



AMRITA
VISHWA VIDYAPEETHAM

Amrita School of Computing

LAB RECORD

23CSE111 – Object Oriented Programs

S K DADA HUSSAIN

CH.SC.U4CSE24144

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

AMRITA VISHWA VIDYAPEETHAM

AMRITA SCHOOL OF COMPUTING

CHENNAI



SCHOOL OF
COMPUTING

AMRITA VISHWA VIDYAPEETHAM

AMRITA SCHOOL OF COMPUTING, CHENNAI

BONAFIDE CERTIFICATE

This is to certify that the Lab Record work for 23CSE111- object oriented programs Subject submitted by *sk Dada Hussain-ch.sc.u4cse24144* 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 8/04/2025

Index

S.NO	TITLE	PAGE.NO
	UML DIAGRAM	
1.	Railway Management system	6
	1.a) Use Case Diagram	6
	1.b) Class Diagram	7
	1.c) Sequence Diagram	8
	1.d) state Diagram	9
	1.e) Activity Diagram	10
2.	Online food Ordering System	
	2.a) Use Case Diagram	11
	2.b) Class Diagram	12
	2.c) Sequence Diagram	12
	2.d) state Diagram	13
	2.e) Activity Diagram	13
3.	BASIC JAVA PROGRAMS	
	3.a) Count Digits	14
	3.b) Count Down	14
	3.c) Even Odd	15
	3.d) Hollow Square	15
	3.e) Largest Digit	16
	3.f) Power of Number	17
	3.g) Reverse String	18
	3.h) Right Angle Triangle	20
	3.i) Simple Interest	21
	3.j) Sum of N Natural Number	22
	INHERITANCE	
4.	SINGLE INHERITANCE PROGRAMS	23
	4.a) <u>Employer</u>	24

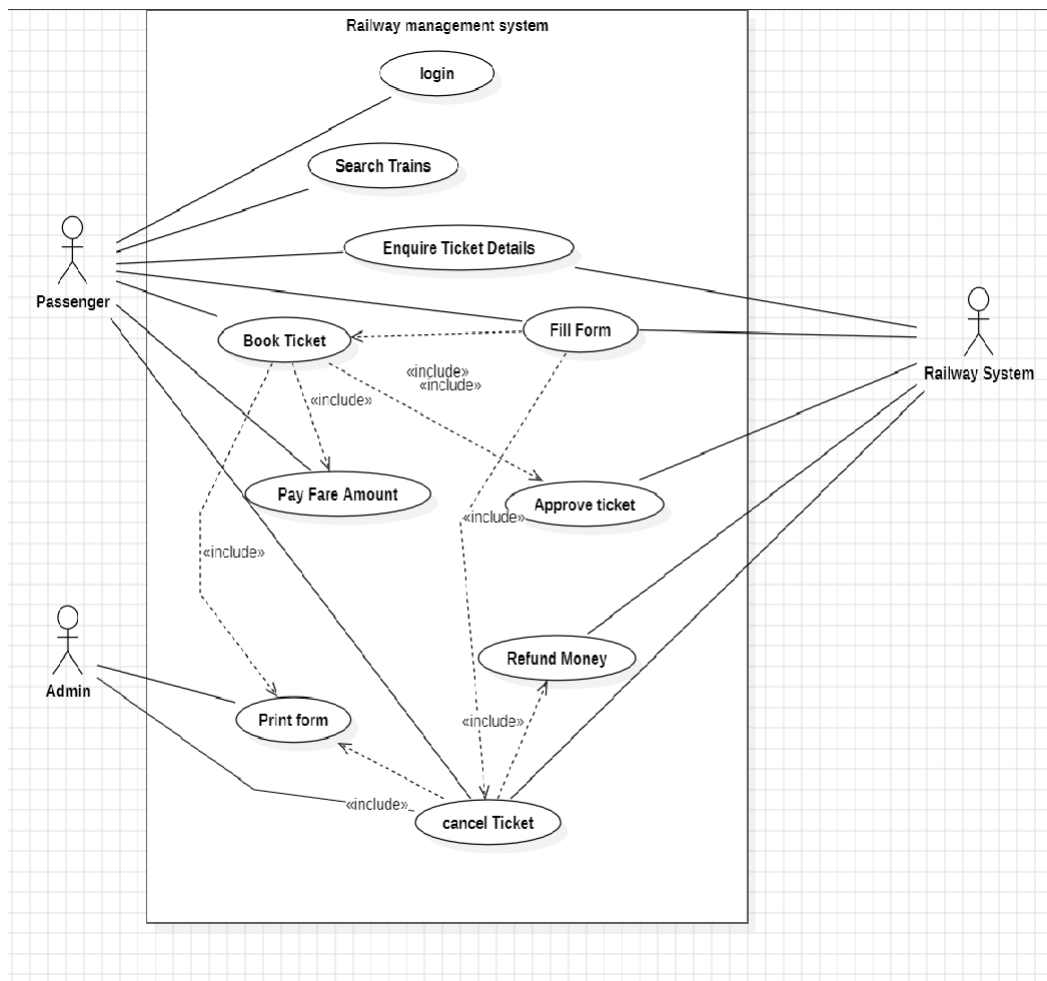
	4.b) <u>UserBankAccount.</u>	25
5.	MULTILEVEL INHERITANCE PROGRAMS	25
	5.a) <u>Money.</u>	26
	5.b) <u>Multi</u>	27
6.	HIERARCHICAL INHERITANCE PROGRAMS	28
	6.a) <u>TeacherSystem</u>	28
	6.b) <u>UniversityHierarchy</u>	29
7.	HYBRID INHERITANCE PROGRAMS	30
	7.a) <u>BankSystem</u>	30
	7.b) <u>UniversitySystem</u>	31
	POLYMORPHISM	32
8.	CONSTRUCTOR PROGRAMS	32
	8.a) <u>PatientConstructor</u>	32
9.	CONSTRUCTOR OVERLOADING PROGRAMS	34
	9.a) <u>HotelConstructorOverloading.</u>	34
10.	METHOD OVERLOADING PROGRAMS	35
	10.a) <u>Currency</u>	35
	10.b) <u>LibraryMethodOverloading</u>	36
11.	METHOD OVERRIDING PROGRAMS	36
	11.a) <u>RobotMethodOverriding</u>	37
	11.b) <u>TransportMethodOverriding</u>	38
	ABSTRACTION	38
12.	INTERFACE PROGRAMS	38
	12.a) <u>Bottle</u>	39
	12.b) <u>Charging</u>	39
	12.c) <u>Phones</u>	40
	12.d) <u>Tracker.</u>	41
13.	ABSTRACT CLASS PROGRAMS	42
	13.a) <u>Air</u>	42
	13.b) <u>Bank</u>	43
	13.c) <u>Cook</u>	43
	13.d) <u>Music</u>	44
	ENCAPSULATION	45
14.	ENCAPSULATION PROGRAMS	45

	14.a) <u>Main.</u>	454
	14.b) <u>Making</u>	46
	14.c) <u>Mobile</u>	47
	14.d) <u>Shop</u>	47
15.	PACKAGES PROGRAMS	48
	15.a) User Defined Packages- <u>Contact</u>	49
	15.b) User Defined Packages- <u>Task Manager</u>	51
	15.c) Built – in Package(3 Packages)- SumOfIntegers	53
	15.d) Built – in Package(3 Packages)- RandomNumber	54
16.	EXCEPTION HANDLING PROGRAMS	56
	16.a) ArrayExample	56
	16.b) DivisionExample	57
	16.c) NumberFormatExample	57
	16.d) FileReadExample	58
17.	FILE HANDLING PROGRAMS	59
	17.a) <u>FileAppendExample</u>	59
	17.b) <u>FileDeleteExample</u>	60
	17.c) <u>FileReadExample</u>	61
	17.d) <u>FileWriteExample.</u>	62

a)

Aim : To Demonstrate use case Diagram of Railway Reservation System

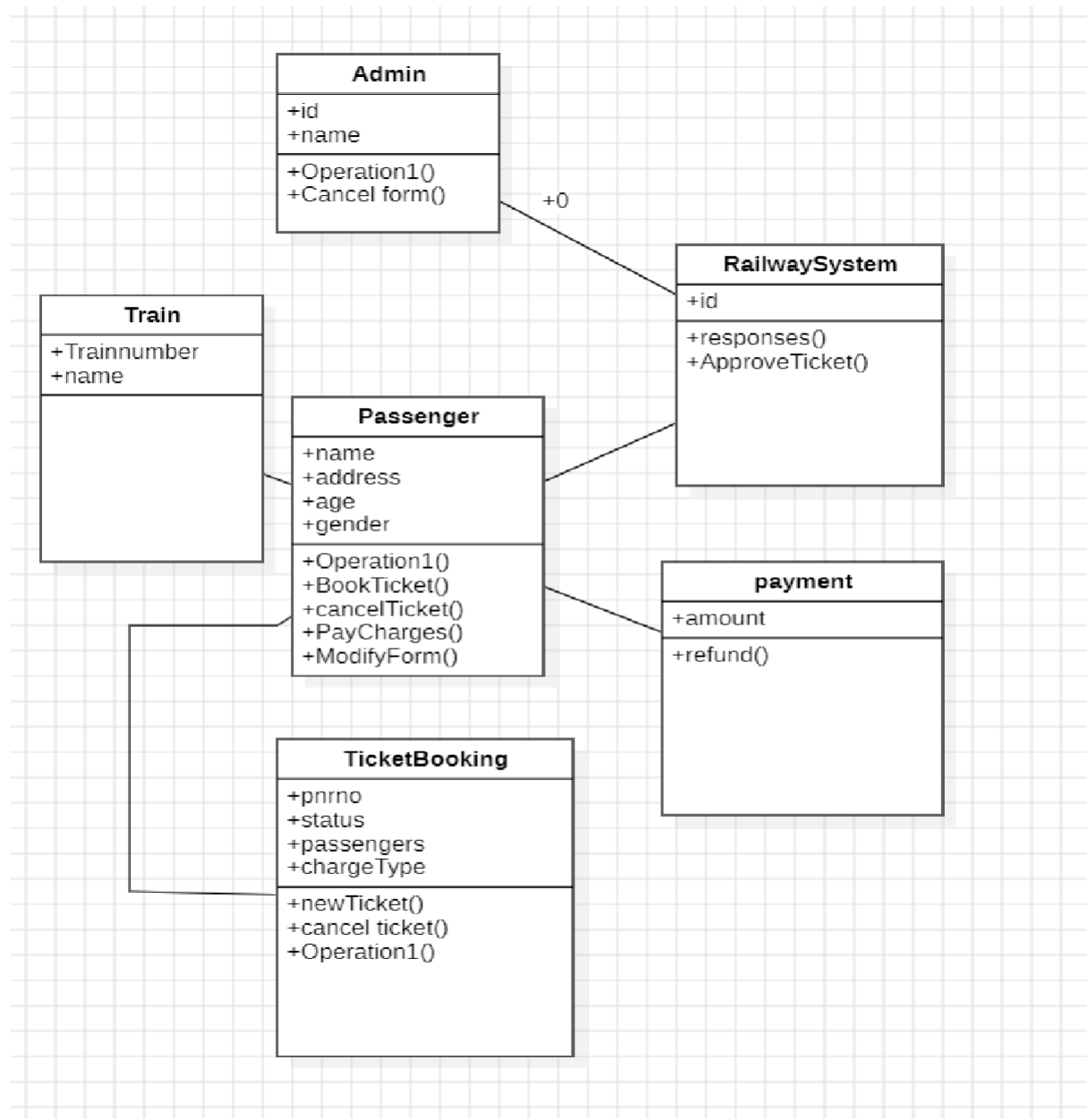
Diagram:



b)

Aim : To Demonstrate class Diagram of Railway Reservation System

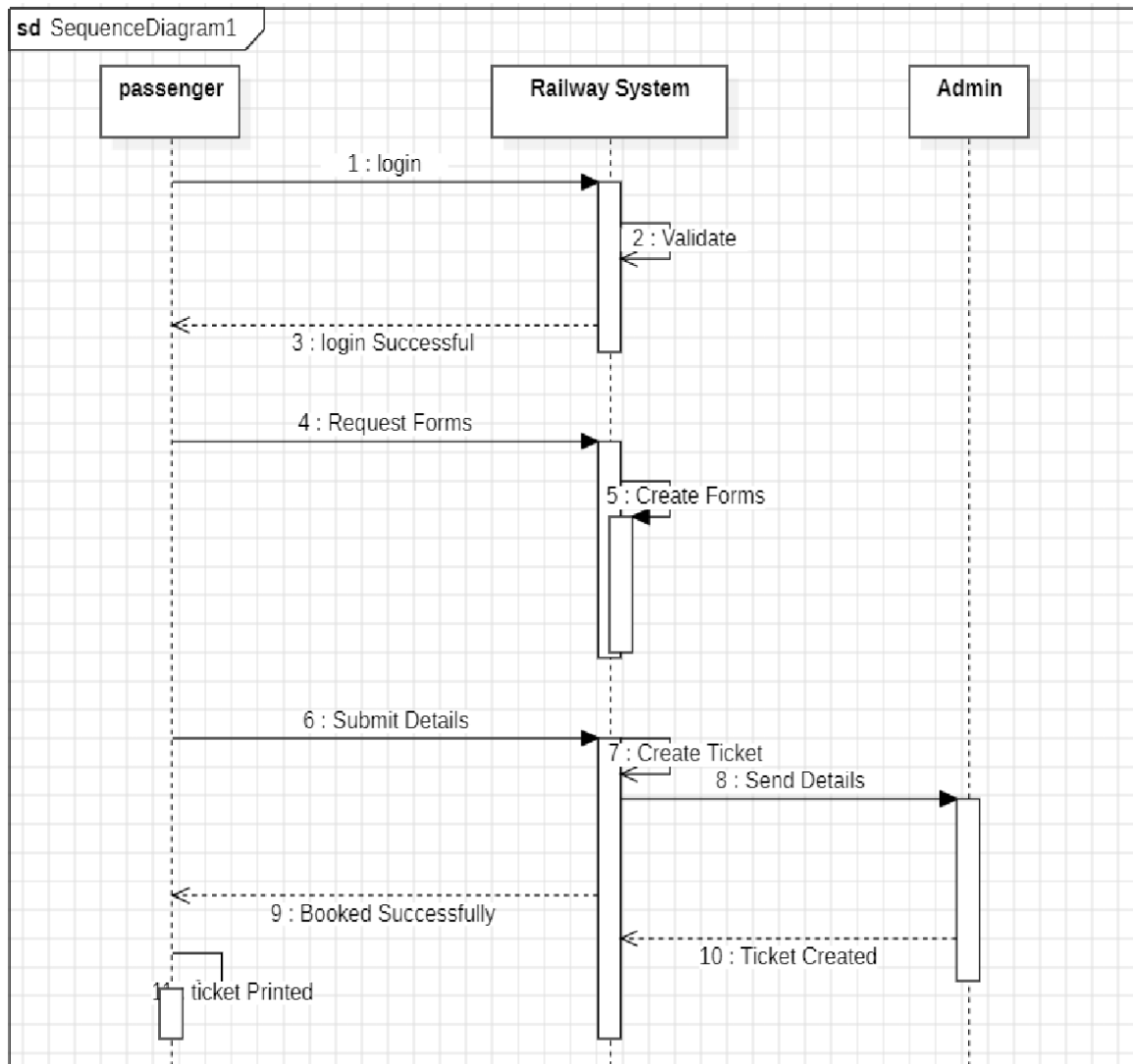
Diagram:



c)

Aim : To Demonstrate Sequence Diagram of Railway Reservation System

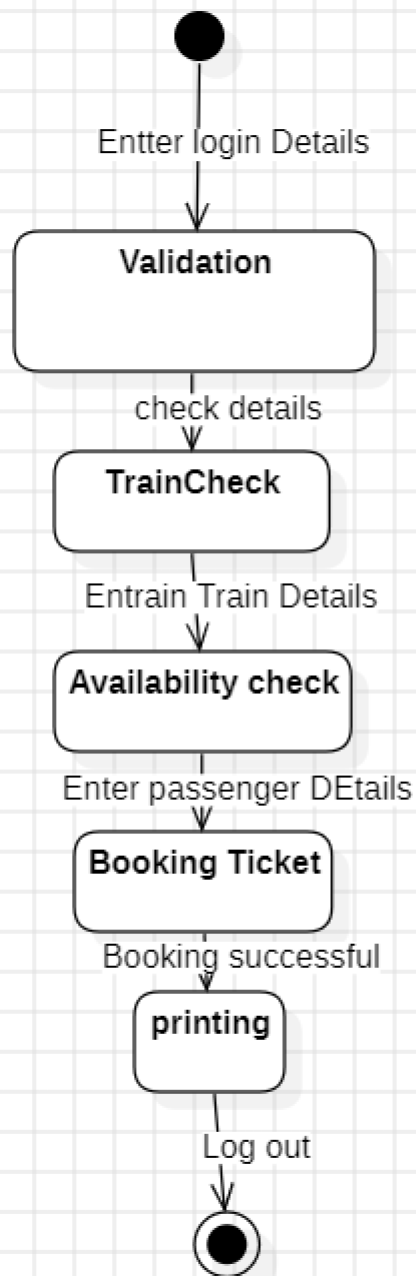
Diagram:



d)

Aim : To Demonstrate StateChart Diagram of Railway Reservation System

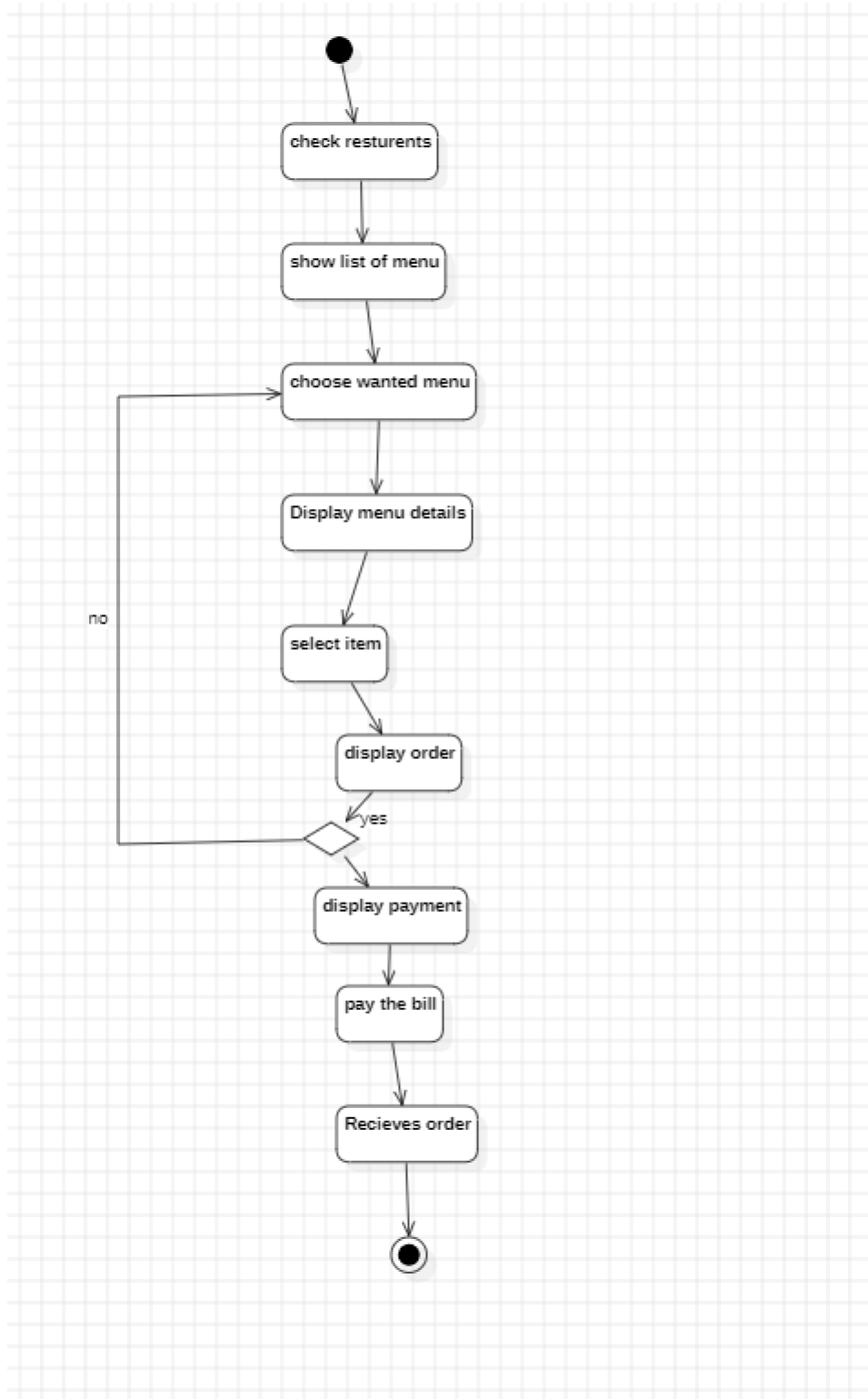
Diagram:



e)

Aim : To Demonstrate Activity Diagram of Railway Reservation System

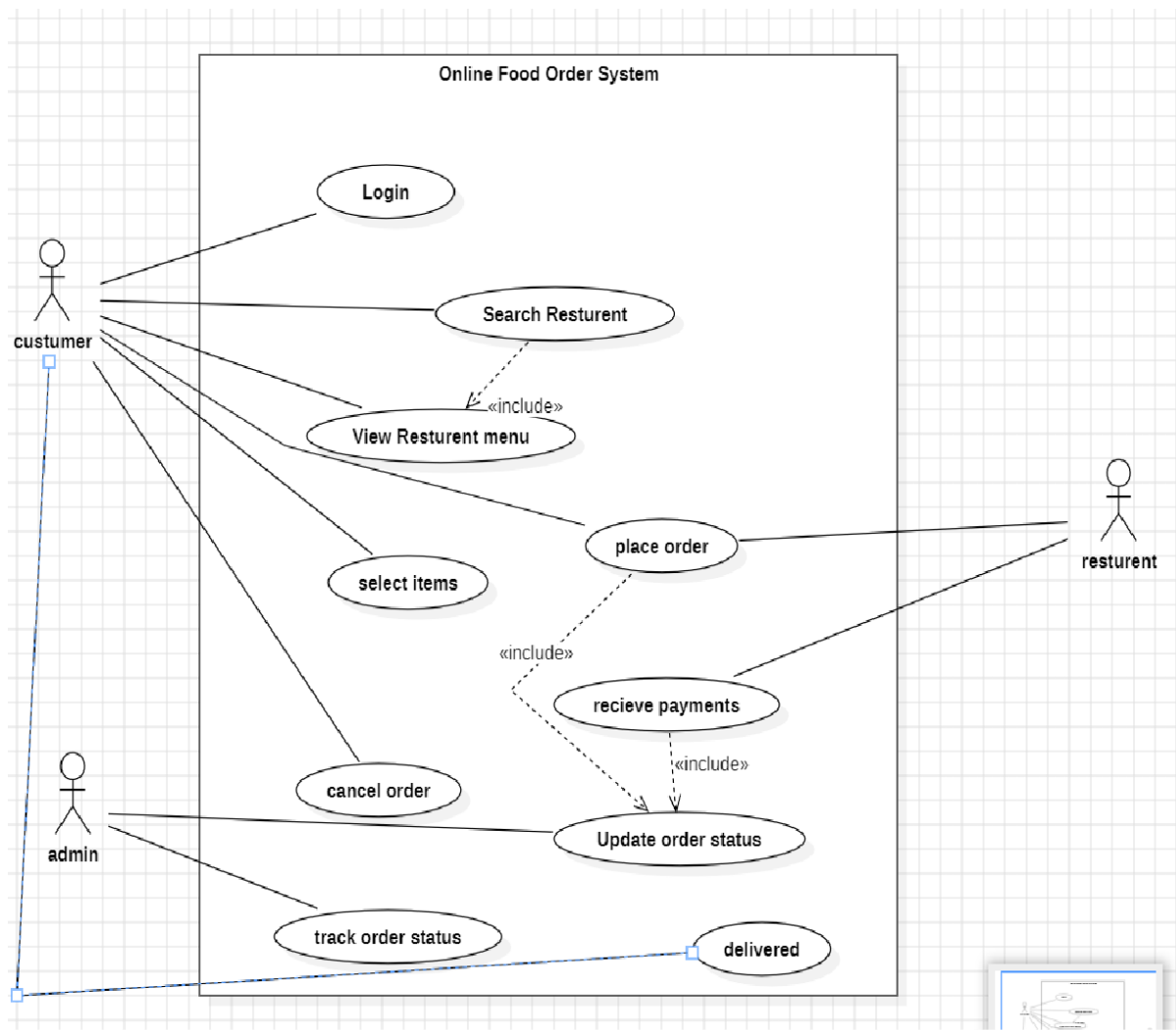
Diagram:



a)

Aim : To Demonstrate Use case Diagram of Online Food Order System

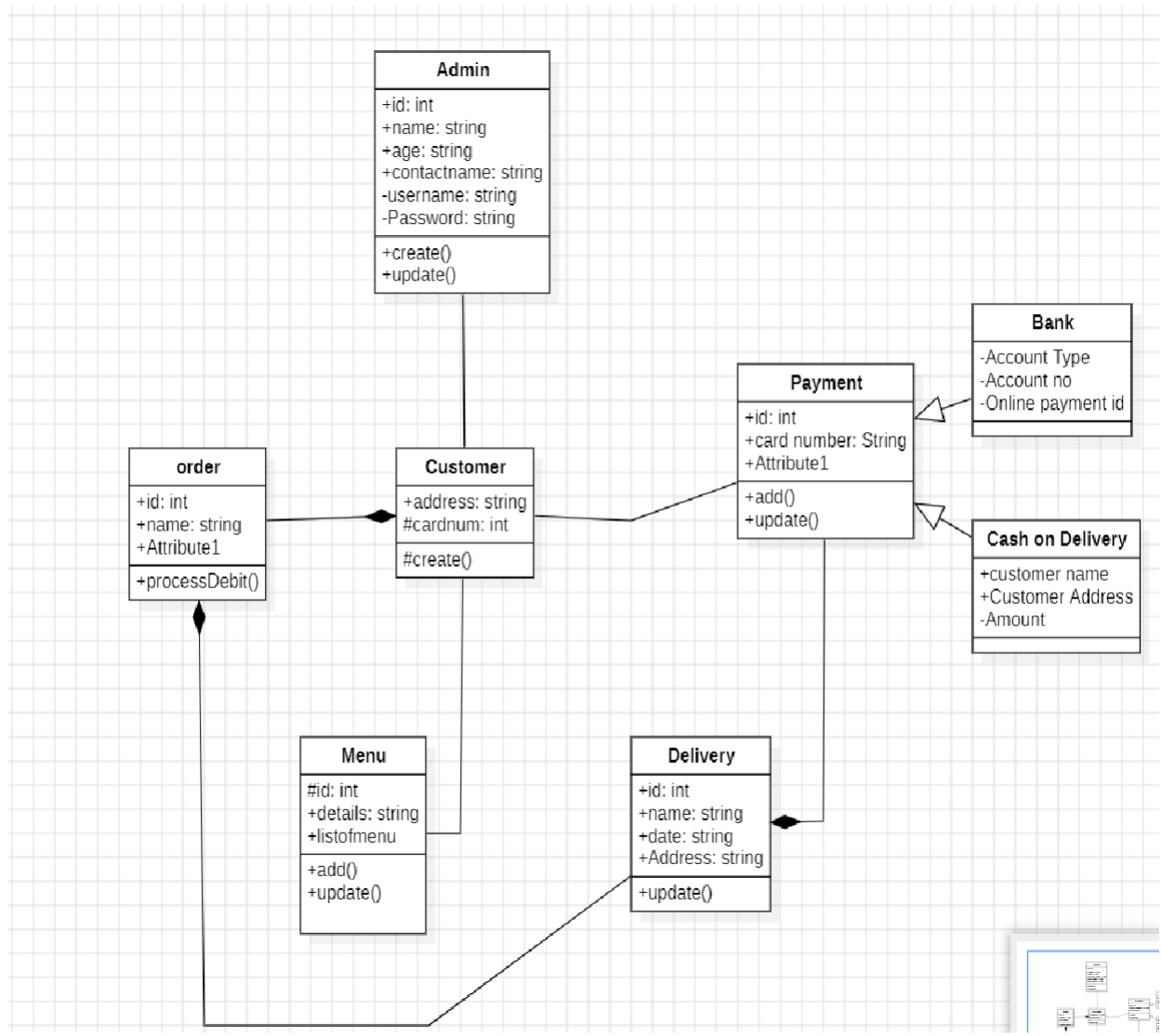
Diagram:



b)

Aim : To Demonstrate Class Diagram of Online Food Order System

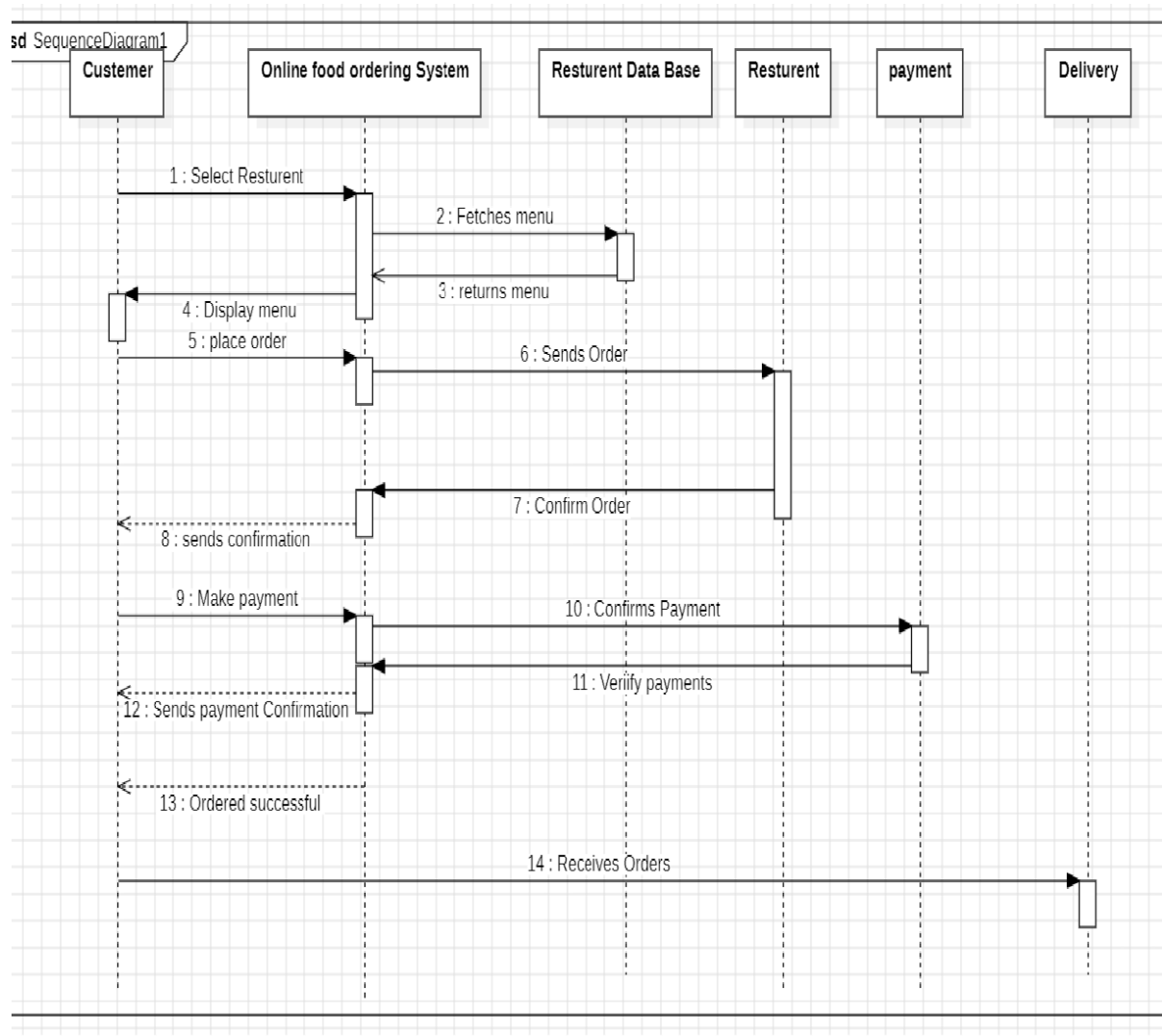
Diagram:



c)

Aim : To Demonstrate Sequence Diagram of Online Food Order System

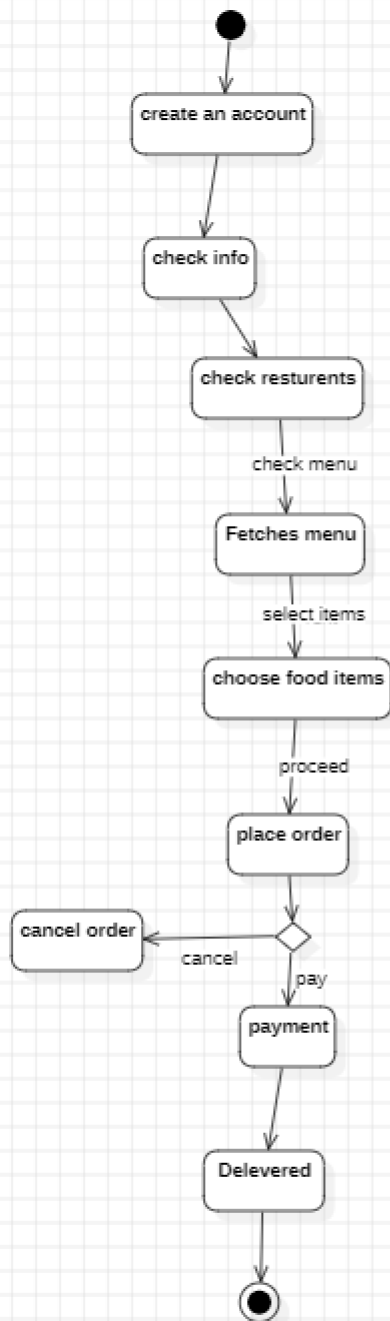
Diagram:



d)

Aim : To Demonstrate StateChart Diagram of Online Food Order System

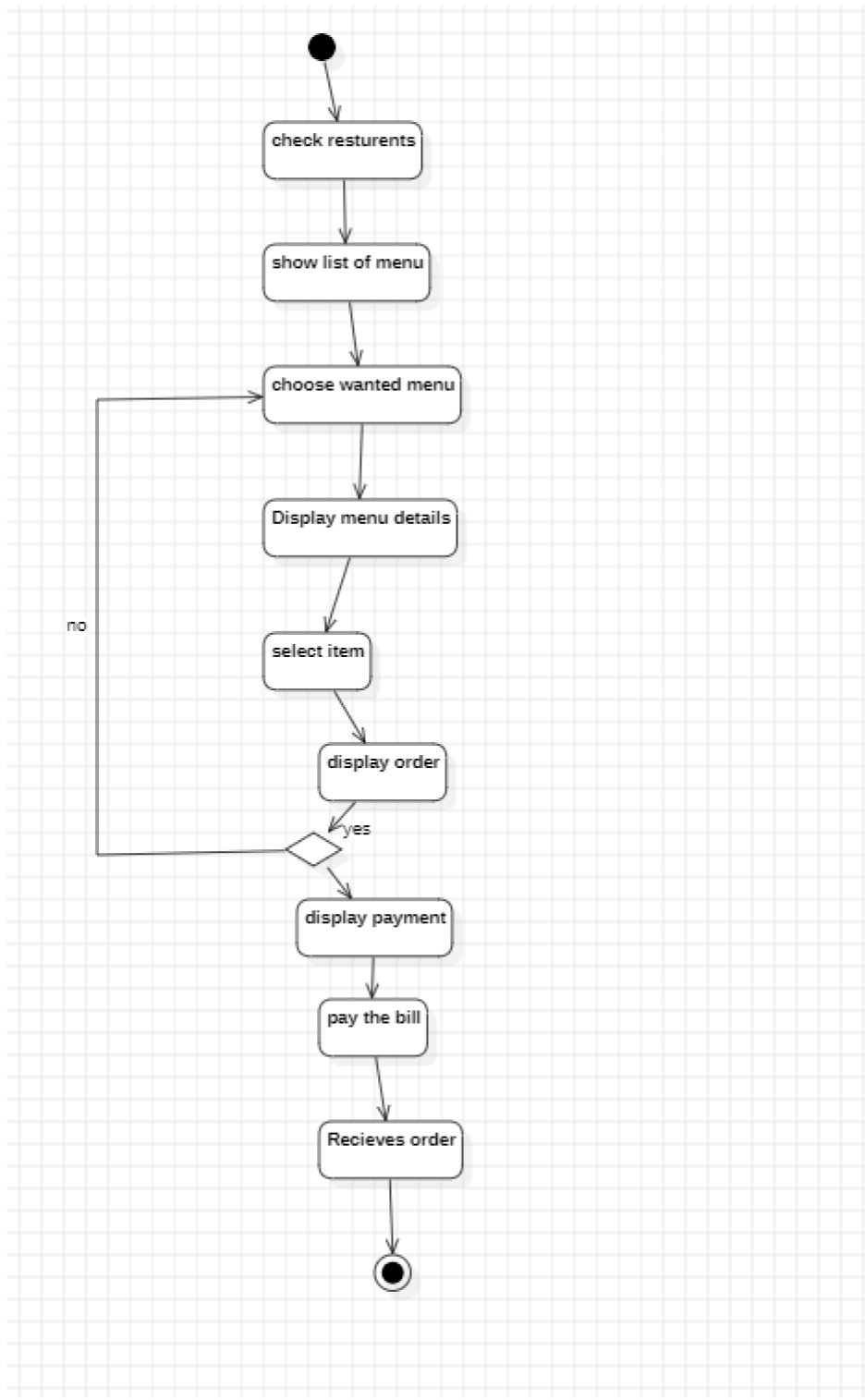
Diagram:



e)

Aim : To Demonstrate Activity Diagram of Online Food Order System

Diagram:

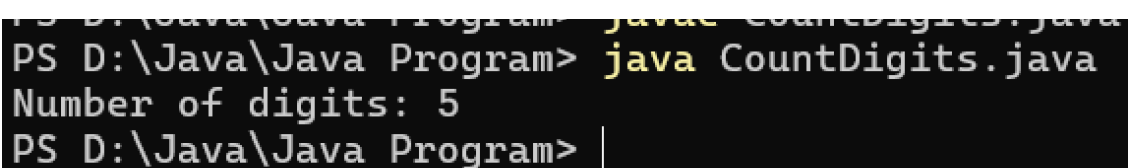


a) Count numbers

Code:

```
public class CountDigits {  
    public static void main(String[] args) {  
        int num = 12345, count = 0;  
        while (num != 0) {  
            num /= 10;  
            count++;  
        }  
        System.out.println("Number of digits: " + count);  
    }  
}
```

Output:



```
PS D:\Java\Java Program> java CountDigits.java  
Number of digits: 5  
PS D:\Java\Java Program> |
```


b) Count Down

Code:

```
public class Countdown {  
    public static void main(String[] args) throws InterruptedException {  
        int start = 10;  
        for (int i = start; i >= 0; i--) {  
            System.out.println(i);  
            Thread.sleep(1000); // Delay for 1 second  
        }  
        System.out.println("Time's up!");  
    }  
}
```

Output:

```
PS D:\Java\Java Program> javac Countdown.java  
PS D:\Java\Java Program> java Countdown.java  
10  
9  
8  
7  
6  
5  
4  
3  
2  
1  
0  
Time's up!  
PS D:\Java\Java Program> |
```

c) Even Odd

Code:

```
public class EvenOdd {  
    public static void main(String[] args) {  
        int n = 20;  
        System.out.println("Even numbers:");  
        for (int i = 1; i <= n; i++) {  
            if (i % 2 == 0) System.out.print(i + " ");  
        }  
        System.out.println("\nOdd numbers:");  
        for (int i = 1; i <= n; i++) {  
            if (i % 2 != 0) System.out.print(i + " ");  
        }  
    }  
}
```

Output:

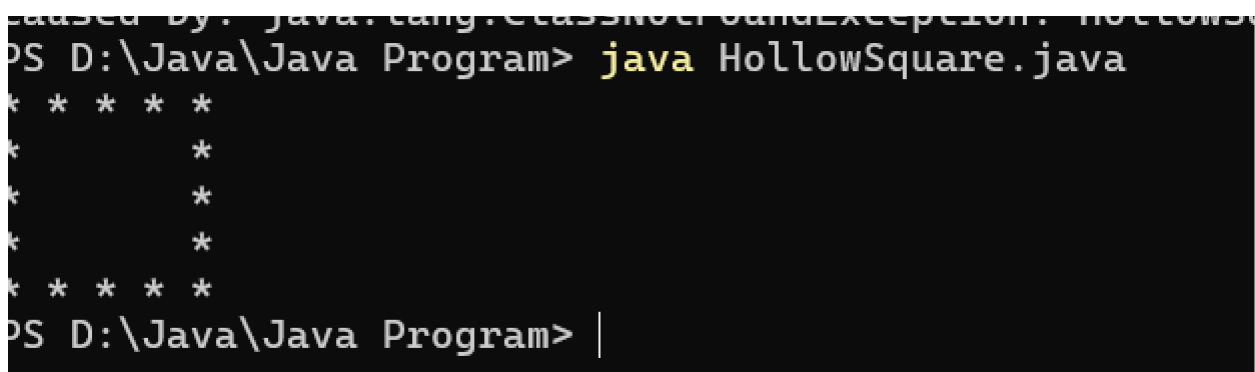
```
PS D:\Java\Java Program> java EvenOdd.java  
Even numbers:  
2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40 42 44 46 48 50  
Odd numbers:  
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49  
PS D:\Java\Java Program> |
```

d) Hollow Square

Code:

```
public class HollowSquare {  
    public static void main(String[] args) {  
        int size = 5;  
        for (int i = 1; i <= size; i++) {  
            for (int j = 1; j <= size; j++) {  
                if (i == 1 || i == size || j == 1 || j == size)  
                    System.out.print("* ");  
                else  
                    System.out.print(" ");  
            }  
            System.out.println();  
        }  
    }  
}
```

Output:




```
PS D:\Java\Java Program> java HollowSquare.java  
* * * * *  
*       *  
*       *  
*       *  
* * * * *  
PS D:\Java\Java Program> |
```

e) LargestDigit

Code:

```
public class LargestDigit {  
    public static void main(String[] args) {  
        int num = 987123, max = 0;  
        while (num > 0) {  
            int digit = num % 10;  
            if (digit > max) max = digit;  
            num /= 10;  
        }  
        System.out.println("Largest digit: " + max);  
    }  
}
```

Output:



```
PS D:\Java\Java Program> java LargestDigit.java  
Largest digit: 9  
PS D:\Java\Java Program> |
```

f) Power of Number

Code:

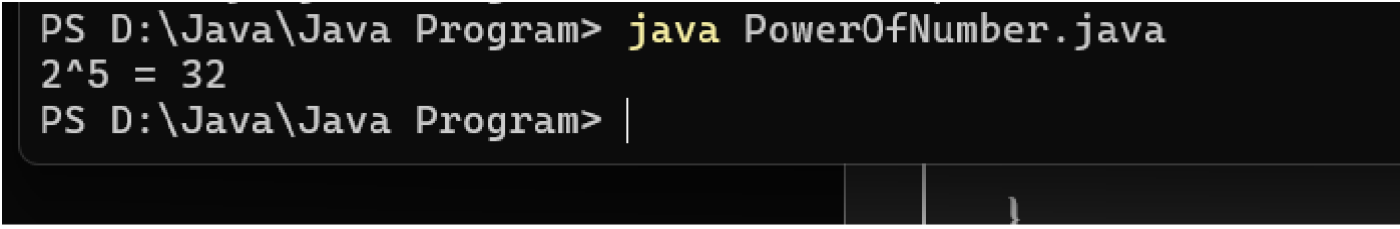
```
public class PowerOfNumber {  
    public static void main(String[] args) {  
        int base = 2, exp = 5, result = 1;  
        for (int i = 0; i < exp; i++) {  
            result *= base;  
        }  
    }  
}
```

```

    }
    System.out.println(base + "^" + exp + " = " + result);
}
}

