\AVV CHENNAI\Object Oriented Programming\JAVA\Exception Handling>javac InputMismatchExample.java

D:\AVV CHENNAI\Object Oriented Programming\JAVA\Exception Handling>java InputMismatchExample.java
Enter a number: Hi
Error: Please enter a valid number.
```

16.d) Number Format Example :

CODE:

```java
public class NumberFormatExample {
    public static void main(String[] args) {
        try {
            int num = Integer.parseInt("abc");
        } catch (NumberFormatException e) {
            System.out.println("Error: Invalid number format.");
        }
    }
}
```

OUTPUT:

```
D:\AVV CHENNAI\Object Oriented Programming\JAVA\Exception Handling>javac NumberFormatExample.java

D:\AVV CHENNAI\Object Oriented Programming\JAVA\Exception Handling>java NumberFormatExample.java
Error: Invalid number format.
```

17. FILE HANDLING PROGRAMS

17.a) Append File:

CODE:

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

public class AppendFile {
    public static void main(String[] args) {
        try (FileWriter writer = new FileWriter("test.txt", true)) {
            writer.write("\nAppending new text!");
            System.out.println("Successfully appended to the file.");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

OUTPUT:

```
D:\AVV CHENNAI\Object Oriented Programming\JAVA\File Handling>javac AppendFile.java

D:\AVV CHENNAI\Object Oriented Programming\JAVA\File Handling>java AppendFile.java
Successfully appended to the file.
```

Text.txt File before running the code:

Text.txt File after running the code:

17.b) Read File:

CODE:

```java
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:

```
D:\AVV CHENNAI\Object Oriented Programming\JAVA\File Handling>javac ReadFile.java

D:\AVV CHENNAI\Object Oriented Programming\JAVA\File Handling>java ReadFile.java
OOPs
Appending new text!
```

17.c) Replace Word:

CODE:

```java
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:

```
D:\AVV CHENNAI\Object Oriented Programming\JAVA\File Handling>javac ReplaceWord.java

D:\AVV CHENNAI\Object Oriented Programming\JAVA\File Handling>java ReplaceWord.java
Replaced all occurrences of 'Java' with 'Python'.
```

Test1.txt before running the code:

Test1.txt after running the code:

17.d) Write File:

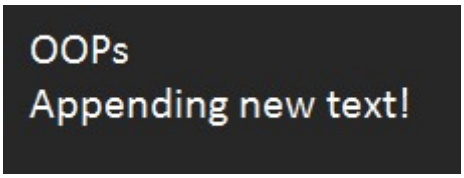
CODE:

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

public class WriteFile {
    public static void main(String[] args) {
        try (FileWriter writer = new FileWriter("test.txt")) {
            writer.write("Hello, Java File Handling!");
            System.out.println("Successfully wrote to the file.");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

OUTPUT:

```
D:\AVV CHENNAI\Object Oriented Programming\JAVA\File Handling>javac WriteFile.java

D:\AVV CHENNAI\Object Oriented Programming\JAVA\File Handling>java WriteFile.java
Successfully wrote to the file.
```

Test.txt File before running the code:



Test.txt File after running the code: