LAB RECORD

23CSE111- Object Oriented Programming

Submitted by

CH.SC.U4CSE24062- G NAVADEEP

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.U4CSE24062– G Navadeep*** in **"Computer Science and Engineering"** is a Bonafide record of the work carried out under my guidance and supervision at Amrita School of Computing, Chennai.

This Lab examination held on / /2025

Internal Examiner 1 Internal Examiner 2

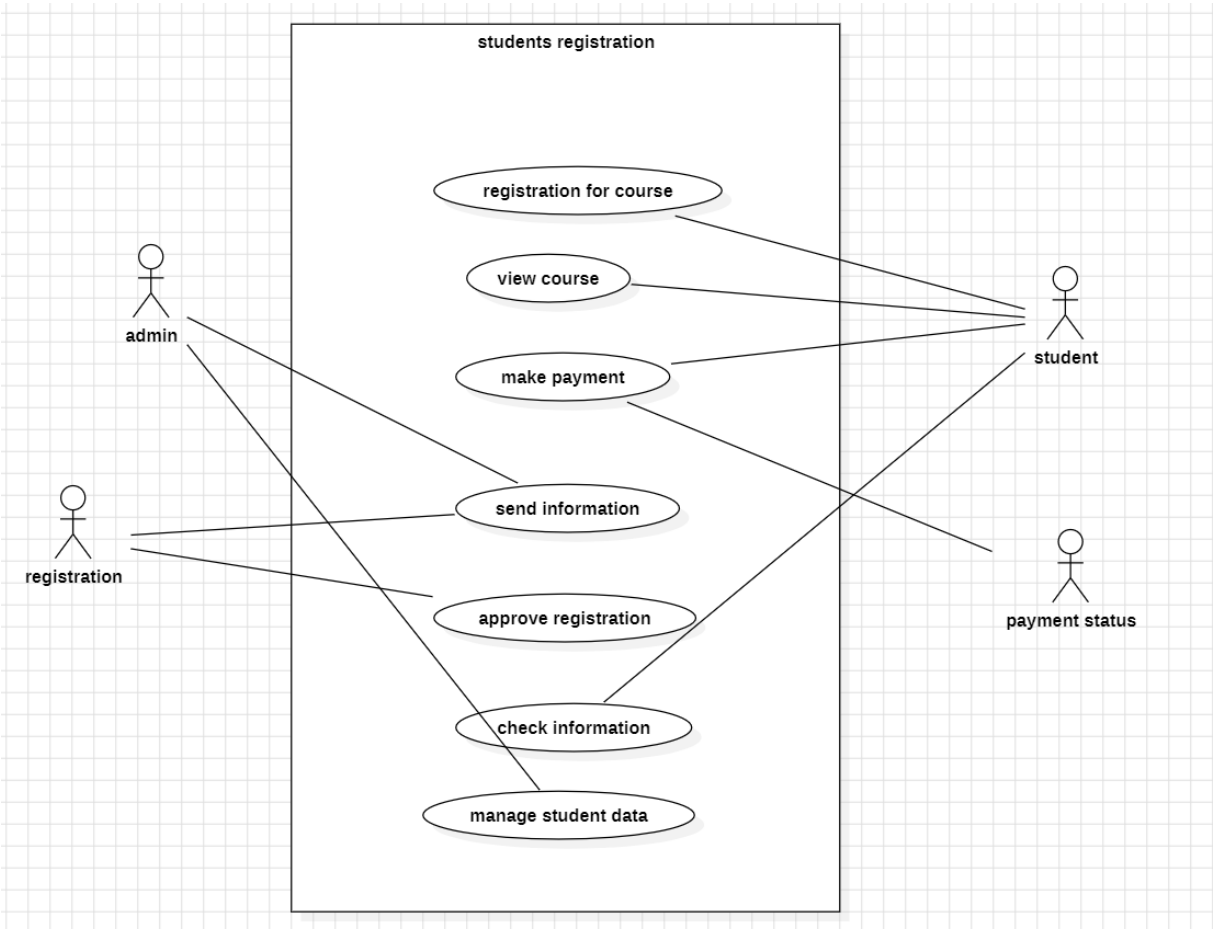
INDEX

S.NO	TITLE	PAGE.NO
	UML DIAGRAM	
1.	**TITLE OF UML DIAGRAM -1**	
	1.a)Use Case Diagram	6
	1.b)Class Diagram	7
	1.c) Sequence Diagram	8
	1.d) object Diagram	9
	1.e)collaboration Diagram	10
2.	**TITLE OF UML DIAGRAM -2**	
	2.a) Use Case Diagram	11
	2.b) Class Diagram	12
	2.c) Sequence Diagram	13
	2.d)object Diagram	14
	2.e)Collabration	15
3.	**BASIC JAVA PROGRAMS**	
	3.a)check even or odd	16
	3.b)find quotient and remainder	17
	3.c)switch case	18
	3.d)print pattern	19
	3.e) TO check prime number	20
	3.f) to print table	21
	3.g)swaping of two numbers	22
	3.h) QUADRATIC EQUATION ROOTS:	23
	3.i) fibonacciseries:	24
	3.j) gcd of two numbers	25
	INHERITANCE	
4.	**SINGLE INHERITANCE PROGRAMS**	
	4.a)Employedeveloper	29
	4.b)Machineprinter	30

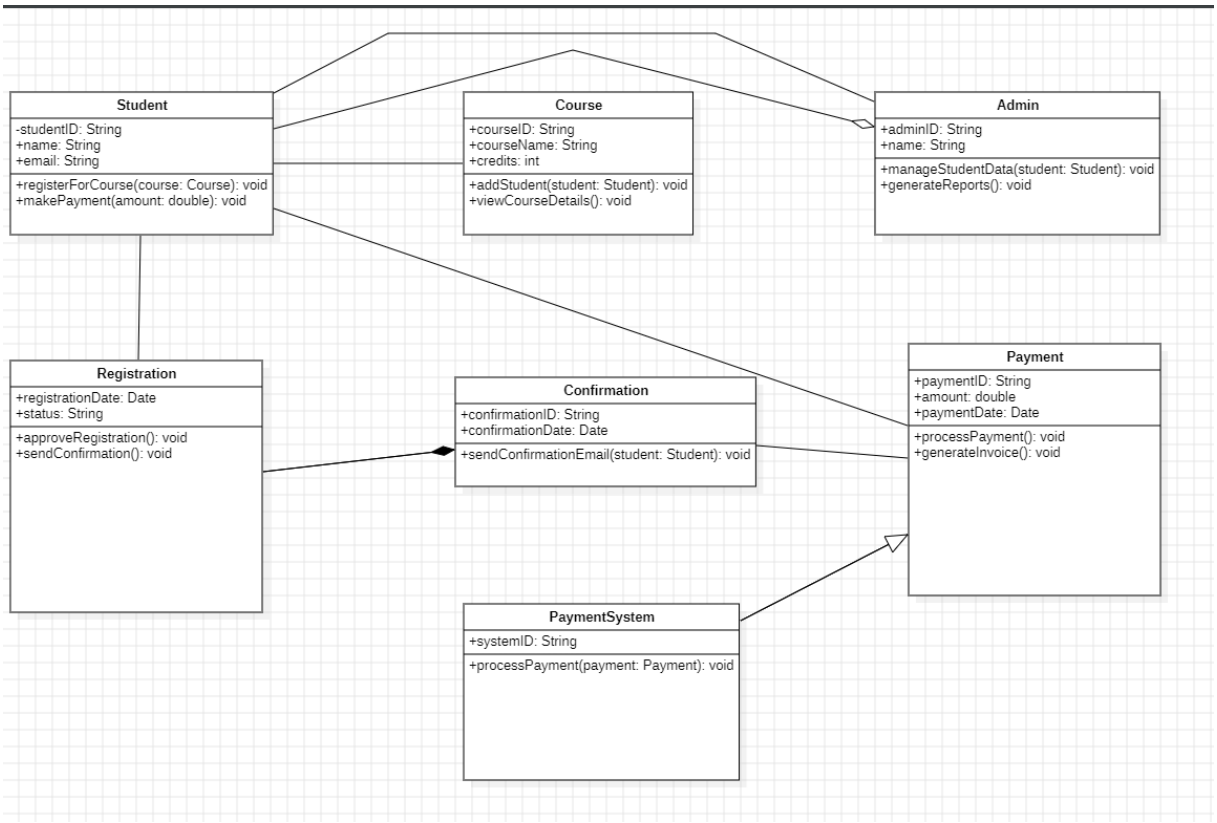
5.	**MULTILEVEL INHERITANCE PROGRAMS**	
	5.a)studentgraduater	31-32
	5.b)device	32-33
6.	**HIERARCHICAL INHERITANCE PROGRAMS**	
	6.a)Appliance	34-35
	6.b)Game	35-36
7.	**HYBRID INHERITANCE PROGRAMS**	
	7.a)Person	36-39
	7.b)Smart device	38-39
	POLYMORPHISM	
8.	**CONSTRUCTOR OVERWRITE PROGRAMS**	
	8.a)student constructor	39-40
9.	**CONSTRUCTOR OVERLOADING PROGRAMS**	
	9.a)STUDENT NAME	41-42
10.	**METHOD OVERLOADING PROGRAMS**	
	10.a)calculator addition	42-43
	10.b)display	43-44
11.	**METHOD OVERRIDING PROGRAMS**	
	11.a) E Commerce	45-46
	11.b)Bank	46-47
	ABSTRACTION	
12.	**INTERFACE PROGRAMS**	47-48
	12.a)program1	48
	12.b)print message	49-50
	12.c) online exam	50-51
	12.d)tax calculation	51-52
13.	**ABSTRACT CLASS PROGRAMS**	
	13.a)draw shape	51-52
	13.b)details of object	52-53
	13.c)employee salary	53-54
	13.d)Vechile	54-55
	ENCAPSULATION	
14.	**ENCAPSULATION PROGRAMS**	
	14.a)print name	56-57
	14.b)get and set method	57-58
	14.c)Employee	58-59
	14.d)Car	59-60
15.	**PACKAGES PROGRAMS**	
	15.a)User Defined Packages	60
	15.b)User Defined Packages	61
	15.c)Built – in Package(3 Packages)	62-63
	15.d)Built – in Package(3 Packages)	63

16.	**EXCEPTION HANDLING PROGRAMS**	
	16.a)Array Exception	63
	16.b) Zero Exception	63-64
	16.c)Custom Exception	64-65
	16.d)finally Block	65
17.	**FILE HANDLING PROGRAMS**	
	17.a)ARRAYLIST	66-67
	17.b)TaskManager	67-70
	17.c)Write file	70-71
	17.d)file Writer	71-73

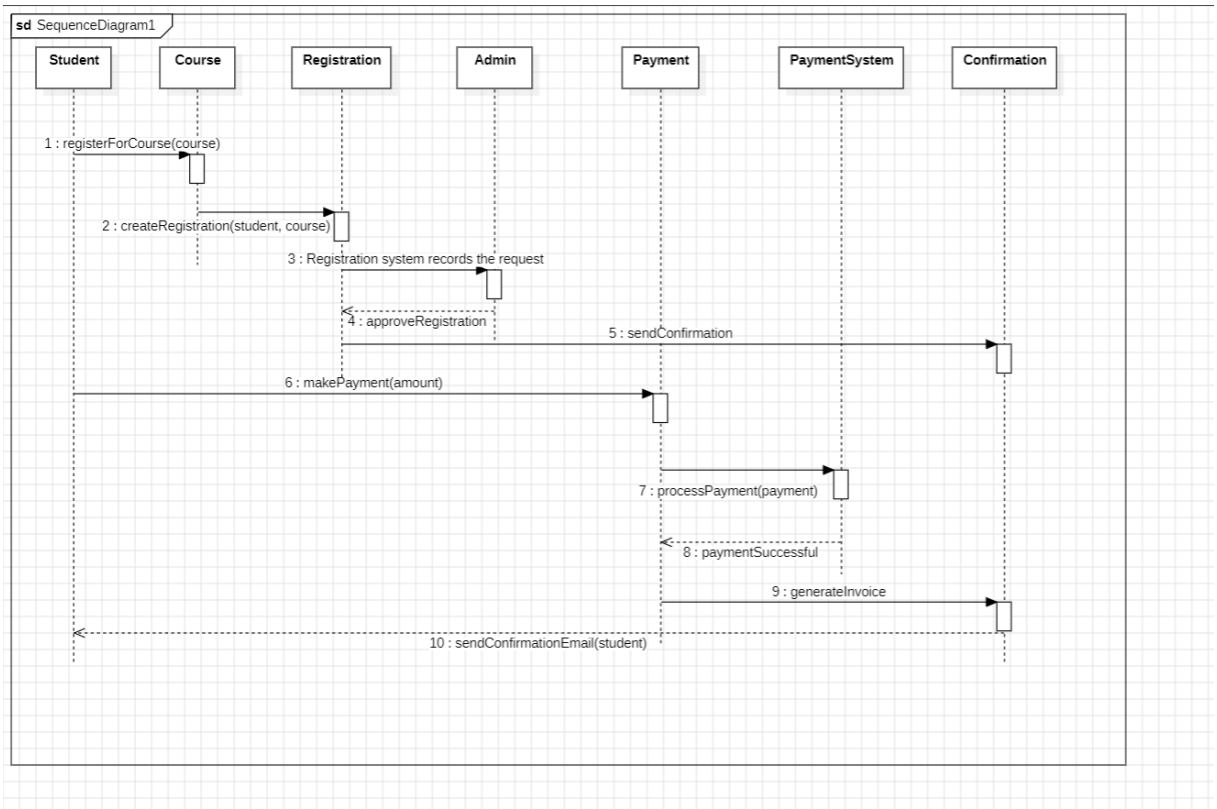
UML DIAGRAMS
STUDENT REGISTRATION
1A.USE CASE DIAGRAM



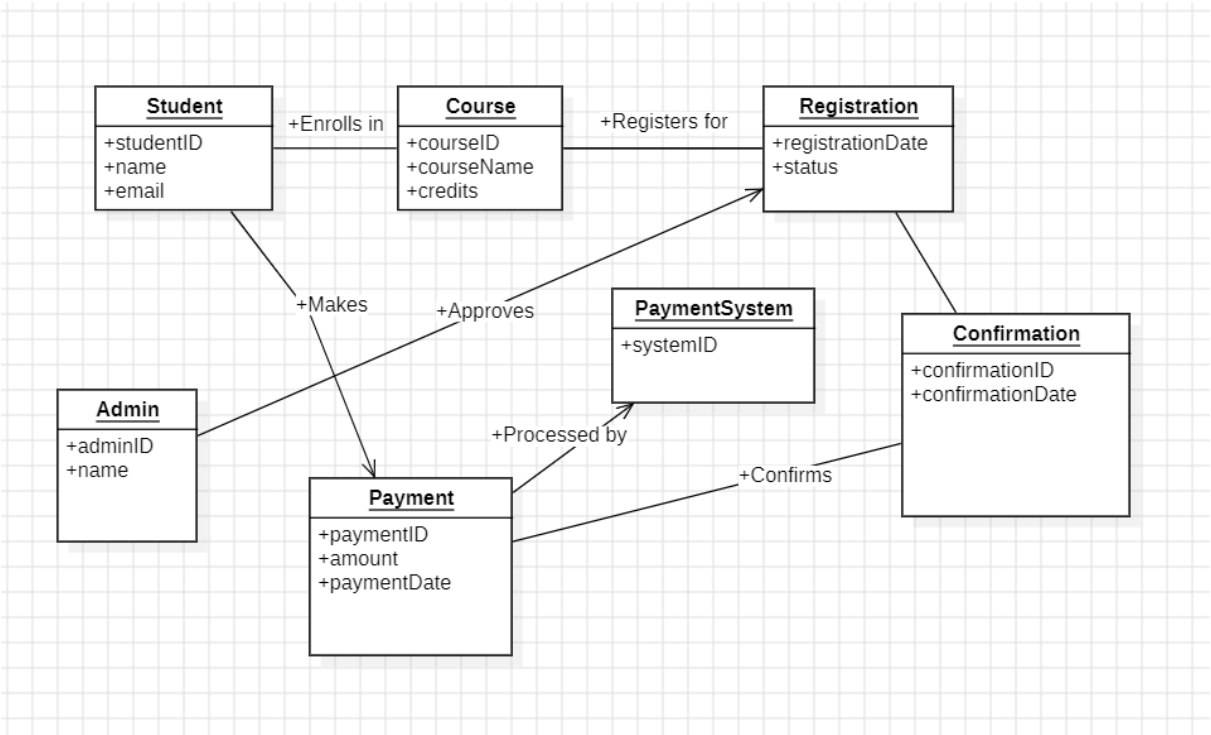
1B.CLASS DIAGRAM



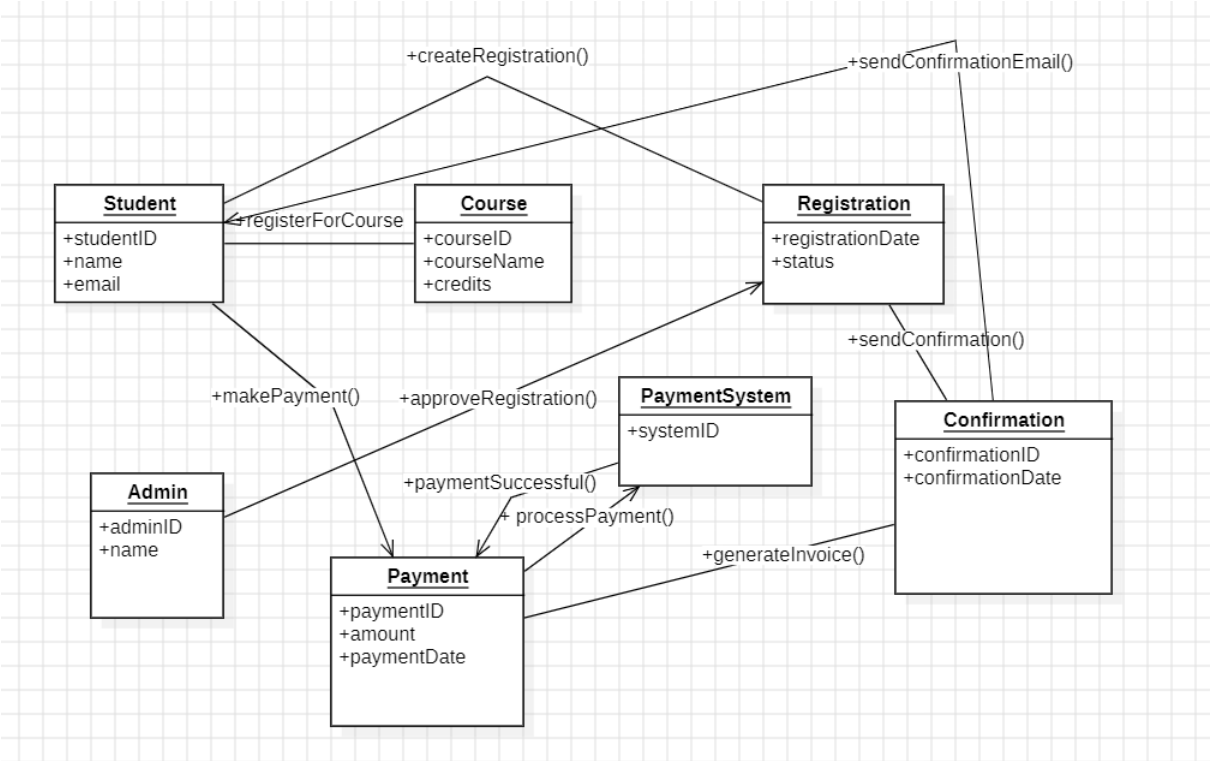
1C.SEQUENCE DIAGRAM:



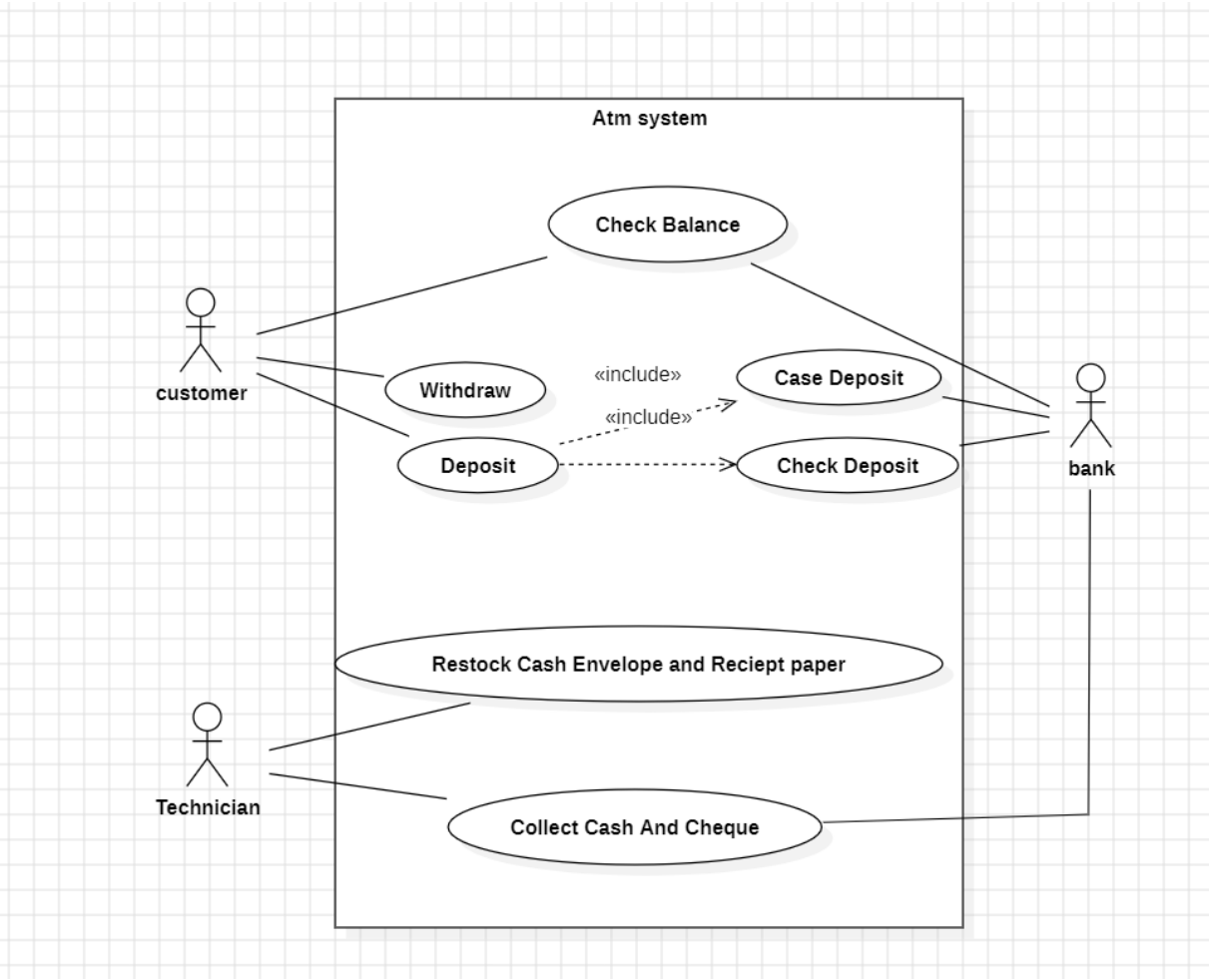
1D.OBJECT DIAGRAM:



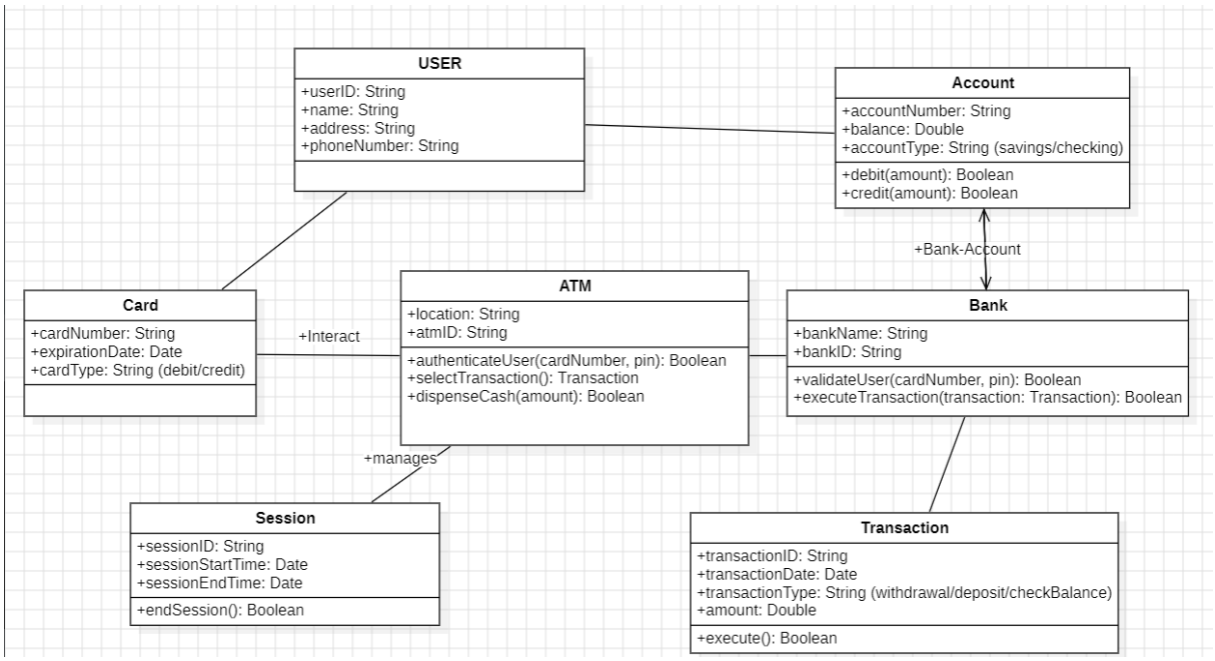
1E.COLLABORATION DIAGRAM:



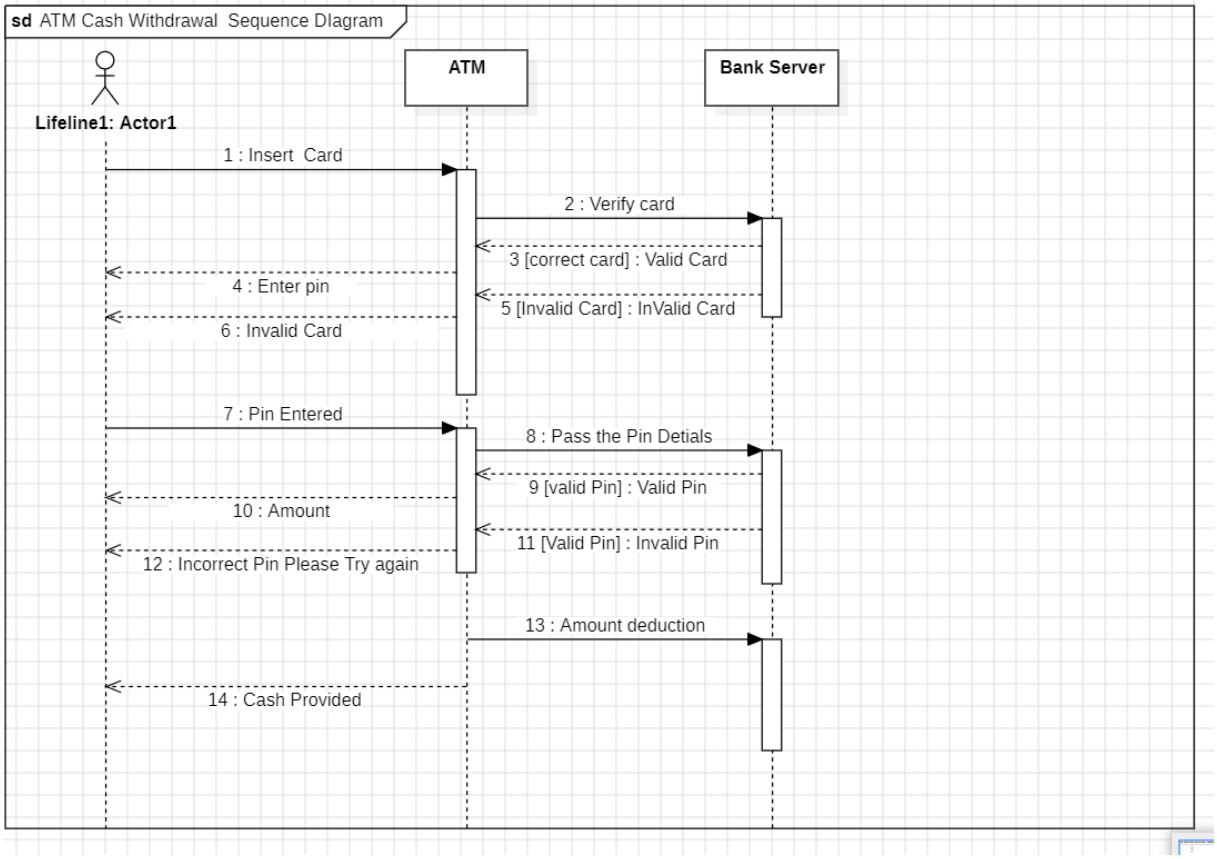
ATM
2A.USE CASE DIAGRAM.



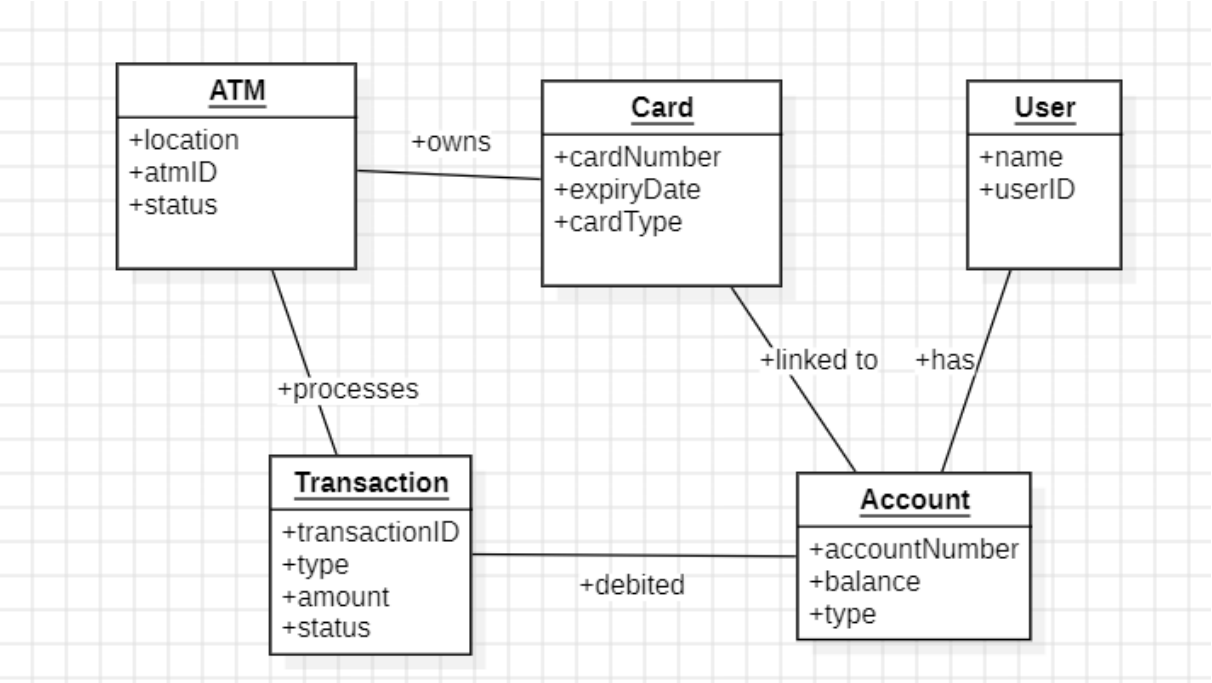
2B.CLASS DIAGRAM.



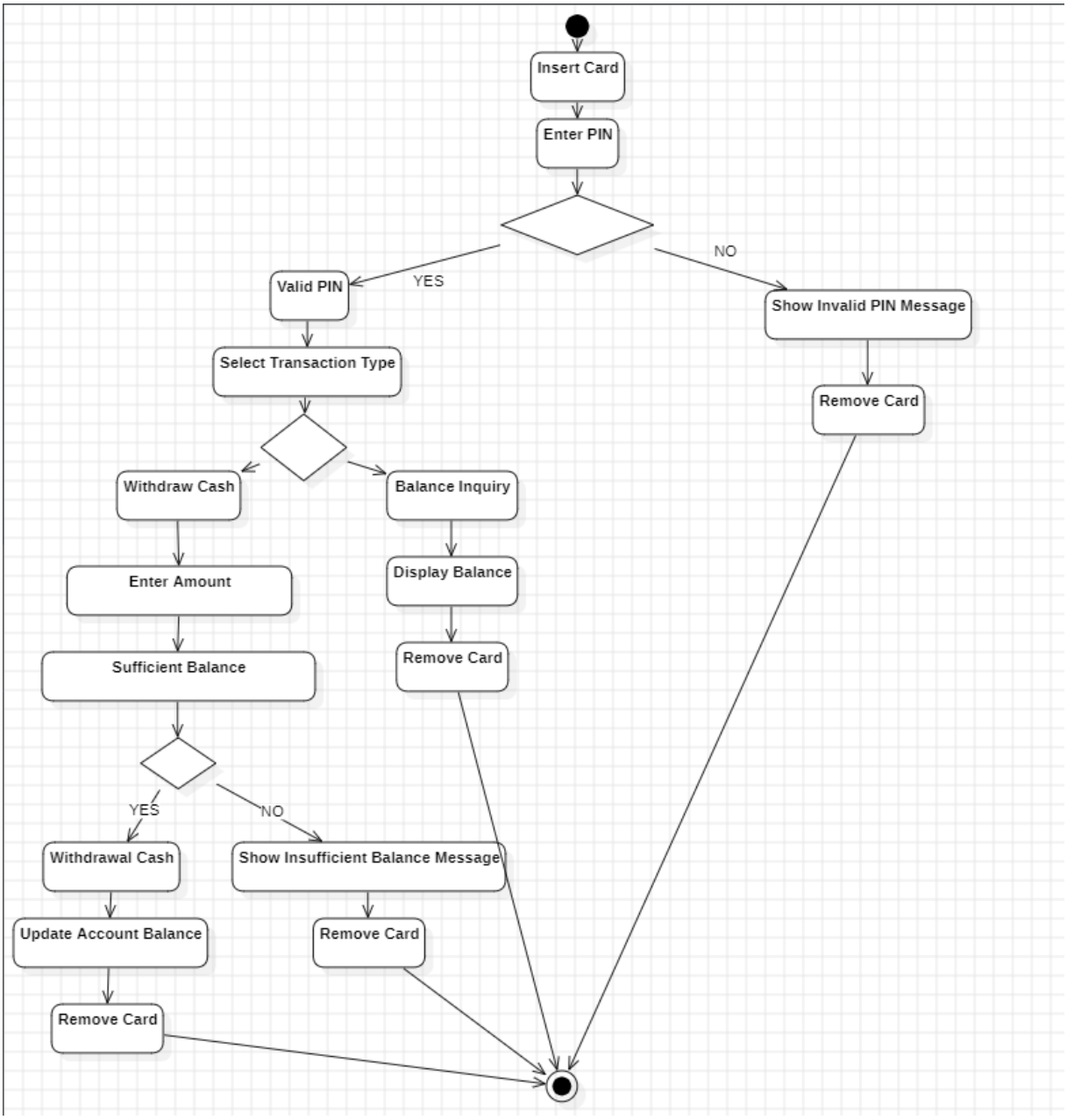
2C.SEQUENCE DIAGRAM.



2D.OBJECT DIAGRAM.



2E.ACTIVITY DIAGRAM.



BASIC JAVA PROGRAMS:

3.a) even or odd

CODE:

```
import java.util.Scanner;
public class Even
{
  public static void main(String[] args)
  {
    int n;
    Scanner s = new Scanner(System.in);
    System.out.print("Enter the number you want to check:");
    n = s.nextInt();
    if(n % 2 == 0)
    {
      System.out.println("The given number "+n+" is Even ");
    }
    else
    {
      System.out.println("The given number "+n+" is Odd ");
}
  }
}
```

OUTPUT:

```
C:\Users\deepu>javac Even.java

C:\Users\deepu>java Even
Enter the number you want to check:45
The given number 45 is Odd

C:\Users\deepu>
```

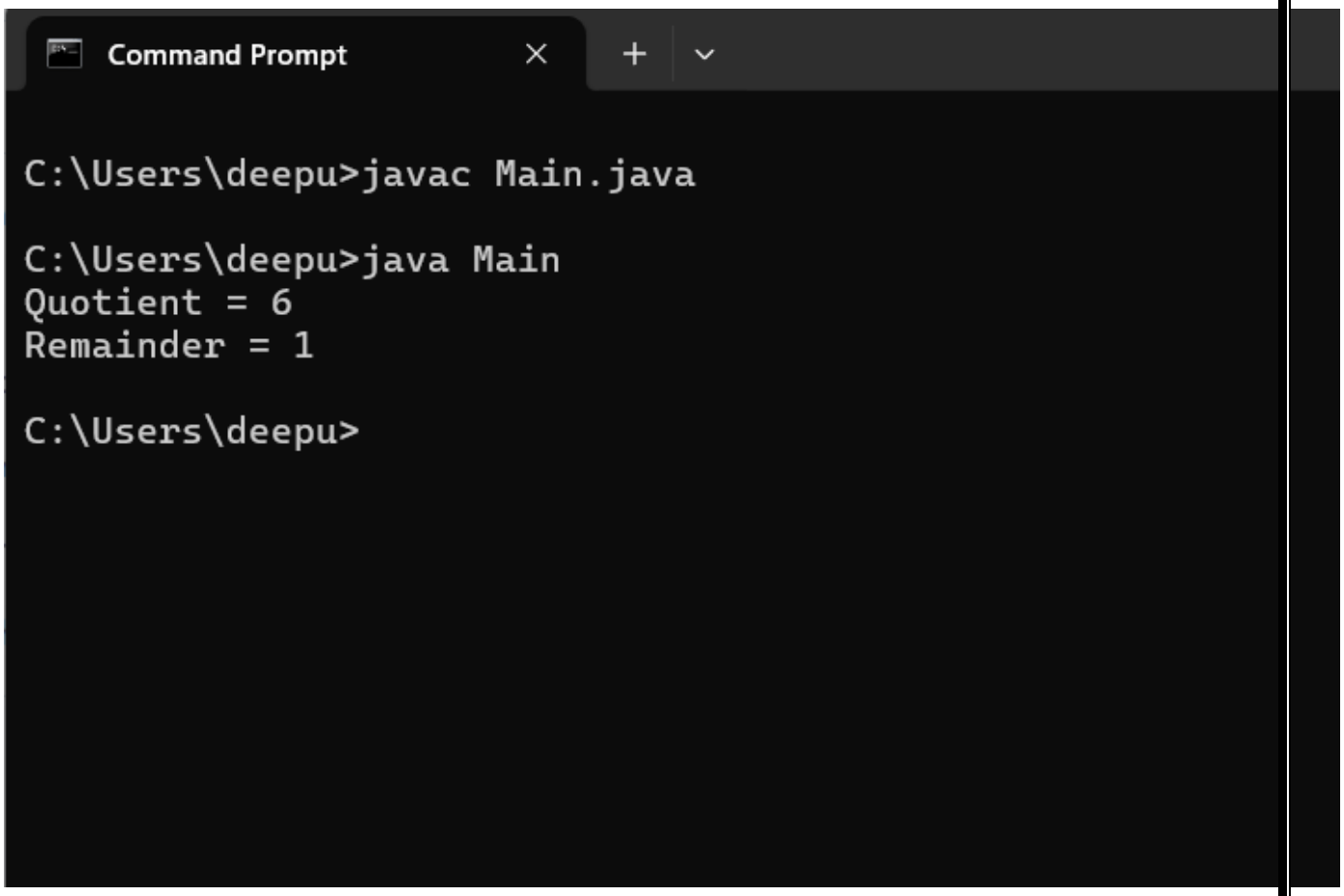
3b)Aim:

Code:

public class Main{

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

    int dividend = 25, divisor = 4;

    int quotient = dividend / divisor;
    int remainder = dividend % divisor;

    System.out.println("Quotient = " + quotient);
    System.out.println("Remainder = " + remainder);
  }
}
```

OUTPUT:



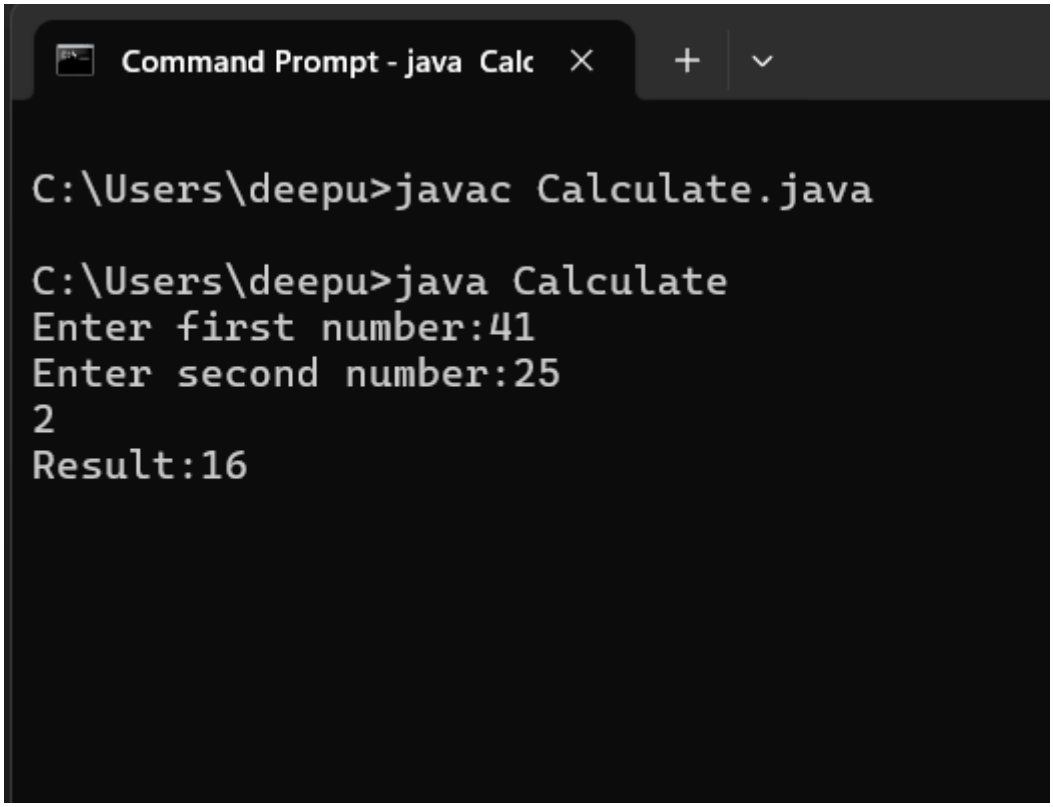
3c) AIM: SWITCH CASE

CODE:

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

```
public class Calculate
{
  public static void main(String[] args)
  {
    int m, n, opt, add, sub, mul;
    double div;
    Scanner s = new Scanner(System.in);
    System.out.print("Enter first number:");
    m = s.nextInt();
    System.out.print("Enter second number:");
    n = s.nextInt();
    while(true)
    {
      opt = s.nextInt();
      switch(opt)
      {
        case 1:
        add = m + n;
        System.out.println("Result:"+add);
        break;

        case 2:
        sub = m - n;
        System.out.println("Result:"+sub);
        break;

        case 3:
        mul = m * n;
        System.out.println("Result:"+mul);
        break;

        case 4:
        div = (double)m / n;
        System.out.println("Result:"+div);
        break;

        case 5:
        System.exit(0);
      }
    }
  }
}
```

Output:



3d) AIM: pattern print

CODE:

```
class p1 {
   public static void main(String[] args) {
      for (int i = 0; i <= 5; i = i + 1) {
         for (int j = 0; j <= 5; j = j + 1) {
            System.out.print("*");
         }
         System.out.println("*");
      }
   }
}
```

OUTPUT:

```
C:\Users\deepu>javac p1.java

C:\Users\deepu>java p1
*******
*******
*******
*******
*******
*******

C:\Users\deepu>
```

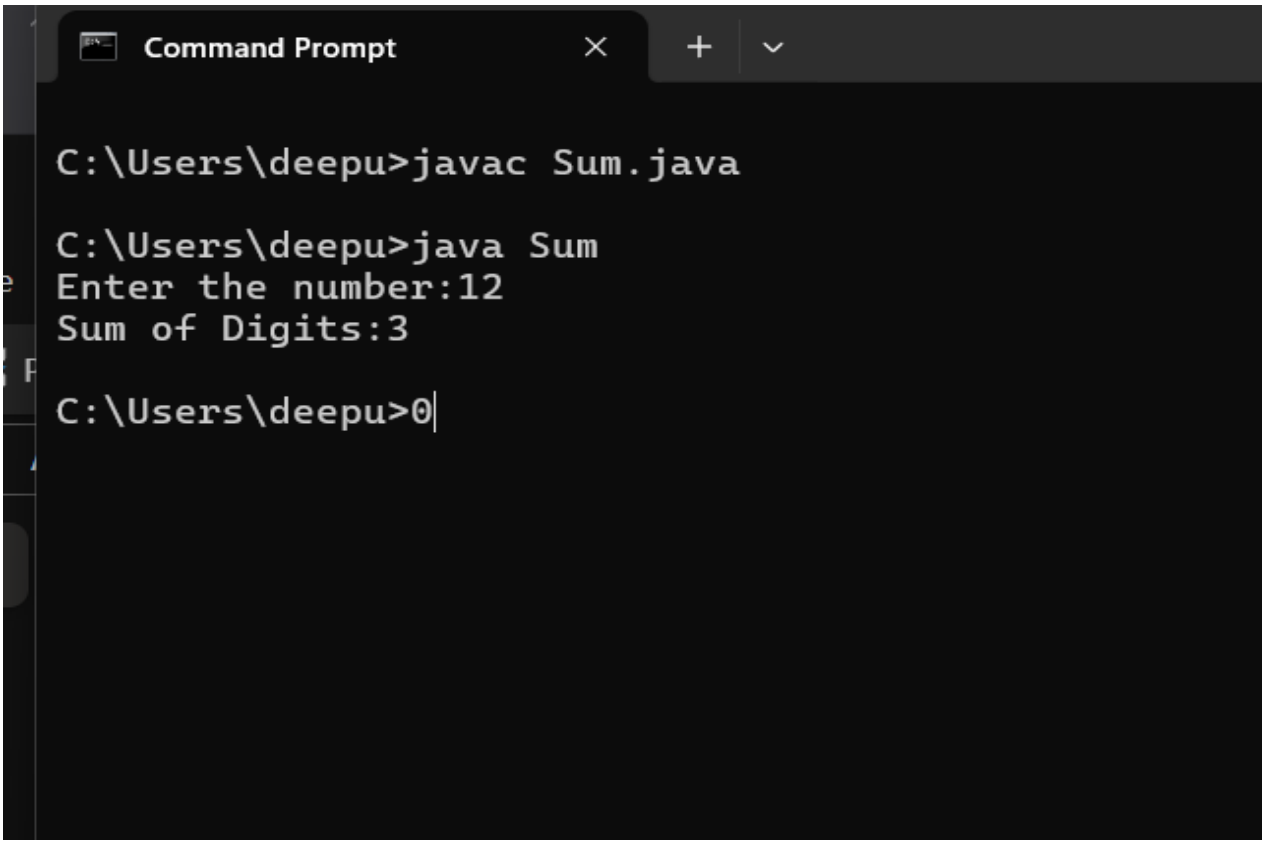
3d) WHILE LOOP

CODE:

```
import java.util.Scanner;
public class Sum
{
   public static void main(String args[])
   {
      int m, n, sum = 0;
      Scanner s = new Scanner(System.in);
      System.out.print("Enter the number:");
      m = s.nextInt();
      while(m > 0)
      {
         n = m % 10;
         sum = sum + n;
         m = m / 10;
      }
```

```
        System.out.println("Sum of Digits:"+sum);
    }
}
```

OUTPUT:



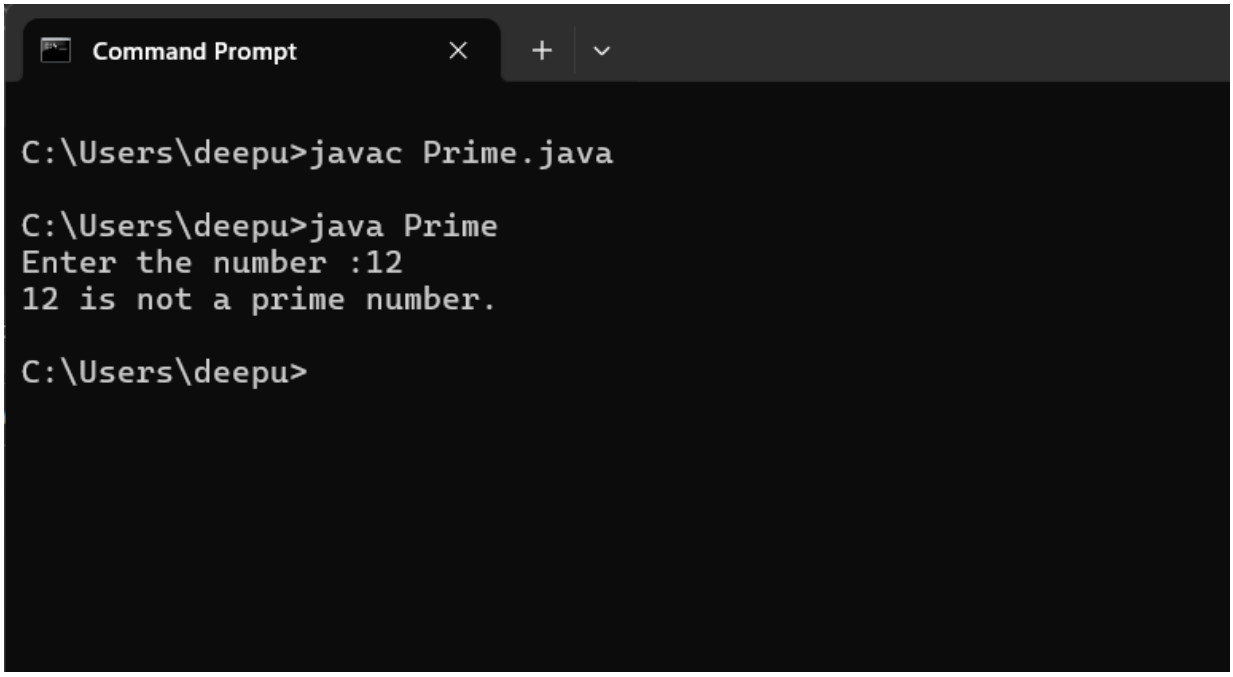
3e)TO CHECK PRIME NUMBER OR NOT

CODE:

```
import java.util.Scanner;
public class Prime
{
    public static void main(String args[])
    {
        int j, num, flag = 0;
        System.out.print("Enter the number :");
        Scanner s = new Scanner(System.in);
        num = s.nextInt();
```

```
      for( j = 2; j < num; j++)
      {
         if(num % j == 0)
         {
            flag = 0;
            break;
         }
         else
         {
            flag = 1;
         }
      }
      if(flag == 1)
      {
         System.out.println(""+num+" is a prime number.");
      }
      else
      {
         System.out.println(""+num+" is not a prime number.");
      }
   }
}
```

OUTPUT:



3f) to print table

Code:

```
import java.util.Scanner;
public class d { public static void main(String[] args) {
 Scanner s = new Scanner(System.in);
 System.out.print("Enter number:");
 int n=s.nextInt();
for(int i=1; i <= 10; i++)
{ System.out.println(n+" * "+i+" = "+n*i);
 }
}
 }
```

OUTPUT:

```
C:\Users\deepu>javac d.java

C:\Users\deepu>java d
Enter number:12
12 * 1 = 12
12 * 2 = 24
12 * 3 = 36
12 * 4 = 48
12 * 5 = 60
12 * 6 = 72
12 * 7 = 84
12 * 8 = 96
12 * 9 = 108
12 * 10 = 120

C:\Users\deepu>
```

3g) swaping of two numbers

CODE:

```
import java.util.Scanner;
public class a
{
   public static void main(String args[])
   {
      int m, n, temp;
      Scanner s = new Scanner(System.in);
      System.out.print("Enter the first number:");
      m = s.nextInt();
      System.out.print("Enter the second number:");
      n = s.nextInt();
      temp = m;
      m = n;
      n = temp;
```

```
        System.out.println("After Swapping");
        System.out.println("First number:"+m);
        System.out.println("Second number:"+n);
    }
}
```

OUTPUT:

```
C:\Users\deepu>javac a.java

C:\Users\deepu>java a
Enter the first number: 1
Enter the second number: 2
After Swapping:
First number: 2
Second number: 1
```

3h)QUADRATIC EQUATION ROOTS:

Code:

```
public class Main {

 public static void main(String[] args) {
   double a = 2.3, b = 4, c = 5.6;
   double root1, root2;
   double discriminant = b * b - 4 * a * c;
   if (discriminant > 0) {

     // two real and distinct roots
     root1 = (-b + Math.sqrt(discriminant)) / (2 * a);
     root2 = (-b - Math.sqrt(discriminant)) / (2 * a);

     System.out.format("root1 = %.2f and root2 = %.2f", root1, root2);
   }
```

```
  else if (discriminant == 0) {

    root1 = root2 = -b / (2 * a);
    System.out.format("root1 = root2 = %.2f;", root1);
  }

  // if discriminant is less than zero
  else {

    // roots are complex number and distinct
    double real = -b / (2 * a);
    double imaginary = Math.sqrt(-discriminant) / (2 * a);
    System.out.format("root1 = %.2f+%.2fi", real, imaginary);
    System.out.format("\nroot2 = %.2f-%.2fi", real, imaginary);
  }
 }
}
```

Output:

```
Microsoft Windows [Version 10.0.26100.3476]
(c) Microsoft Corporation. All rights reserved.

C:\Users\deepu> javac Main.java

C:\Users\deepu>java Main
root1 = -0.87+1.30i
root2 = -0.87-1.30i
C:\Users\deepu>
```

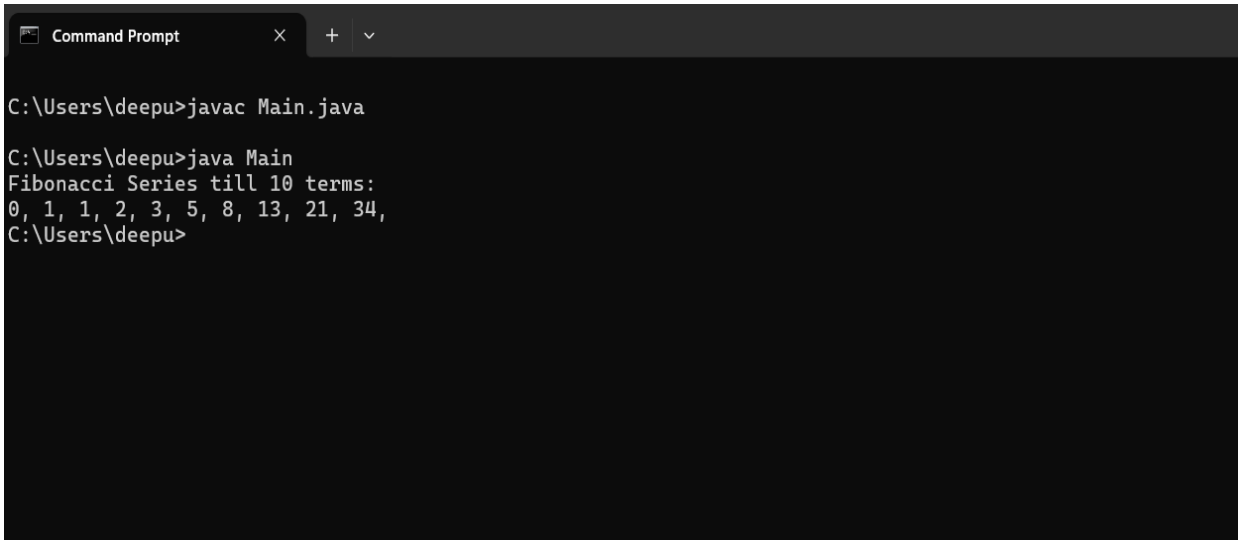
3i) fibonacciseries:

Code:

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

  int n = 10, firstTerm = 0, secondTerm = 1;
```

```
    System.out.println("Fibonacci Series till " + n + " terms:");

    for (int i = 1; i <= n; ++i) {
      System.out.print(firstTerm + ", ");

      // compute the next term
      int nextTerm = firstTerm + secondTerm;
      firstTerm = secondTerm;
      secondTerm = nextTerm;
    }
  }
}
```

Output:



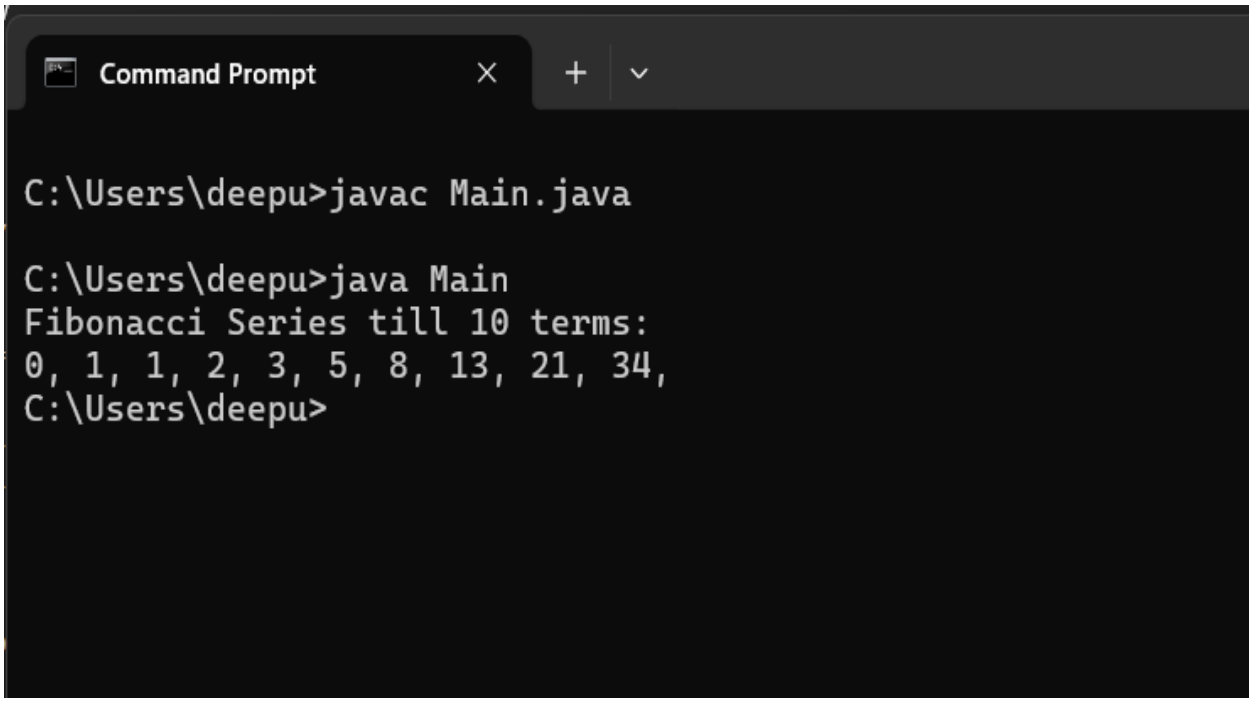
3j) gcd of two numbers

Code:

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

    // find GCD between n1 and n2
    int n1 = 81, n2 = 153;

    // initially set to gcd
    int gcd = 1;

    for (int i = 1; i <= n1 && i <= n2; ++i) {

      // check if i perfectly divides both n1 and n2
      if (n1 % i == 0 && n2 % i == 0)
```

```
      gcd = i;
    }

    System.out.println("GCD of " + n1 +" and " + n2 + " is " + gcd);
  }
}
```

OUTPUT:



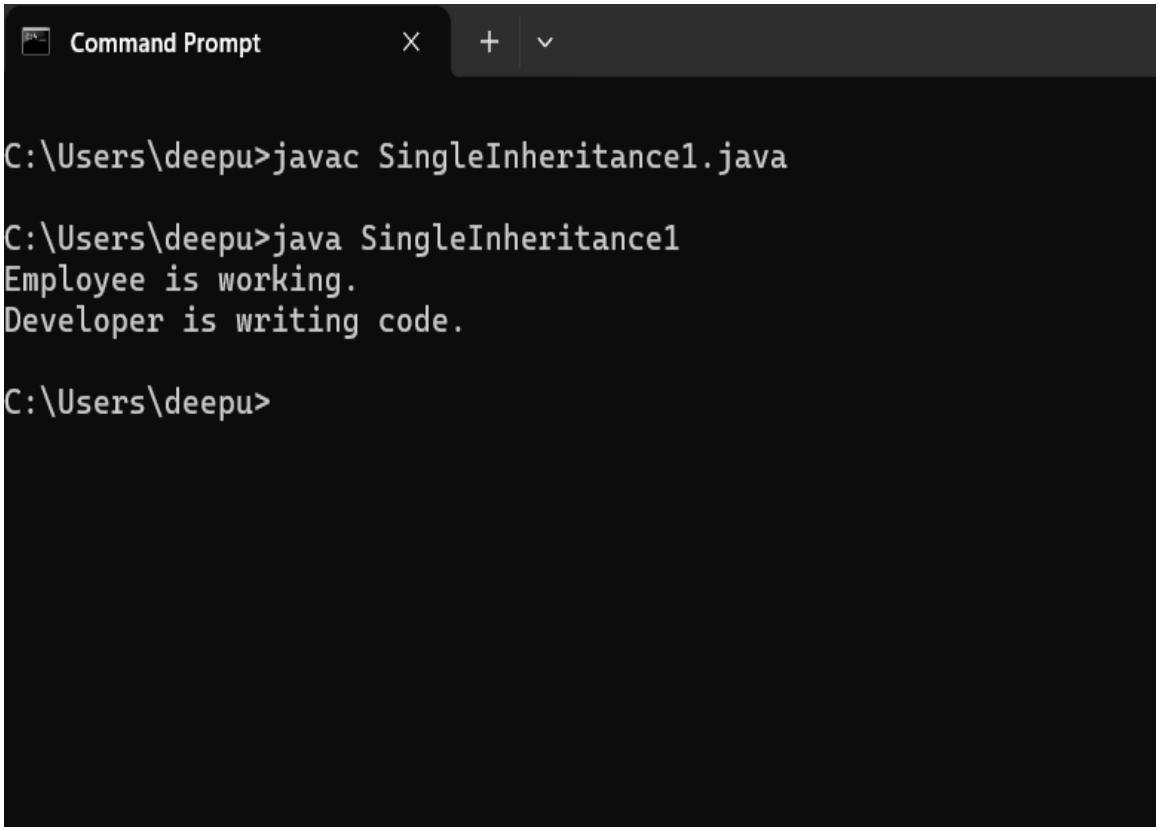
Single INHERITANCE

CODE:

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

class Developer extends Employee {
   void code() {
      System.out.println("Developer is writing code.");
   }
}
```

```
public class SingleInheritance1 {
   public static void main(String[] args) {
      Developer dev = new Developer();
      dev.work();
      dev.code();
   }
}
```

OUTPUT:



4 b) machine-printer

CODE:

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

```
class Printer extends Machine {
   void printDocument() {
      System.out.println("Printer is printing a document.");
   }
}

public class SingleInheritance2 {
   public static void main(String[] args) {
      Printer p = new Printer();
      p.start();
      p.printDocument();
   }
}
```

OUTPUT:

```
C:\Users\deepu>javac SingleInheritance2.java

C:\Users\deepu>java SingleInheritance2
Machine is starting...
Printer is printing a document.
```

AIM: OVERLOAD

CODE:

```
class A {


   public static int Multiply(int a, int b)
   {
```

```
      return a * b;
   }
   public static double Multiply(double a, double b)
   {

      return a * b;
   }
}
public class Overload
{

   public static void main(String[] args) {
    A n=new A();
     System.out.println(n.Multiply(2, 4));
      System.out.println(n.Multiply(5.5, 5.5));
   }
}
```

OUTPUT:

```
C:\Users\deepu>javac SingleInheritance2.java

C:\Users\deepu>java SingleInheritance2
Machine is starting...
Printer is printing a document.

C:\Users\deepu>
```

5.multilevel inheritance Programs

5 a) student-graduate-researcher

CODE:

```
class Student {
   void study() {
```

```
        System.out.println("Student is studying.");
    }
}

class Graduate extends Student {
    void specialize() {
        System.out.println("Graduate is specializing in a subject.");
    }
}

class Researcher extends Graduate {
    void research() {
        System.out.println("Researcher is conducting
         experiments.");
    }
}

public class MultilevelInheritance1 {
    public static void main(String[] args) {
        Researcher r = new Researcher();
        r.study();
        r.specialize();
        r.research();
    }
}
```