```

Output:



```

PS D:\Java\Java Program> java PowerOfNumber.java
2^5 = 32
PS D:\Java\Java Program> |

```

g) Reverse String

Code:

```

public class ReverseString {
    public static void main(String[] args) {
        String str = "Dadu", reversed = "";
        for (int i = str.length() - 1; i >= 0; i--) {
            reversed += str.charAt(i);
        }
        System.out.println("Reversed String: " + reversed);
    }
}

```

Output:

```
PS D:\Java\Java Program> java ReverseString.java
Reversed String: uDaD
PS D:\Java\Java Program> |
```

h) Right Angled Triangle

Code:

```
public class RightAngledTriangle {
    public static void main(String[] args) {
        int rows = 5;
        for (int i = 1; i <= rows; i++) {
            for (int j = 1; j <= i; j++) {
                System.out.print("* ");
            }
            System.out.println();
        }
    }
}
```

Output:


```
PS D:\Java\Java Program> java RightAngledTriangle.java
*
* *
* * *
* * * *
* * * * *
PS D:\Java\Java Program> |
```

i) Simple Interest

Code:

```
public class SimpleInterest {  
    public static void main(String[] args) {  
        double p = 1000, r = 5, t = 3;  
        double si = (p * r * t) / 100;  
        System.out.println("Simple Interest: " + si);  
    }  
}
```

Output:



```
PS D:\Java\Java Program> java SimpleInterest.java  
Simple Interest: 150.0  
PS D:\Java\Java Program> |
```

j) Sum of Natural Numbers

Code:

```
public class SumNaturalNumbers {  
    public static void main(String[] args) {  
        int n = 10, sum = 0;  
        for (int i = 1; i <= n; i++) {  
            sum += i;  
        }  
    }  
}
```

```
System.out.println("Sum of first " + n + " natural numbers: " + sum);  
}  
}
```

Output:

```
PS D:\Java\Java Program> java SumNaturalNumbers.java  
Sum of first 10 natural numbers: 55  
PS D:\Java\Java Program> |
```


	Inheritance	
--	-------------	--

4.) Single Inheritance

4.a) Employer

Code:

```
class Employee {
    String name;
    int empId;
    Employee(String name, int empId) {
        this.name = name;
        this.empId = empId;
    }
    void howDetails() {
        System.out.println("Employee Name: " + name);
        System.out.println("Employee ID: " + empId);
    }
}

class Manager extends Employee {
    String department;

    Manager(String name, int empId, String department) {
        super(name, empId);
        this.department = department;
    }

    void display() {
        showDetails();
        System.out.println("Department: " + department);
    }
}

public class Employer{
    public static void main(String[] args) {
        Manager m = new Manager("Alice Johnson", 1001, "HR");
        m.display();
    }
}
```

```
}  
}
```

OUTPUT:

```
D:\AMRITA\S-2\Java Program\inheritance>java Employer  
Employee Name: Alice Johnson  
Employee ID: 1001  
Department: HR
```

4.b) UserBankAccount.

CODE:

```
import java.util.Scanner;  
class BankAccount {  
    double balance;  
    BankAccount(double balance) {  
        this.balance = balance;  
    }  
    void deposit(double amount) {  
        balance += amount;  
        System.out.println("Deposited: " + amount + ", New Balance: " + balance);  
    }  
    void displayBalance() {  
        System.out.println("Current Balance: " + balance);  
    }  
}  
  
class SavingsAccount extends BankAccount {  
    double interestRate = 5.0;  
  
    SavingsAccount(double balance) {  
        super(balance);  
    }  
    void addInterest() {  
        double interest = (balance * interestRate) / 100;  
        balance += interest;  
        System.out.println("Interest added: " + interest + ", New Balance: " + balance);  
    }  
}  
public class UserBankAccount{
```

```

public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    System.out.print("Enter initial balance: ");
    double initialBalance = sc.nextDouble();
    SavingsAccount sa = new SavingsAccount(initialBalance);
    System.out.print("Enter amount to deposit: ");
    double depositAmount = sc.nextDouble();
    sa.deposit(depositAmount);
    sa.addInterest();
    sa.displayBalance();
    sc.close();
}
}

```

OUTPUT:

```

D:\AMRITA\S-2\Java Program\inheritance>java UserBankAccount
Enter initial balance: 1233
Enter amount to deposit: 2222
Deposited: 2222.0, New Balance: 3455.0
Interest added: 172.75, New Balance: 3627.75
Current Balance: 3627.75

```

5.Multilevel Inheritance

5a) Money.

CODE:

```

class Payment {
    void transaction() {
        System.out.println("Transaction initiated...");
    }
}

class OnlinePayment extends Payment {
    void processPayment() {
        System.out.println("Online Payment Processed.");
    }
}

class RefundablePayment extends OnlinePayment {
    void processRefund() {
        System.out.println("Refund Processed.");
    }
}

```

```

}
public class Money{
    public static void main(String[] args) {
        RefundablePayment rp = new RefundablePayment();
        rp.transaction();
        rp.processPayment();
        rp.processRefund();
    }
}

```

OUTPUT:

```

D:\AMRITA\S-2\Java Program\inheritance>java Money
Transaction initiated...
Online Payment Processed.
Refund Processed.

D:\AMRITA\S-2\Java Program\inheritance>

```

5b) Multi

CODE:

```

class Person {
    void display() {
        System.out.println("Person Details");
    }
}

class Employee extends Person {
    void work() {
        System.out.println("Employee is working.");
    }
}

class Intern extends Employee {
    void study() {
        System.out.println("Intern is studying.");
    }
}

public class Multi{
    public static void main(String[] args) {
        Intern intern = new Intern();
    }
}

```

```

    intern.display();
    intern.work();
    intern.study();
}
}

```

OUTPUT:

```

D:\AMRITA\S-2\Java Program\inheritance>java Multi
Person Details
Employee is working.
Intern is studying.

D:\AMRITA\S-2\Java Program\inheritance>

```

6.HIERARCHICAL INHERITANCE

6.a) TeacherSystem

CODE:

```

class Teacher {
    void teach() {
        System.out.println("Teacher is teaching...");
    }
}

class MathTeacher extends Teacher {
    void explainMath() {
        System.out.println("Math teacher is explaining algebra.");
    }
}

class ScienceTeacher extends Teacher {
    void explainScience() {
        System.out.println("Science teacher is explaining physics.");
    }
}

public class TeacherSystem {
    public static void main(String[] args) {
        MathTeacher mt = new MathTeacher();
        ScienceTeacher st = new ScienceTeacher();
    }
}

```

```

System.out.println("--- Math Teacher ---");
mt.teach();
mt.explainMath();
System.out.println("\n--- Science Teacher ---");
st.teach();
st.explainScience();
}
}

```

OUTPUT:

```

--- Math Teacher ---
Teacher is teaching...
Math teacher is explaining algebra.

--- Science Teacher ---
Teacher is teaching...
Science teacher is explaining physics.

D:\AMRITA\S-2\Java Program\inheritance>

```

6b) UniversityHierarchy

CODE:

```

import java.util.Scanner;

class University {
    String universityName;

    University(String name) {
        this.universityName = name;
    }

    void displayUniversity() {
        System.out.println("University: " + universityName);
    }
}

class Student extends University {
    String studentName;

```

```

int marks;

Student(String uniName, String studentName, int marks) {
    super(uniName);
    this.studentName = studentName;
    this.marks = marks;
}

void displayStudentDetails() {
    displayUniversity();
    System.out.println("Student Name: " + studentName);
    System.out.println("Marks Obtained: " + marks);
    if (marks >= 50) {
        System.out.println("Result: PASS");
    } else {
        System.out.println("Result: FAIL");
    }
}

}

class Professor extends University {
    String professorName;
    String subject;

    Professor(String uniName, String professorName, String subject) {
        super(uniName);
        this.professorName = professorName;
        this.subject = subject;
    }

    void displayProfessorDetails() {
        displayUniversity();
        System.out.println("Professor Name: " + professorName);
        System.out.println("Subject: " + subject);
    }
}

public class UniversityHierarchy {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter university name: ");

```

```

String uniName = sc.nextLine();
System.out.print("Enter student name: ");
String studentName = sc.nextLine();
System.out.print("Enter student marks: ");
int marks = sc.nextInt();
sc.nextLine();
System.out.print("Enter professor name: ");
String professorName = sc.nextLine();
System.out.print("Enter subject taught: ");
String subject = sc.nextLine();
Student student = new Student(uniName, studentName, marks);
Professor professor = new Professor(uniName, professorName, subject);
System.out.println("\n--- Student Details ---");
student.displayStudentDetails();
System.out.println("\n--- Professor Details ---");
professor.displayProfessorDetails();

sc.close();
}
}

```

OUTPUT:

```

D:\AMRITA\S-2\Java Program\inheritance>java UniversityHierarchy
Enter university name: Amrita
Enter student name: dadu
Enter student marks: 99
Enter professor name: suthir sir
Enter subject taught: oops

--- Student Details ---
University: Amrita
Student Name: dadu
Marks Obtained: 99
Result: PASS

--- Professor Details ---
University: Amrita
Professor Name: suthir sir
Subject: oops

```

7 .HYBRID INHERITANCE PROGRAMS

7a) BankSystem

CODE:

```
class Bank {
    void bankDetails() {
        System.out.println("Welcome to ABC Bank.");
    }
}

class Account extends Bank {
    void accountType() {
        System.out.println("This is a general bank account.");
    }
}

class LoanAccount extends Account {
    void loanDetails() {
        System.out.println("Loan account with 5% interest.");
    }
}

class SavingsAccount extends Account {
    void savingsDetails() {
        System.out.println("Savings account with 4% interest.");
    }
}

public class BankSystem {
    public static void main(String[] args) {
        LoanAccount la = new LoanAccount();
        SavingsAccount sa = new SavingsAccount();
        System.out.println("--- Loan Account Details ---");
        la.bankDetails();
        la.accountType();
        la.loanDetails();

        System.out.println("\n--- Savings Account Details ---");
        sa.bankDetails();
        sa.accountType();
    }
}
```

```
        sa.savingsDetails();
    }
}
```

OUTPUT:

```
--- Loan Account Details ---
Welcome to ABC Bank.
This is a general bank account.
Loan account with 5% interest.

--- Savings Account Details ---
Welcome to ABC Bank.
This is a general bank account.
Savings account with 4% interest.

D:\AMRITA\S-2\Java Program\inheritance>
```