OUTPUT:

```
C:\Users\deepu>javac MultilevelInheritance1.java

C:\Users\deepu>java MultilevelInheritance1
Student is studying.
Graduate is specializing in a subject.
Researcher is conducting experiments.

C:\Users\deepu>
```

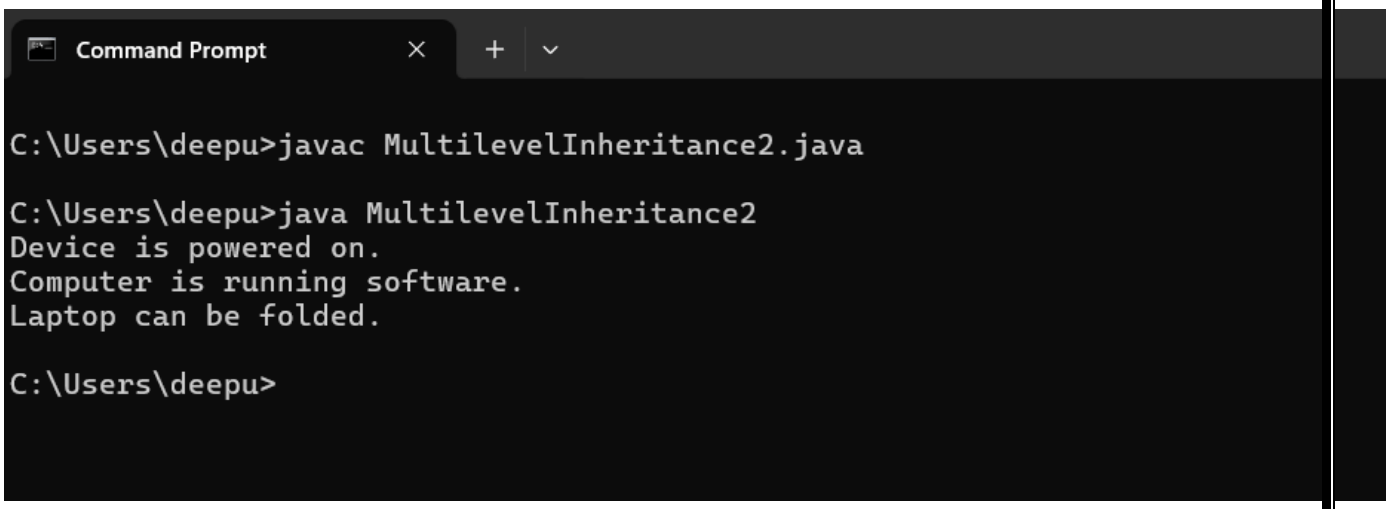
computer-laptop

CODE:

```
class Device {
   void powerOn() {
      System.out.println("Device is powered on.");
   }
}

class Computer extends Device {
   void runSoftware() {
      System.out.println("Computer is running software.");
   }
}

class Laptop extends Computer {
   void fold() {
      System.out.println("Laptop can be folded.");
   }
}

public class MultilevelInheritance2 {
   public static void main(String[] args) {
      Laptop myLaptop = new Laptop();
```

```
        myLaptop.powerOn();
        myLaptop.runSoftware();
        myLaptop.fold();
    }
}
```

OUTPUT:



6 hierarchical inheritance Programs

6 a) Appliance :

CODE:

```
class Appliance {
    void consumeElectricity() {
        System.out.println("Appliance consumes electricity.");
    }
}

class WashingMachine extends Appliance {
    void washClothes() {
        System.out.println("Washing Machine is washing clothes.");
```

```
    }
}

class Refrigerator extends Appliance {
    void keepFoodFresh() {
        System.out.println("Refrigerator keeps food fresh.");
    }
}

public class HierarchicalInheritance1 {
    public static void main(String[] args) {
        WashingMachine wm = new WashingMachine();
        wm.consumeElectricity(); // Inherited
        wm.washClothes();       // Own method

        Refrigerator fridge = new Refrigerator();
        fridge.consumeElectricity(); // Inherited
        fridge.keepFoodFresh();     // Own method
    }
}
```

OUTPUT:

```
C:\Users\deepu>javac HierarchicalInheritance1.java

C:\Users\deepu>java HierarchicalInheritance1
Appliance consumes electricity.
Washing Machine is washing clothes.
Appliance consumes electricity.
Refrigerator keeps food fresh.

C:\Users\deepu>
```

6 b) Game:

CODE:

```
class Game {
   void startGame() {
      System.out.println("Game has started.");
   }
}

class Chess extends Game {
   void movePiece() {
      System.out.println("Moving a chess piece.");
   }
}

class Football extends Game {
   void kickBall() {
      System.out.println("Kicking the football.");
   }
}

public class HierarchicalInheritance2 {
   public static void main(String[] args) {
      Chess c = new Chess();
      c.startGame(); // Inherited
      c.movePiece(); // Own method

      Football f = new Football();
      f.startGame(); // Inherited
      f.kickBall();  // Own method
   }
}
```

OUTPUT:

```
C:\Users\deepu>javac HierarchicalInheritance2.java

C:\Users\deepu>java HierarchicalInheritance2.java
Game has started.
Moving a chess piece.
Game has started.
Kicking the football.

C:\Users\deepu>
```

7.hybrid inheritance Programs

7 a) person

CODE:

```
interface Worker {
   void performDuties();
}

class Person {
   void eat() {
      System.out.println("Person is eating.");
   }
}

class Doctor extends Person implements Worker {
   public void performDuties() {
      System.out.println("Doctor is treating patients.");
   }
}

class Engineer extends Person implements Worker {
   public void performDuties() {
```

```
        System.out.println("Engineer is designing a project.");
    }
}

public class HybridInheritance1 {
    public static void main(String[] args) {
        Doctor d = new Doctor();
        d.eat();         // From Person
        d.performDuties(); // From Worker

        Engineer e = new Engineer();
        e.eat();         // From Person
        e.performDuties(); // From Worker
    }
}
```

OUTPUT:

```
C:\Users\deepu>javac HybridInheritance1.java

C:\Users\deepu>java HybridInheritance1
Person is eating.
Doctor is treating patients.
Person is eating.
Engineer is designing a project.
```

7 b) smart device

CODE:

```
interface Connectivity {
    void connectToInternet();
}

class SmartDevice {
    void powerOn() {
        System.out.println("Smart Device is powered on.");
    }
```

```
}

class Smartphone extends SmartDevice implements Connectivity
    {
  public void connectToInternet() {
    System.out.println("Smartphone is connected to the
     internet.");
  }
}

class SmartWatch extends SmartDevice implements Connectivity
    {
  public void connectToInternet() {
    System.out.println("Smartwatch is connected to the
     internet.");
  }
}

public class HybridInheritance2 {
  public static void main(String[] args) {
    Smartphone phone = new Smartphone();
    phone.powerOn();
    phone.connectToInternet();

    SmartWatch watch = new SmartWatch();
    watch.powerOn();
    watch.connectToInternet();
  }
}
```

OUTPUT:

```
C:\Users\deepu>javac HybridInheritance2.java

C:\Users\deepu>java HybridInheritance2
Smart Device is powered on.
Smartphone is connected to the internet.
Smart Device is powered on.
Smartwatch is connected to the internet.

C:\Users\deepu>
```

POLYMORPHISM

8 a) student constructor

CODE:

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

   Student(String n, int a) {
      name = n;
      age = a;
   }

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

   public static void main(String[] args) {
      Student s1 = new Student("GOKUL", 20);
      s1.display();
```

```
   }
}
```

OUTPUT:

```
C:\Users\deepu>javac Student.java

C:\Users\deepu>java Student
Name: Navadeep, Age: 18

C:\Users\deepu>
```

CONSTRUCTOR OVERLOAD

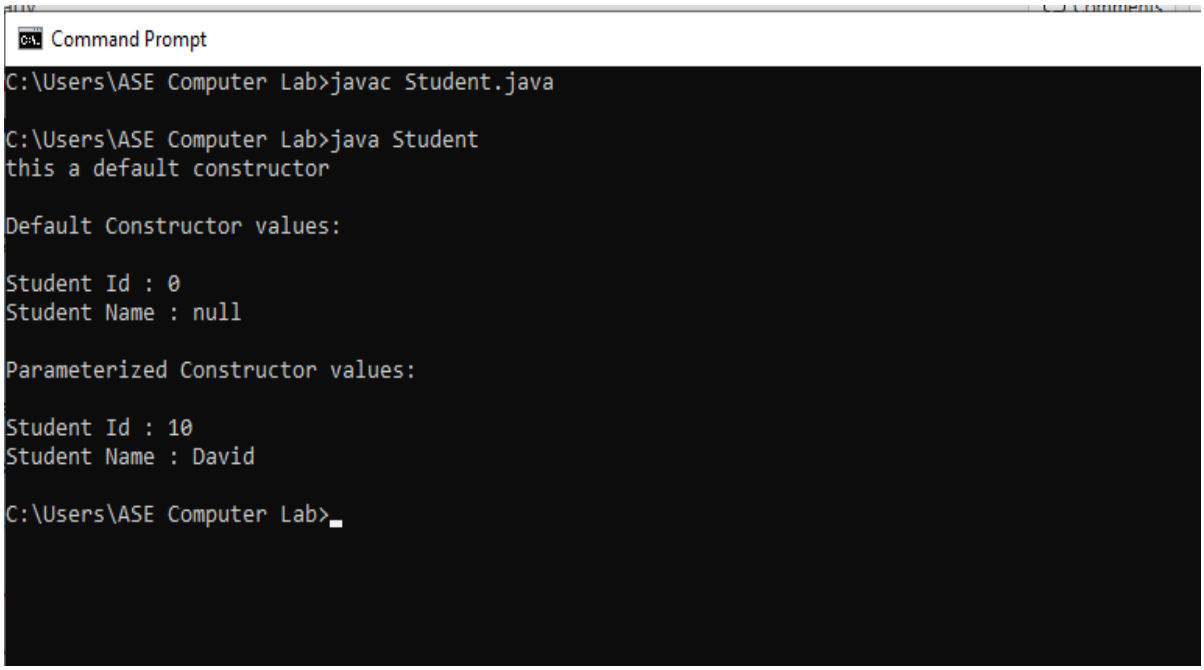
9a) studentid and age

Code:

```
public class Student{
int id;
String name;

Student(){
System.out.println("this a default constructor");
}

Student(int i, String n){
id = i;
name = n;
}

public static void main(String[] args) {
//object creation
```

```
Student s = new Student();
System.out.println("\nDefault Constructor values: \n");
System.out.println("Student Id : "+s.id + "\nStudent Name :
        "+s.name);

System.out.println("\nParameterized Constructor values: \n");
Student student = new Student(10, "David");
System.out.println("Student Id : "+student.id + "\nStudent Name
        : "+student.name);
}
}
```

OUTPUT:



10.Method overloading Programs

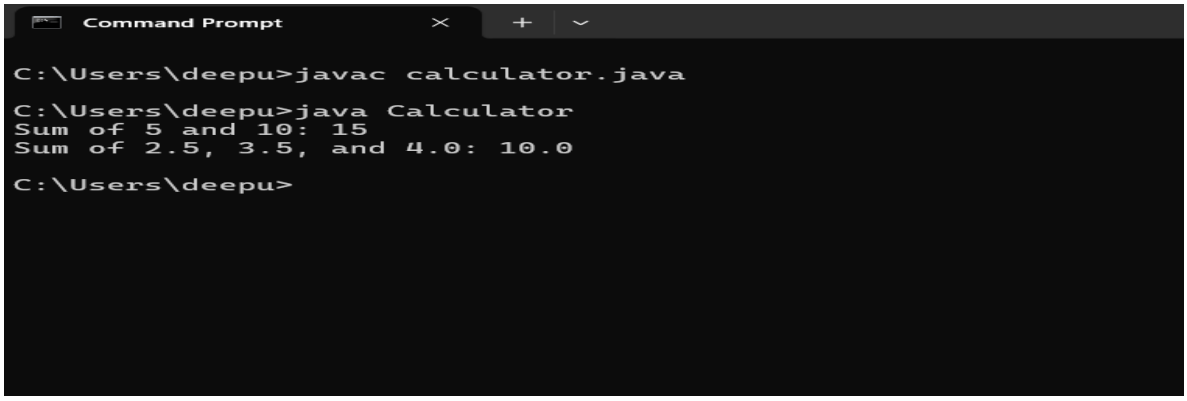
10 a) Calculator addition overloading

CODE:

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

```
    }

    public double add(double a, double b, double c) {
        return a + b + c;
    }

    public static void main(String[] args) {
        Calculator calc = new Calculator();
        System.out.println("Sum of 5 and 10: " + calc.add(5, 10));
        System.out.println("Sum of 2.5, 3.5, and 4.0: " + calc.add(2.5,
         3.5, 4.0));
    }
}
```

OUTPUT:



10 b)

CODE:

```
class Display {
    public void show(int number) {
        System.out.println("Integer: " + number);
    }

    public void show(String message) {
        System.out.println("Message: " + message);
    }
```

```
    public static void main(String[] args) {
        Display obj = new Display();
        obj.show(42);
        obj.show("Polymorphism");
    }
}
```

OUTPUT:

```
C:\Users\deepu>javac Display.java

C:\Users\deepu>java Display
Integer: 42
Message: Polymorphism

C:\Users\deepu>
```

11.METHOD OVERRIDING

11 a)E-COMMERCE:

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:

```
PS C:\Users\user\OneDrive\Documents\Java Programs> java ECommerce.java
=== Products ===
Laptop - $999.99
Type: Electronic

Milk - $3.99
Type: Grocery
```

11 b) BANK

CODE:

```
class Bank {
    public double getInterestRate() {
        return 5.0;
    }
}

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

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

```
}

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

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

OUTPUT:

```
PS C:\Users\user\OneDrive\Documents\Java Programs> java Main.java
Bank Interest Rate: 5.0%
SBI Interest Rate: 6.5%
HDFC Interest Rate: 7.0%
PS C:\Users\user\OneDrive\Documents\Java Programs>
```

12.a) interface program:

Code:

```
 interface Inf1{
  public void method1();
}
interface Inf2 extends Inf1 {
  public void method2();
}
public class interface1 implements Inf2{
   public void method1(){
System.out.println("method1");
```

```
   }
   public void method2(){
System.out.println("method2");
   }
   public static void main(String args[]){
Inf2 obj = new interface1();
obj.method2();
   }
}
```

Output:

```
C:\Users\amma>javac interface1.java

C:\Users\amma>java interface1
method2

C:\Users\amma>
```

12.b) print a message in cmd

Code

```
interface Printable{
  void print();
}

class Printer implements Printable{
  public void print(){System.out.println("Hello");}
}
public class interface2{
  public static void main(String args[]){
    Printable p=new Printer();
    p.print();
}
}
```

OUTPUT:

```
:\Users\amma>javac interface2.java

:\Users\amma>java interface2
ello

:\Users\amma>
```

12c) OnlineExamSystem

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:

```
Starting Multiple Choice Questions exam...
Starting Coding Exam...
```

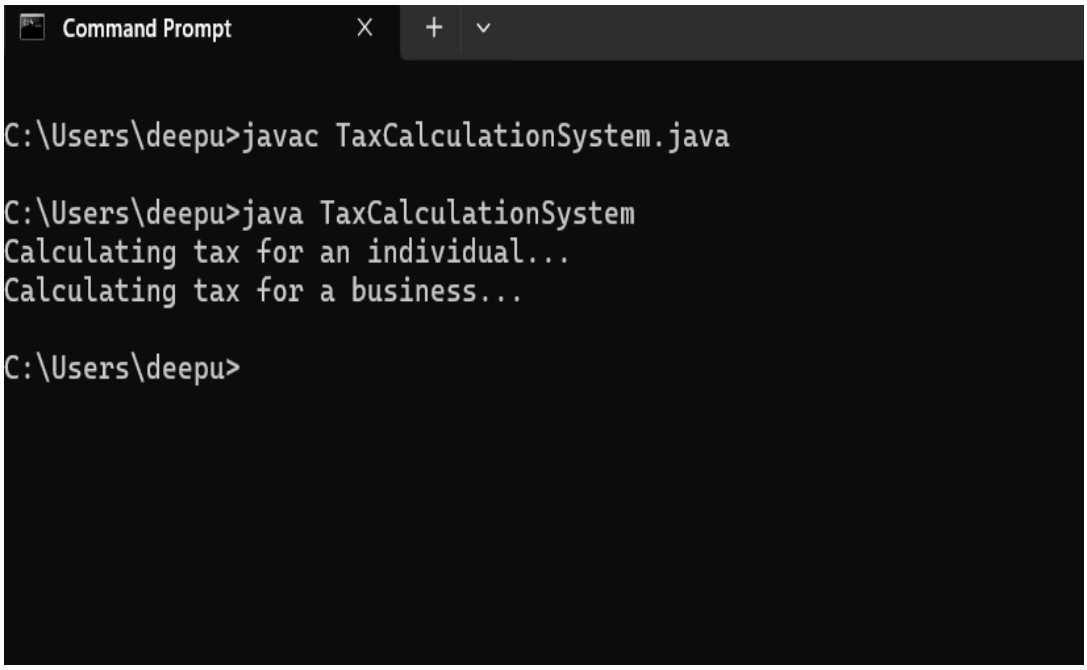
12d) TaxCalculationSystem

CODE:

```
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:



13a) draw circle

Code:

```
abstract class Shape{
abstract void draw();
}

class Rectangle extends Shape{
void draw(){System.out.println("drawing rectangle");}
}
class Circle extends Shape{
void draw(){System.out.println("drawing circle");}
}
public class abstraction1{
public static void main(String args[]){
  Shape s=new Circle();
  s.draw();
```

```
}
}
```

OUTPUT:

```
C:\Users\amma>javac abstraction1.java

C:\Users\amma>java abstraction1
drawing circle
```

13.b) DETAILS OF DIAGRAM

CODE:

```
abstract class Shape { protected String color;
public Shape(String color) {
        this.color = color;
    }

    public void setColor(String color) {
        this.color = color;
    }

    abstract double calculateArea();
    abstract double calculatePerimeter();

    public void displayInfo() {
        System.out.println("Color: " + color);
        System.out.println("Area: " + calculateArea());
        System.out.println("Perimeter: " + calculatePerimeter());
    }

}
class Circle extends Shape { private double radius;
public Circle(String color, double radius) {
        super(color);
        this.radius = radius;
    }

    @Override
    public double calculateArea() {
        return Math.PI * radius * radius;
    }
```

```
    @Override
    public double calculatePerimeter() {
      return 2 * Math.PI * radius;
    }

}
class Rectangle extends Shape { private double length; private
    double width;
public Rectangle(String color, double length, double width) {
      super(color);
      this.length = length;
      this.width = width;
    }

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

    @Override
    public double calculatePerimeter() {
      return 2 * (length + width);
    }

}
public class abstract2{ public static void main(String[] args) {
    Circle circle = new Circle("Red", 5.0); Rectangle rectangle =
    new Rectangle("Blue", 4.0, 6.0);
  circle.displayInfo();
      System.out.println();

      rectangle.displayInfo();
      System.out.println();

      rectangle.setColor("Green");
      rectangle.displayInfo();
    }

}
```

OUTPUT:

```
Circle Area: 78.53981633974483
```

```
Employee {
abstract double calculateSalary();
}
class FullTimeEmployee extends Employee { private double
    monthlySalary;
public FullTimeEmployee(double salary) { this.monthlySalary =
    salary;
}

public double calculateSalary() { return monthlySalary;
}
}

class PartTimeEmployee extends Employee { private int
    hoursWorked;
private double hourlyRate;

public PartTimeEmployee(int hoursWorked, double hourlyRate)
    { this.hoursWorked = hoursWorked;
this.hourlyRate = hourlyRate;
}
public double calculateSalary() { return hoursWorked *
    hourlyRate;
}
}

public class EmployeeTest {
public static void main(String[] args) {
Employee fullTime = new FullTimeEmployee(5000); Employee
    partTime = new PartTimeEmployee(20, 15);

System.out.println("Full-time Salary: $" +
    fullTime.calculateSalary()); System.out.println("Part-time
```

```
        Salary: $" + partTime.calculateSalary());
}
}
```

```
C:\Users\amma>javac abstract2.java

C:\Users\amma>java abstract2
Color: Red
Area: 78.53981633974483
Perimeter: 31.41592653589793

Color: Blue
Area: 24.0
Perimeter: 20.0

Color: Green
Area: 24.0
Perimeter: 20.0
```

13.d)Vehicle:

CODE:

```
abstract class Vehicle {
abstract void start();
}


class Car extends Vehicle { public void start() {
System.out.println("Car is starting with a key...");
}
}

class Bike extends Vehicle { public void start() {
System.out.println("Bike is starting with a self-start button...");
}
}
```

```
public class VehicleTest {
public static void main(String[] args) { Vehicle car = new Car();
Vehicle bike = new Bike();

car.start();
bike.start();
}
}
```

OUTPUT:

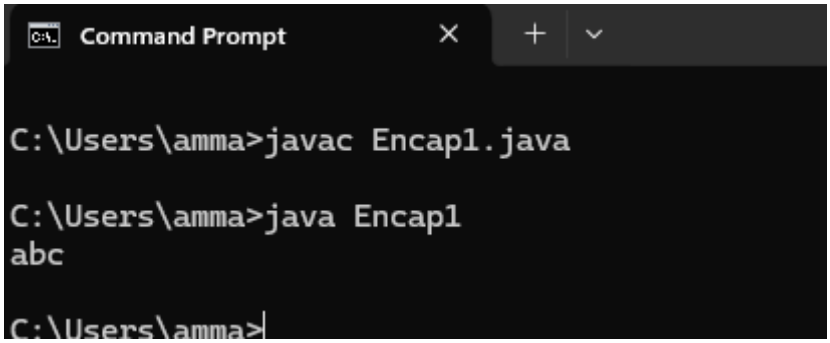
```
Car is starting with a key...
Bike is starting with a self-start button...
```

ENCAPTULATION

CODE: PRINT MESSAGE

```
class Hello{
   private String name;


   public String getName() { return name; }
   public void setName(String name) { this.name = name; }
}

public class Encap1{
   public static void main(String[] args) {
      Hello p = new Hello();
      p.setName("abc");
      System.out.println(p.getName());
   }
}
```

OUTPUT:



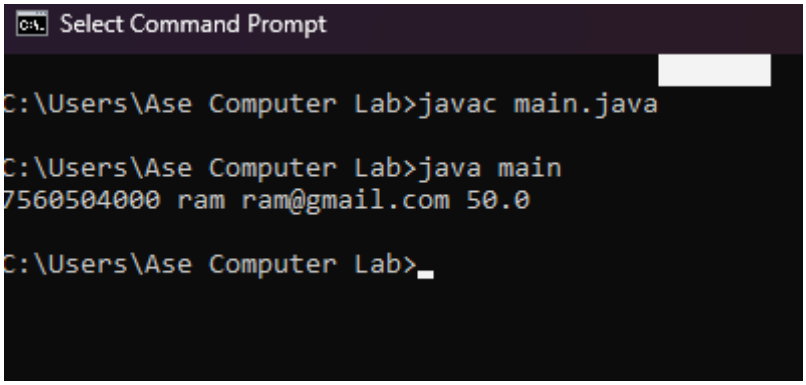
14 b) fully encaptulation

Code:

```
class Account {
//private data members
private long acc_no;
private String name,email;
private float amount;
//public getter and setter methods
public long getAcc_no() {
 return acc_no;
}
public void setAcc_no(long acc_no) {
 this.acc_no = acc_no;
}
public String getName() {
 return name;
}
public void setName(String name) {
 this.name = name;
}
public String getEmail() {
 return email;
}
public void setEmail(String email) {
 this.email = email;
```

```
}
public float getAmount() {
 return amount;
}
public void setAmount(float amount) {
 this.amount = amount;
}
}
//A Java class to test the encapsulated class Account.
public class main {
public static void main(String[] args) {
 //creating instance of Account class
 Account acc=new Account();
 //setting values through setter methods
 acc.setAcc_no(7560504000L);
 acc.setName("ram");
 acc.setEmail("ram@gmail.com");
 acc.setAmount(50f);
 //getting values through getter methods
 System.out.println(acc.getAcc_no()+" "+acc.getName()+"
      "+acc.getEmail()+" "+acc.getAmount());
}
}
```

OUTPUT:



14. C) FULLY ENCAPTULATION ON OFFICE MANAGEMENT

CODE:

```
 class Employee {
private String emp_name;
 private String emp_id;
 private double net_salary;
 public Employee(String emp_name, String emp_id, double
      net_salary) {
 this.emp_name = emp_name;
 this.emp_id = emp_id;
 this.net_salary = net_salary; }
 public String getEmpName() { return emp_name; }
public String getEmpId() { return emp_id; }
public double getSalary() { return net_salary; }
public void setEmpName(String emp_name) { this.emp_name =
      emp_name; }
public void setEmpId(String emp_id) { this.emp_id = emp_id; }
public void setSalary(double net_salary) { this.net_salary =
      net_salary; } }
 public class main { public static void main(String args[]) {
 Employee emp = new Employee("Robert", "EMP001",
      75450.00);
// Printing data
      System.out.println("Employee (Intial Values):");
      System.out.println(emp.getEmpId() + " , " +
      emp.getEmpName() + " , " + emp.getSalary());

      // Updating values using setter methods
      emp.setEmpName("Riyan");
      emp.setEmpId("EMP002");
      emp.setSalary(90500.00);

      // Printing data
      System.out.println("Employee (Updated Values):");
      System.out.println(emp.getEmpId() + " , " +
      emp.getEmpName() + " , " + emp.getSalary());

} }
```

OUTPUT:

```
C:\Users\Ase Computer Lab>javac main.java

C:\Users\Ase Computer Lab>java main
Employee (Intial Values):
EMP001 , Robert , 75450.0
Employee (Updated Values):
EMP002 , Riyan , 90500.0

C:\Users\Ase Computer Lab>_
```

PACKAGES:

15.a) CODE

```
public class Addition {
   public int add(int a, int b) {
      return a + b;
   }
}
import mathoperations.Addition;

public class UserPackageExample1 {
   public static void main(String[] args) {
      Addition obj = new Addition();
      System.out.println("Sum: " + obj.add(5, 10));
   }
}
```