7b) UniversitySystem

CODE:

```
class University {
    void universityInfo() {
        System.out.println("This is a top-ranked university.");
    }
}

class Department extends University {
    void departmentInfo() {
        System.out.println("Department: Computer Science");
    }
}

class Professor extends University {
    void professorInfo() {
        System.out.println("Professor: Dr. Smith, Expert in AI.");
    }
}

class Student extends Department {
    String studentName;

    Student(String name) {
        this.studentName = name;
    }

    void studentInfo() {
```

```

        System.out.println("Student Name: " + studentName);
    }
}
public class UniversitySystem {
    public static void main(String[] args) {
        Student s = new Student("John Doe");
        Professor p = new Professor();
        System.out.println("--- Student Details ---");
        s.universityInfo();
        s.departmentInfo();
        s.studentInfo();
        System.out.println("\n--- Professor Details ---");
        p.universityInfo();
        p.professorInfo();
    }
}

```

OUTPUT:

```

--- Student Details ---
This is a top-ranked university.
Department: Computer Science
Student Name: John Doe

--- Professor Details ---
This is a top-ranked university.
Professor: Dr. Smith, Expert in AI.

D:\AMRITA\S-2\Java Program\inheritance>

```

8. CONSTRUCTOR PROGRAMS

8a) PatientConstructor

CODE:

```
class Patient {
    String name;
    int age;

    Patient() {
        this.name = "Unknown";
        this.age = 0;
    }

    Patient(String name, int age) {
        this.name = name;
        this.age = age;
    }

    void showInfo() {
        System.out.println("Patient Name: " + name + ", Age: " + age);
    }
}

public class main PatientCostructor{
    public static void main(String[] args) {
        java.util.Scanner sc = new java.util.Scanner(System.in);
        System.out.print("Enter patient name: ");
        String patientName = sc.nextLine();
        System.out.print("Enter patient age: ");
        int patientAge = sc.nextInt();
        Patient p1 = new Patient();
        Patient p2 = new Patient(patientName, patientAge);
        p1.showInfo();
        p2.showInfo();
        sc.close();
    }
}
```

OUTPUT:

```
D:\AMRITA\S-2\Java Program\polymorphism>java PatientConstructor
Enter patient name: dheeraj
Enter patient age: 18
Patient Name: Unknown, Age: 0
Patient Name: dheeraj, Age: 18

D:\AMRITA\S-2\Java Program\polymorphism>
```

9. CONSTRUCTOR OVERLOADING PROGRAMS

9a) HotelConstructorOverloading.

CODE:

```
class Hotel {
    String hotelName;
    int rating;
    Hotel() {
        this.hotelName = "Default Hotel";
        this.rating = 3;
    }
    Hotel(String hotelName) {
        this.hotelName = hotelName;
        this.rating = 4;
    }
    Hotel(String hotelName, int rating) {
        this.hotelName = hotelName;
        this.rating = rating;
    }
    void showDetails() {
        System.out.println("Hotel: " + hotelName + ", Rating: " + rating + "
stars");
    }
}

public class HotelConstructorOverloading{
    public static void main(String[] args) {
        Hotel h1 = new Hotel();
        Hotel h2 = new Hotel("Grand Palace");
    }
}
```

```

        Hotel h3 = new Hotel("Seaside Resort", 5);
        h1.showDetails();
        h2.showDetails();
        h3.showDetails();
    }
}

```

OUTPUT:

```

D:\AMRITA\S-2\Java Program\polymorphism>java HotelConstructorOverloading
Hotel: Default Hotel, Rating: 3 stars
Hotel: Grand Palace, Rating: 4 stars
Hotel: Seaside Resort, Rating: 5 stars

D:\AMRITA\S-2\Java Program\polymorphism>

```

10. METHOD OVERLOADING PROGRAMS

10.a) Currency

CODE:

```

class CurrencyConverter {
    double convert(double amount, String currencyType) {
        if (currencyType.equalsIgnoreCase("USD")) {
            return amount * 1.1;
        } else if (currencyType.equalsIgnoreCase("EUR")) {
            return amount * 0.9;
        } else {
            return amount;
        }
    }
    double convert(double amount) {
        return amount * 75.0; /
    }
}

public class Currency{
    public static void main(String[] args) {
        java.util.Scanner sc = new java.util.Scanner(System.in);
        CurrencyConverter converter = new CurrencyConverter();
    }
}

```

```

        System.out.print("Enter amount in INR: ");
        double amount = sc.nextDouble();
        sc.nextLine();
        System.out.print("Enter currency type (USD/EUR): ");
        String currencyType = sc.nextLine();
        System.out.println("Converted amount: " + converter.convert(amount,
currencyType));
        System.out.println("Default conversion to INR: " +
converter.convert(amount));
        sc.close();
    }
}

```

OUTPUT:

```

D:\AMRITA\S-2\Java Program\polymorphism>java Currency
Enter amount in INR: 200000
Enter currency type (USD/EUR): india
Converted amount: 200000.0
Default conversion to INR: 1.5E7

D:\AMRITA\S-2\Java Program\polymorphism>|

```

10.b) LibraryMethodOverloading

CODE:

```

class Library {
    void search(String title) {
        System.out.println("Searching book: " + title);
    }
    void search(String title, String author) {
        System.out.println("Searching book: " + title + " by " + author);
    }
    void search(int year) {
        System.out.println("Searching books published in: " + year);
    }
}

```

```

public class LibraryMethoOverloading{
    public static void main(String[] args) {
        Library lib = new Library();
        lib.search("Java Programming");
        lib.search("Data Structures", "Robert Lafore");
        lib.search(2021);
    }
}

```

OUTPUT:

```

D:\AMRITA\S-2\Java Program\polymorphism>java LibraryMethodOverloading
Searching book: Java Programming
Searching book: Data Structures by Robert Lafore
Searching books published in: 2021

D:\AMRITA\S-2\Java Program\polymorphism>

```

11. METHOD OVERRIDING PROGRAMS

11a) RobotMethodOverriding

CODE:

```

class Robot {
    void action() {
        System.out.println("Performing generic robot tasks.");
    }
}

class CleaningRobot extends Robot {
    void action() {
        System.out.println("Cleaning the floor efficiently.");
    }
}

public class RobotMethodOverriding{
    public static void main(String[] args) {
        Robot r = new Robot();
        CleaningRobot c = new CleaningRobot();
        r.action();
        c.action();
    }
}

```



```
}
```

OUTPUT:

```
D:\AMRITA\S-2\Java Program\polymorphism>java RobotMethodOverriding
Performing generic robot tasks.
Cleaning the floor efficiently.
```

```
D:\AMRITA\S-2\Java Program\polymorphism>|
```

11b) TransportMethodOverriding

CODE:

```
class Transport {
    void travel() {
        System.out.println("Traveling via transport.");
    }
}

class Airplane extends Transport {
    void travel() {
        System.out.println("Flying in the sky.");
    }
}

class Train extends Transport {
    void travel() {
        System.out.println("Moving on railway tracks.");
    }
}

public class TransportMethodOverriding{
    public static void main(String[] args) {
        Transport t1 = new Airplane();
        Transport t2 = new Train();

        t1.travel();
        t2.travel();
    }
}
```

OUTPUT:

```
D:\AMRITA\S-2\Java Program\polymorphism>java TransportMethodOverriding
Flying in the sky.
Moving on railway tracks.
```

```
D:\AMRITA\S-2\Java Program\polymorphism>|
```

	ABSTRACTION	
--	--------------------	--

12) INTERFACE PROGRAMS

12.a) Bottle

```
interface Refillable {
    void refill();
    void use();
}
class WaterBottle implements Refillable {
    private int waterLevel = 0;
    public void refill() {
        waterLevel = 100;
        System.out.println("Water bottle refilled.");
    }
    public void use() {
        if (waterLevel > 0) {
            System.out.println("Drinking water...");
            waterLevel -= 50;
        } else {
            System.out.println("Bottle is empty! Please refill.");
        }
    }
}

public class Bottle{
    public static void main(String[] args) {
        WaterBottle bottle = new WaterBottle();
        bottle.use();
        bottle.refill();
    }
}
```

```

        bottle.use();
        bottle.use();
    }
}

```

OUTPUT:

```

D:\AMRITA\S-2\Java Program\Interface>java Bottle
Bottle is empty! Please refill.
Water bottle refilled.
Drinking water...
Drinking water...

D:\AMRITA\S-2\Java Program\Interface>|

```

12b) Charging

CODE:

```

interface Chargeable {
    void charge();
    void stopCharging();
    int batteryLevel();
}

class ElectricCar implements Chargeable {
    private int battery = 50;
    public void charge() {
        battery = 100;
        System.out.println("Electric car is fully charged.");
    }
    public void stopCharging() {
        System.out.println("Charging stopped at " + battery + "%.");
    }
    public int batteryLevel() {
        return battery;
    }
}

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

```

```

    ElectricCar car = new ElectricCar();
    System.out.println("Battery level: " + car.batteryLevel() + "%");
    car.charge();
    car.stopCharging();
}
}

```

OUTPUT:

```

D:\AMRITA\S-2\Java Program\Interface>java Charging
Battery level: 50%
Electric car is fully charged.
Charging stopped at 100%.

D:\AMRITA\S-2\Java Program\Interface>

```

12.c) Phones

CODE:

```

interface Communicable {
    void sendMessage(String message);
    void receiveMessage(String message);
    void call(String contact);
}

class MobilePhone implements Communicable {
    public void sendMessage(String message) {
        System.out.println("Sending message: " + message);
    }
    public void receiveMessage(String message) {
        System.out.println("Received message: " + message);
    }
    public void call(String contact) {
        System.out.println("Calling " + contact + "...");
    }
}

public class Phones{
    public static void (String[] args) {
        MobilePhone phone = new MobilePhone();
        phone.sendMessage("Hello!");
        phone.receiveMessage("Hey, how are you?");
    }
}

```

```
    phone.call("Hussain");  
}  
}
```

OUTPUT:

```
D:\AMRITA\S-2\Java Program\Interface>java Phones  
Sending message: Hello!  
Received message: Hey, how are you?  
Calling Hussain...  
  
D:\AMRITA\S-2\Java Program\Interface>|
```

12.d) Tracker.