OUTPUT:

```
Sum: 15
```

15 b) user defined package

Package file:

CODE: area calculator

```
package shapes;

public class Circle {
   private double radius;

   public Circle(double radius) {
      this.radius = radius;
   }

   public double area() {
      return Math.PI * radius * radius;
   }
}

import shapes.Circle;

public class UserPackageExample2 {
   public static void main(String[] args) {
      Circle c = new Circle(5);
      System.out.println("Circle Area: " + c.area());
   }
}
```

OUTPUT:

```
Circle Area: 78.53981633974483
```

15.c)CODE:

```
import java.io.Console;

class pack1 {
  public static void main(String args []) {
```

```
    Console cs = System.console();
    System.out.println("Enter your name : ");
    String name = cs.readLine();
    System.out.println("Welcome : "+name);
  }
}
```

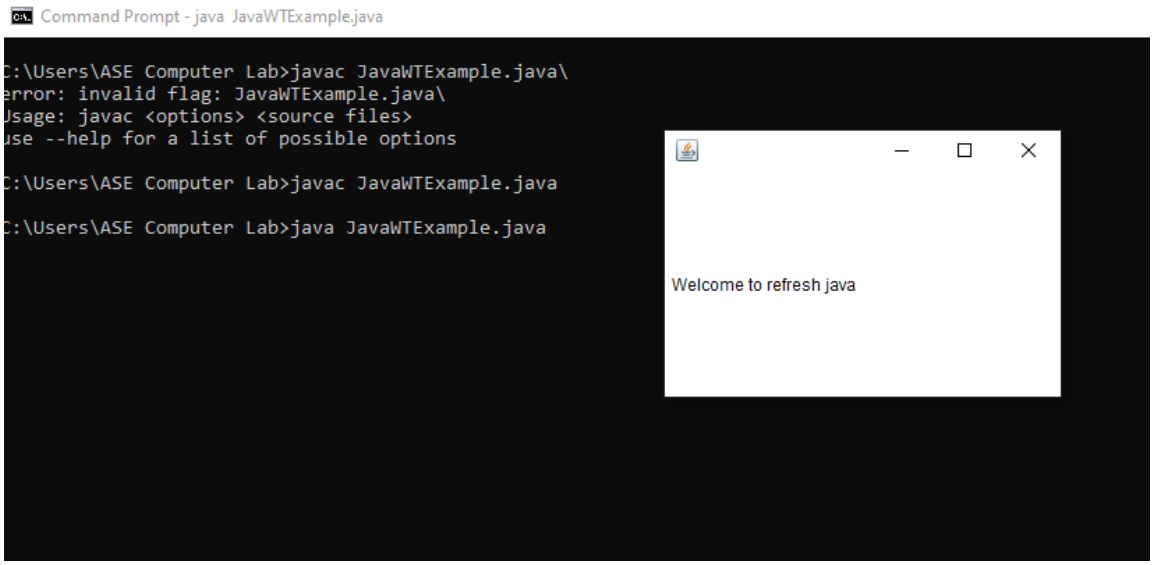
OUTPUT:

```
C:\Users\ASE Computer Lab>javac pack1.java

C:\Users\ASE Computer Lab>java pack1
Enter your name :
ram
Welcome : ram

C:\Users\ASE Computer Lab>_
```

Code:

```
import java.awt.*;

class JavaAWTExample {
  // Declaring constructor
  public JavaAWTExample() {
    Frame fm = new Frame(); //Creating a frame
    Label lb = new Label(" Welcome to refresh java"); //Creating a
      label
    fm.add(lb); //adding label to the frame
    fm.setSize(300, 200); //setting frame size
    fm.setVisible(true); //setting frame visibility as true
  }
  public static void main(String args []) {
    JavaAWTExample awt = new JavaAWTExample();
  }
}
```

OUTPUT:



16. Exception Handling Programs

16.a) Array Exceptiion

CODE:

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

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

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

OUTPUT:

```
Error: Index out of bounds!
Program continues...
```

16.b) Divide By Zero Exception :

CODE:

```
public class DivideByZero {
public static void main(String[] args) { try {
int result = 10 / 0; // This will cause ArithmeticException
    System.out.println("Result: " + result);
} catch (ArithmeticException e) { System.out.println("Error:
    Cannot divide by zero!");
}
System.out.println("Program continues...");
}
}
```

OUTPUT:

```
Error: Cannot divide by zero!
Program continues...
```

16.c) Custom Exception

CODE:

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

public class CustomExceptionExample {
static void checkAge(int age) throws AgeException { if (age < 18)
    {
throw new AgeException("Not eligible to vote!");
} else {
System.out.println("Eligible to vote.");
}
}

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

```
checkAge(16); // This will throw an exception
} catch (AgeException e) { System.out.println("Exception: " +
        e.getMessage());
}
}
}
```

OUTPUT:

```
Exception: Not eligible to vote!
```

16d) FINALLY BLOCK

CODE:

```
public class FinallyExample {
public static void main(String[] args) { try {
int num = 10 / 2; System.out.println("Result: " + num);
} catch (ArithmeticException e) { System.out.println("Error:
        Division by zero!");
} finally {
System.out.println("This will always execute.");
}

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

OUTPUT:

```
C:\Users\deepu>javac FinallyExample.java

C:\Users\deepu>java FinallyExample
Result: 5
This will always execute.
Program continues...

C:\Users\deepu>
```

FILE HANDLING PROGRAMS

17a)AIM:Array list.

CODE:

```
import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;
import java.util.ArrayList;
public class gh { public static void main(String[] args) {
    ArrayList names = new ArrayList<>();
    names.add("Alice");
names.add("Bob");
names.add("Charlie");
names.add("David");
String filePath = "output.txt";
try (BufferedWriter writer = new BufferedWriter(new
    FileWriter(filePath)))
{ for (String name : names) { writer.write(name);
writer.newLine(); }
System.out.println("ArrayList successfully written to
    file!"); }
catch (IOException e)
{ System.out.println("An error occurred: " +
    e.getMessage());

}
}
}
```

OUTPUT:

```
Command Prompt
Microsoft Windows [Version 10.0.26100.3476]
(c) Microsoft Corporation. All rights reserved.

C:\Users\deepu>javac gh.java

C:\Users\deepu>java gh
ArrayList successfully written to file!

C:\Users\deepu>
```

17b)AIM: Task Manager.

CODE:

```
import java.io.*;
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;
public class TaskManager {
   private static final String FILE_NAME = "tasks.txt";
   private static List<String> tasks = new ArrayList<>();
   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);
      while (true) {
         System.out.println("\nTo-Do List Manager");
         System.out.println("1. Add Task");
         System.out.println("2. View Tasks");
         System.out.println("3. Exit and Save");
         System.out.print("Enter your choice: ");
```

```
        int choice = scanner.nextInt();
        scanner.nextLine();

        switch (choice) {
            case 1:
                System.out.print("Enter task description: ");
                String task = scanner.nextLine();
                tasks.add(task);
                System.out.println("Task added!");
                break;
            case 2:
                if (tasks.isEmpty()) {
                    System.out.println("No tasks available.");
                } else {
                    System.out.println("\nYour Tasks:");
                    for (int i = 0; i < tasks.size(); i++) {
                        System.out.println((i + 1) + ". " +
tasks.get(i));
                    }
                }
                break;
            case 3:
                saveTasks();
                System.out.println("Tasks saved. Exiting...");
                scanner.close();
                return;
            default:
```

```
            System.out.println("Invalid choice! Please try
    again.");
        }
    }
}
private static void loadTasks() {
    try (BufferedReader br = new BufferedReader(new
    FileReader(FILE_NAME))) {
        String line;
        while ((line = br.readLine()) != null) {
            tasks.add(line);
        }
    } catch (FileNotFoundException e) {
        System.out.println("No previous tasks found.");
    } catch (IOException e) {
        System.out.println("Error reading tasks: " +
    e.getMessage());
    }
}
private static void saveTasks() {
    try (BufferedWriter bw = new BufferedWriter(new
    FileWriter(FILE_NAME))) {
        for (String task : tasks) {
            bw.write(task);
            bw.newLine();
        }
    } catch (IOException e) {
        System.out.println("Error saving tasks: " +
```

```
        e.getMessage());
        }
    }
}
```

OUTPUT:

```
To-Do List Manager
1. Add Task
2. View Tasks
3. Exit and Save
Enter your choice: asf
Exception in thread "main" java.util.InputMismatchException
        at java.base/java.util.Scanner.throwFor(Scanner.java:964)
        at java.base/java.util.Scanner.next(Scanner.java:1619)
        at java.base/java.util.Scanner.nextInt(Scanner.java:2284)
        at java.base/java.util.Scanner.nextInt(Scanner.java:2238)
        at TaskManager.main(TaskManager.java:18)
```

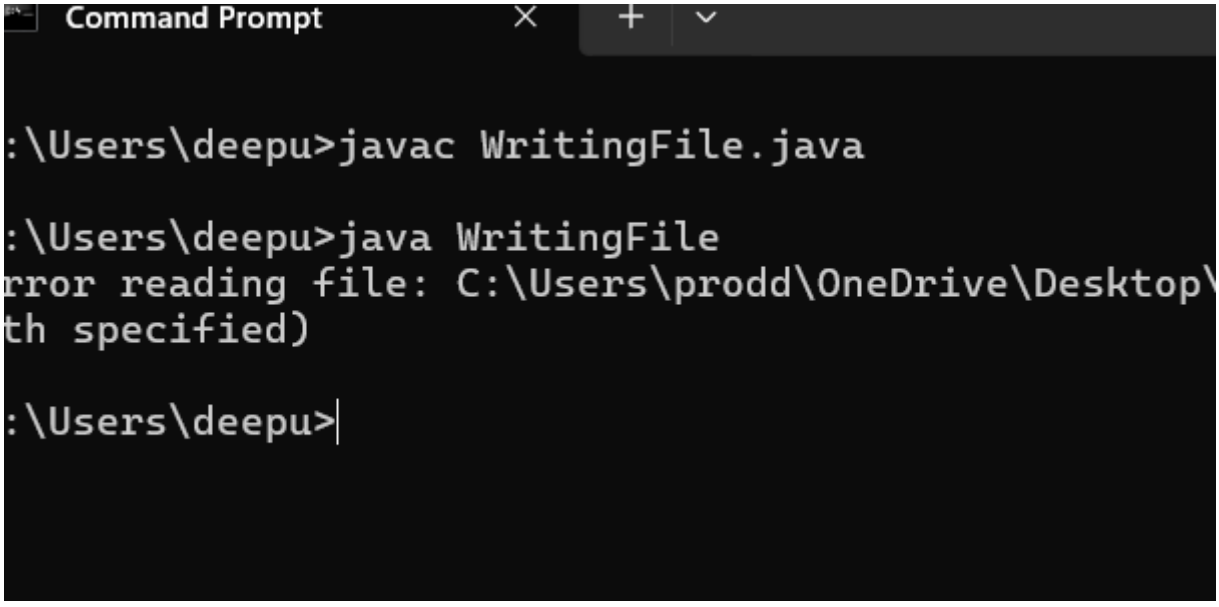
17c)AIM:Write file

CODE:

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
public class WritingFile {
    public static void main(String[] args) {
        try {
            FileReader fileReader = new
        FileReader("C:\\Users\\prodd\\OneDrive\\Desktop\
        \Sheshank\\OOPS\\File Handling\\config.txt");
            BufferedReader bufferedReader = new
        BufferedReader(fileReader);
            String line;
            System.out.println("Reading configuration:");
```

```
        while ((line = bufferedReader.readLine()) != null) {
            System.out.println(line);
        }
        bufferedReader.close();
    } catch (IOException e) {
        System.out.println("Error reading file: " +
    e.getMessage());
    }
  }
}
```

OUTPUT:



17d)AIM:File writer

CODE:

```
import java.io.*;
public class FileHandlingExample {
  public static void main(String[] args) {
    String fileName = "example.txt";
    try (FileWriter writer = new FileWriter(fileName)) {
      writer.write("Hello, this is a test file!\n");
```

```
    writer.write("This is line 2.");
    System.out.println("Successfully wrote to the file.");
} catch (IOException e) {
    System.out.println("Error writing to file: " +
e.getMessage());
}
try (BufferedReader reader = new
BufferedReader(new FileReader(fileName))) {
    System.out.println("\nFile contents:");
    String line;
    while ((line = reader.readLine()) != null) {
        System.out.println(line);
    }
} catch (IOException e) {
    System.out.println("Error reading file: " +
e.getMessage());
}
try (FileWriter writer = new FileWriter(fileName,
true)) {
    writer.write("\nThis is an appended line.");
    System.out.println("\nSuccessfully appended to the
file.");
} catch (IOException e) {
    System.out.println("Error appending to file: " +
e.getMessage());
}
try (FileInputStream fis = new
FileInputStream(fileName)) {
```

```
        System.out.println("\nReading using
    FileInputStream:");
        int content;
        while ((content = fis.read()) != -1) {
            System.out.print((char) content);
        }
    } catch (IOException e) {
        System.out.println("Error reading file: " +
    e.getMessage());
    }
  }
}
```

OUTPUT:

```
C:\Users\deepu>javac FileHandlingExample.java

C:\Users\deepu>java FileHandlingExample.java
error: can't find main(String[]) method in class: FileHandlingExample

C:\Users\deepu>
```