CODE:

```
interface Trackable {  
    void startTracking();  
    void stopTracking();  
    String getLocation();  
}  
  
class GPSDevice implements Trackable {  
    private String location = "Unknown";  
    public void startTracking() {  
        location = "Downtown, City Center";  
        System.out.println("GPS tracking started.");  
    }  
    public void stopTracking() {  
        System.out.println("GPS tracking stopped.");  
    }  
    public String getLocation() {  
        return location;  
    }  
}  
  
public class Tracker{  
    public static void main(String[] args) {  
        GPSDevice gps = new GPSDevice();  
        gps.startTracking();  
        System.out.println("Current location: " + gps.getLocation());  
    }  
}
```

```
        gps.stopTracking();
    }
}
```

OUTPUT:

```
D:\AMRITA\S-2\Java Program\Interface>java Tracker
GPS tracking started.
Current location: Downtown, City Center
GPS tracking stopped.

D:\AMRITA\S-2\Java Program\Interface>|
```

13) ABSTRACT CLASS PROGRAMS

13a) AIR

CODE:

```
abstract class Spacecraft {
    abstract void launch();
    abstract void land();
}

class Rocket extends Spacecraft {
    void launch() {
        System.out.println("Rocket is launching.");
    }
    void land() {
        System.out.println("Rocket is landing back on Earth.");
    }
}

public class Air{
    public static void main(String[] args) {
        Rocket rocket = new Rocket();
        rocket.launch();
        rocket.land();
    }
}
```

OUTPUT:

```
D:\AMRITA\S-2\Java Program\Abstract>java Air
Rocket is launching.
Rocket is landing back on Earth.

D:\AMRITA\S-2\Java Program\Abstract>|
```

13b)Bank

CODE:

```
abstract class BankAccount {
    abstract void deposit(double amount);
    abstract void withdraw(double amount);
    void accountDetails() {
        System.out.println("Bank accounts store money securely.");
    }
}

class SavingsAccount extends BankAccount {
    private double balance = 0;
    void deposit(double amount) {
        balance += amount;
        System.out.println("Deposited: " + amount);
    }
    void withdraw(double amount) {
        if (amount <= balance) {
            balance -= amount;
            System.out.println("Withdrawn: " + amount);
        } else {
            System.out.println("Insufficient balance.");
        }
    }
}

public class Bank{
    public static void main(String[] args) {
        SavingsAccount account = new SavingsAccount();
        account.accountDetails();
        account.deposit(500);
        account.withdraw(200);
    }
}
```

OUTPUT:

```
D:\AMRITA\S-2\Java Program\Abstract>java Bank
Bank accounts store money securely.
Deposited: 500.0
Withdrawn: 200.0

D:\AMRITA\S-2\Java Program\Abstract>|
```

13c) Cook

CODE:

```
abstract class CookingAppliance {
    abstract void cook();
    abstract void stop();
}
class Microwave extends CookingAppliance {
    void cook() {
        System.out.println("Microwave is heating food.");
    }
    void stop() {
        System.out.println("Microwave is stopped.");
    }
}
public class Cook{
    public static void main(String[] args) {
        Microwave microwave = new Microwave();
        microwave.cook();
        microwave.stop();
    }
}
```

OUTPUT:

```
D:\AMRITA\S-2\Java Program\Abstract>java Cook
Microwave is heating food.
Microwave is stopped.
D:\AMRITA\S-2\Java Program\Abstract>
```


13d)Music

CODE:

```
abstract class MusicalInstrument {
    abstract void play();
    abstract void tune();
    void commonFeature() {
        System.out.println("All instruments produce sound.");
    }
}

class Piano extends MusicalInstrument {
    void play() {
        System.out.println("Playing the piano.");
    }
    void tune() {
        System.out.println("Tuning the piano.");
    }
}

public class Music{
    public static void main(String[] args) {
        Piano piano = new Piano();
        piano.commonFeature();
        piano.tune();
        piano.play();
    }
}
```

OUTPUT:

```
D:\AMRITA\S-2\Java Program\Abstract>java Music
All instruments produce sound.
Tuning the piano.
Playing the piano.

D:\AMRITA\S-2\Java Program\Abstract>|
```

	ENCAPSULATION	
--	----------------------	--

14) ENCAPSULATION PROGRAMS

14a) Main.

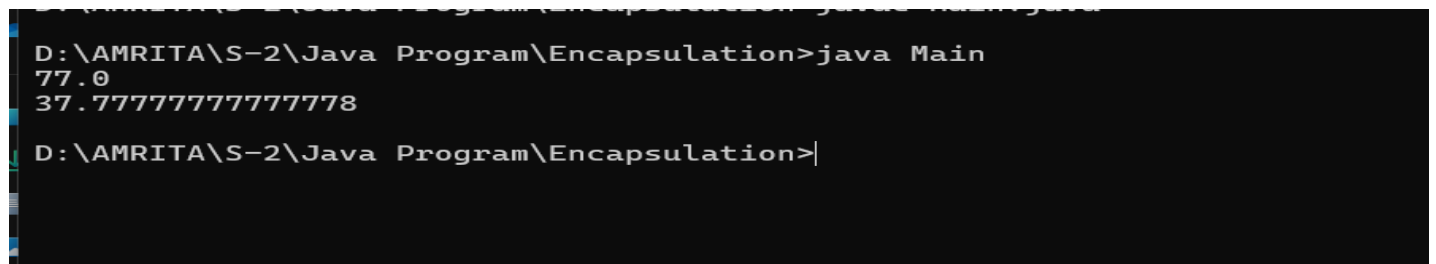
CODE:

```
class TemperatureConverter {
    private double celsius;

    public TemperatureConverter(double celsius) {
        this.celsius = celsius;
    }
    public void setTemperatureCelsius(double celsius) {
        this.celsius = celsius;
    }
    public double getTemperatureCelsius() {
        return celsius;
    }
    public double getTemperatureFahrenheit() {
        return (celsius * 9/5) + 32;
    }
    public void setTemperatureFahrenheit(double fahrenheit) {
        this.celsius = (fahrenheit - 32) * 5/9;
    }
}

public class Main{
    public static void main(String[] args) {
        TemperatureConverter temp = new TemperatureConverter(25);
        System.out.println(temp.getTemperatureFahrenheit());
        temp.setTemperatureFahrenheit(100);
        System.out.println(temp.getTemperatureCelsius());
    }
}
```

OUTPUT:



```
D:\AMRITA\S-2\Java Program\Encapsulation>java Main
77.0
37.77777777777778
D:\AMRITA\S-2\Java Program\Encapsulation>
```

14b) Making

CODE:

```
class Recipe {
    private String name;
    private String[] ingredients;
    private String instructions;

    public Recipe(String name, String[] ingredients, String instructions) {
        this.name = name;
        this.ingredients = ingredients;
        this.instructions = instructions;
    }
    public String getName() {
        return name;
    }
    public String[] getIngredients() {
        return ingredients;
    }
    public String getInstructions() {
        return instructions;
    }
    public void displayRecipe() {
        System.out.println("Recipe: " + name);
        System.out.println("Ingredients: ");
        for (String ingredient : ingredients) {
            System.out.println("- " + ingredient);
        }
        System.out.println("Instructions: " + instructions);
    }
}

public class Making {
    public static void main(String[] args) {
        String[] ingredients = {"2 cups flour", "1 cup sugar", "1/2 cup butter", "2 eggs"};
        Recipe recipe = new Recipe("Cake", ingredients, "Mix all ingredients and bake at 350°F for 30 minutes.");
        recipe.displayRecipe();
    }
}
```

OUTPUT:

```
D:\AMRITA\S-2\Java Program\Encapsulation>java MakingRecipe
Recipe: Cake
Ingredients:
- 2 cups flour
- 1 cup sugar
- 1/2 cup butter
- 2 eggs
Instructions: Mix all ingredients and bake at 350°F for 30 minutes.

D:\AMRITA\S-2\Java Program\Encapsulation>
```

14c) Mobile

CODE:

```
class Contact {
    private String name;
    private String phoneNumber;
    private String email;

    public Contact(String name, String phoneNumber, String email) {
        this.name = name;
        this.phoneNumber = phoneNumber;
        this.email = email;
    }
    public String getName() {
        return name;
    }
    public String getPhoneNumber() {
        return phoneNumber;
    }
    public String getEmail() {
        return email;
    }
    public void displayContactInfo() {
        System.out.println("Name: " + name + ", Phone: " + phoneNumber + ", Email: " +
email);
    }
}

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

```

        Contact contact = new Contact("Dadu", "9381083122",
"dadukolimi75@gmail.com");
        contact.displayContactInfo();
    }
}

```

OUTPUT:

```

:\AMRITA\S-2\Java Program\Encapsulation>java Mobile
ame: Dadu, Phone: 9381083122, Email: dadukolimi75@gmail.com

:\AMRITA\S-2\Java Program\Encapsulation>|

```

14d)Shop

CODE:

```

class Product {
    private String productName;
    private double price;
    private int quantity;
    public Product(String productName, double price, int quantity) {
        this.productName = productName;
        this.price = price;
        this.quantity = quantity;
    }
    public void setPrice(double price) {
        if (price > 0) {
            this.price = price;
        } else {
            System.out.println("Price must be positive.");
        }
    }
    public double getPrice() {
        return price;
    }
    public void setQuantity(int quantity) {
        if (quantity >= 0) {
            this.quantity = quantity;
        } else {
            System.out.println("Quantity cannot be negative.");
        }
    }
}

```

```

    }
}
public int getQuantity() {
    return quantity;
}
public void displayProductInfo() {
    System.out.println("Product: " + productName + ", Price: " + price + ", Quantity: "
+ quantity);
}
}
public class Shop {
    public static void main(String[] args) {
        Product product = new Product("Laptop", 1200.00, 5);
        product.displayProductInfo();
        product.setPrice(1100.00);
        product.setQuantity(10);
        product.displayProductInfo();
    }
}

```

OUTPUT:

```

D:\AMRITA\S-2\Java Program\Encapsulation>java Shop
Product: Laptop, Price: 1200.0, Quantity: 5
Product: Laptop, Price: 1100.0, Quantity: 10

D:\AMRITA\S-2\Java Program\Encapsulation>

```

	PACKAGES	
--	-----------------	--

15) User Defined Packages

15.a) User Defined Packages-Contact

CODE:

Contact.java
package contactmanagement;

```

public class Contact {
    private String name;
    private String phoneNumber;
    public Contact(String name, String phoneNumber) {
        this.name = name;
        this.phoneNumber = phoneNumber;
    }
    public String getName() {
        return name;
    }
    public String getPhoneNumber() {
        return phoneNumber;
    }

    @Override
    public String toString() {
        return "Contact [Name: " + name + ", Phone Number: " + phoneNumber
+ "]\n";
    }
}

```

ContactManager.java

```

package contactmanagement;
public class ContactManager {
    private Contact contact
    public void addContact(String name, String phoneNumber) {
        contact = new Contact(name, phoneNumber);
    }
    public void displayContact() {
        if (contact != null) {
            System.out.println(contact);
        } else {
            System.out.println("No contact found.");
        }
    }
}

```

ContactMain.java

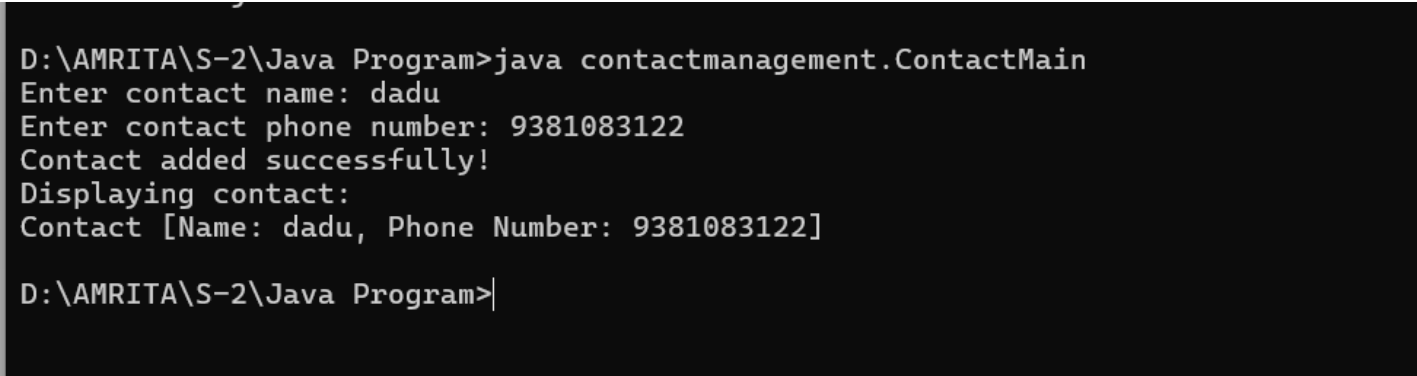
```
package contactmanagement;

import java.util.Scanner;

public class ContactMain {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        ContactManager contactManager = new ContactManager();
        System.out.print("Enter contact name: ");
        String name = scanner.nextLine();
        System.out.print("Enter contact phone number: ");
        String phoneNumber = scanner.nextLine();
        contactManager.addContact(name, phoneNumber);
        System.out.println("Contact added successfully!");
        System.out.println("Displaying contact:");
        contactManager.displayContact();

        scanner.close();
    }
}
```

OUTPUT:



```
D:\AMRITA\S-2\Java Program>java contactmanagement.ContactMain
Enter contact name: dadu
Enter contact phone number: 9381083122
Contact added successfully!
Displaying contact:
Contact [Name: dadu, Phone Number: 9381083122]

D:\AMRITA\S-2\Java Program>|
```

15b) User Defined Packages-Task Management

CODE:

Task.java


```
package taskmanagement;
```

```
public class Task {  
    private String title;  
    private String description;  
    public Task(String title, String description) {  
        this.title = title;  
        this.description = description;  
    }  
    public String getTitle() {  
        return title;  
    }  
    public String getDescription() {  
        return description;  
    }  
    @Override  
    public String toString() {  
        return "Task [Title: " + title + ", Description: " + description + "];"  
    }  
}
```

TaskManagement.java

```
package taskmanagement;
```

```
public class TaskManager {  
    private Task task;  
  
    public void addTask(String title, String description) {  
        task = new Task(title, description);  
    }  
  
    public void displayTask() {  
        if (task != null) {  
            System.out.println(task);  
        } else {  
            System.out.println("No task found.");  
        }  
    }  
}
```

TaskMain.java

```
package taskmanagement;

import java.util.Scanner;

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

        System.out.print("Enter task title: ");
        String title = scanner.nextLine();

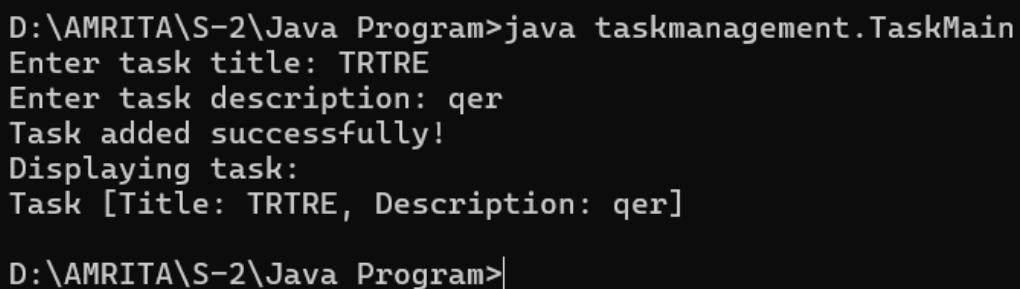
        System.out.print("Enter task description: ");
        String description = scanner.nextLine();

        taskManager.addTask(title, description);
        System.out.println("Task added successfully!");

        System.out.println("Displaying task:");
        taskManager.displayTask();

        scanner.close();
    }
}
```

OUTPUT:



```
D:\AMRITA\S-2\Java Program>java taskmanagement.TaskMain
Enter task title: TRTRE
Enter task description: qer
Task added successfully!
Displaying task:
Task [Title: TRTRE, Description: qer]

D:\AMRITA\S-2\Java Program>
```

15)Built in Packages

15c) SumOfIntegers

CODE:

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.time.LocalDate;
import java.util.ArrayList;
import java.util.List;

class SumOfIntegers {
    public static void main(String[] args) {
        String filePath = "numbers.txt";
        List<Integer> numbers = new ArrayList<>();
        int sum = 0;

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

        try (BufferedReader reader = new BufferedReader(new FileReader(filePath))) {
            String line;
            while ((line = reader.readLine()) != null) {
                try {
                    int number = Integer.parseInt(line);
                    numbers.add(number);
                    sum += number;
                } catch (NumberFormatException e) {
                    System.out.println("Invalid number format in line: " + line);
                }
            }
        } catch (IOException e) {
            System.out.println("Error reading the file: " + e.getMessage());
        }

        System.out.println("Numbers from the file: " + numbers);
        System.out.println("Sum of the numbers: " + sum);
    }
}
```

OUTPUT:

```
D:\AMRITA\S-2\Java Program\Built in>java SumOfIntegers
Today's Date: 2025-04-04
Error reading the file: numbers.txt (The system cannot find the file specified)
Numbers from the file: []
Sum of the numbers: 0

D:\AMRITA\S-2\Java Program\Built in>
```

15d)Random Numbers

CODE:

```
import java.util.ArrayList;
import java.util.List;
import java.util.Random;

class RandomNumber {
    public static void main(String[] args) {
        Random random = new Random();
        List<Integer> numbers = new ArrayList<>();
        int count = 10;
        for (int i = 0; i < count; i++) {
            int randomNumber = random.nextInt(100);
            numbers.add(randomNumber);
        }
        double average =
numbers.stream().mapToInt(Integer::intValue).average().orElse(0.0);

        System.out.println("Generated Random Numbers: " + numbers);
        System.out.println("Average: " + average);
    }
}
```

OUTPUT:

```
D:\AMRITA\S-2\Java Program\Built in>java RandomNumber
Generated Random Numbers: [64, 51, 9, 6, 43, 21, 19, 83, 44, 16]
Average: 35.6

D:\AMRITA\S-2\Java Program\Built in>
```

16) **EXCEPTION HANDLING PROGRAMS**

6.a) ArrayExample

CODE:

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

```
public class ArrayExample {  
    public static void main(String[] args) {  
        int[] numbers = {1, 2, 3, 4, 5};  
        Scanner scanner = new Scanner(System.in);  
        System.out.print("Enter an index (0-4): ");  
        int index = scanner.nextInt();  
  
        try {  
            System.out.println("Value at index " + index + ": " + numbers[index]);  
        } catch (ArrayIndexOutOfBoundsException e) {  
            System.out.println("Error: Index out of bounds.");  
        } finally {  
            scanner.close();  
        }  
    }  
}
```

OUTPUT:

16b) DivisionExample

CODE:

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

```
public class DivisionExample {  
    public static void main(String[] args) {  
        Scanner scanner = new Scanner(System.in);  
        System.out.print("Enter numerator: ");  
        int numerator = scanner.nextInt();  
        System.out.print("Enter denominator: ");  
        int denominator = scanner.nextInt();  
  
        try {  
            int result = numerator / denominator;
```

```

        System.out.println("Result: " + result);
    } catch (ArithmeticException e) {
        System.out.println("Error: Cannot divide by zero.");
    } finally {
        scanner.close();
    }
}
}

```

OUTPUT:

```

D:\AMRITA\S-2\Java Program\exception>java DivisionExample
Enter numerator: 4
Enter denominator: 2
Result: 2

D:\AMRITA\S-2\Java Program\exception>|

```

16c) NumberFormatException

CODE:

```

import java.util.Scanner;

public class NumberFormatException {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a number: ");
        String input = scanner.nextLine();

        try {
            int number = Integer.parseInt(input);
            System.out.println("You entered: " + number);
        } catch (NumberFormatException e) {
            System.out.println("Error: Invalid number format.");
        } finally {
            scanner.close();
        }
    }
}

```

OUTPUT:

```
D:\AMRITA\S-2\Java Program\exception>java NumberFormatExample
Enter a number: 20
You entered: 20

D:\AMRITA\S-2\Java Program\exception>|
```

16d)File read

CODE:

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

public class FileReadExample {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the file name to read: ");
        String fileName = scanner.nextLine();

        try {
            Scanner fileScanner = new Scanner(new File(fileName));
            while (fileScanner.hasNextLine()) {
                System.out.println(fileScanner.nextLine());
            }
            fileScanner.close();
        } catch (FileNotFoundException e) {
            System.out.println("Error: File not found.");
        } finally {
            scanner.close();
        }
    }
}
```

OUTPUT:

```
D:\AMRITA\S-2\Java Program\exception>java FileReadExample
Enter the file name to read: air.java
Error: File not found.

D:\AMRITA\S-2\Java Program\exception>|
```

17) FILE HANDLING PROGRAMS

17a) FileAppendExample

CODE:

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

public class FileAppendExample {
    public static void main(String[] args) {
        try {
            FileWriter writer = new FileWriter("example.txt", true);
            writer.write("\nAppending new data to the file.");
            writer.close();
            System.out.println("Data appended successfully.");
        } catch (IOException e) {
            System.out.println("An error occurred while appending to the file.");
            e.printStackTrace();
        }
    }
}
```

OUTPUT:

```
D:\AMRITA\S-2\Java Program\file handling>java FileAppendExample
Data appended successfully.

D:\AMRITA\S-2\Java Program\file handling>|
```

17b) FileDeleteExample

CODE:

```
import java.io.File;

public class FileDeleteExample {
    public static void main(String[] args) {
        File file = new File("example.txt");
        if (file.delete()) {
            System.out.println("File deleted successfully: " + file.getName());
        } else {
            System.out.println("Failed to delete the file.");
        }
    }
}
```



```
}  
}  
}
```

OUTPUT:

```
D:\AMRITA\S-2\Java Program\file handling>java FileDeleteExample  
File deleted successfully: example.txt  
  
D:\AMRITA\S-2\Java Program\file handling>|
```

17c) FileReadExample

CODE:

```
import java.io.File;  
import java.io.FileNotFoundException;  
import java.util.Scanner;  
  
public class FileReadExample {  
    public static void main(String[] args) {  
        try {  
            File file = new File("example.txt");  
            Scanner scanner = new Scanner(file);  
            while (scanner.hasNextLine()) {  
                System.out.println(scanner.nextLine());  
            }  
            scanner.close();  
        } catch (FileNotFoundException e) {  
            System.out.println("File not found.");  
            e.printStackTrace();  
        }  
    }  
}
```

OUTPUT:

```
D:\AMRITA\S-2\Java Program\file handling>java FileReadExample  
File not found.
```

17d) FileWriteExample

CODE:

```
import java.io.FileWriter;
import java.io.IOException;
public class FileWriteExample {
    public static void main(String[] args) {
        try {
            FileWriter writer = new FileWriter("example.txt");
            writer.write("Hello, this is a file write example in Java!");
            writer.close();
            System.out.println("File written successfully.");
        } catch (IOException e) {
            System.out.println("An error occurred while writing to the file.");
            e.printStackTrace();
        }
    }
}
```

OUTPUT:

```
D:\AMRITA\S-2\Java Program\file handling>java FileWriteExample
File written successfully.

D:\AMRITA\S-2\Java Program\file handling>
```