LAB RECORD

23CSE111- Object Oriented Programming

Submitted by

CH.SC.U4CSE24121- Kurra Venkata Gokul

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

AMRITA VISHWA VIDYAPEETHAM

AMRITA SCHOOL OF COMPUTING

CHENNAI

March - 2025

AMRITA VISHWA VIDYAPEETHAM

AMRITA SCHOOL OF COMPUTING, CHENNAI

BONAFIDE CERTIFICATE

This is to certify that the Lab Record work for 23CSE111- Object Oriented Programming Subject submitted by ***CH.SC.U4CSE24142 – Kurra Venkata Gokul*** 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 0 8 / 04 /2025

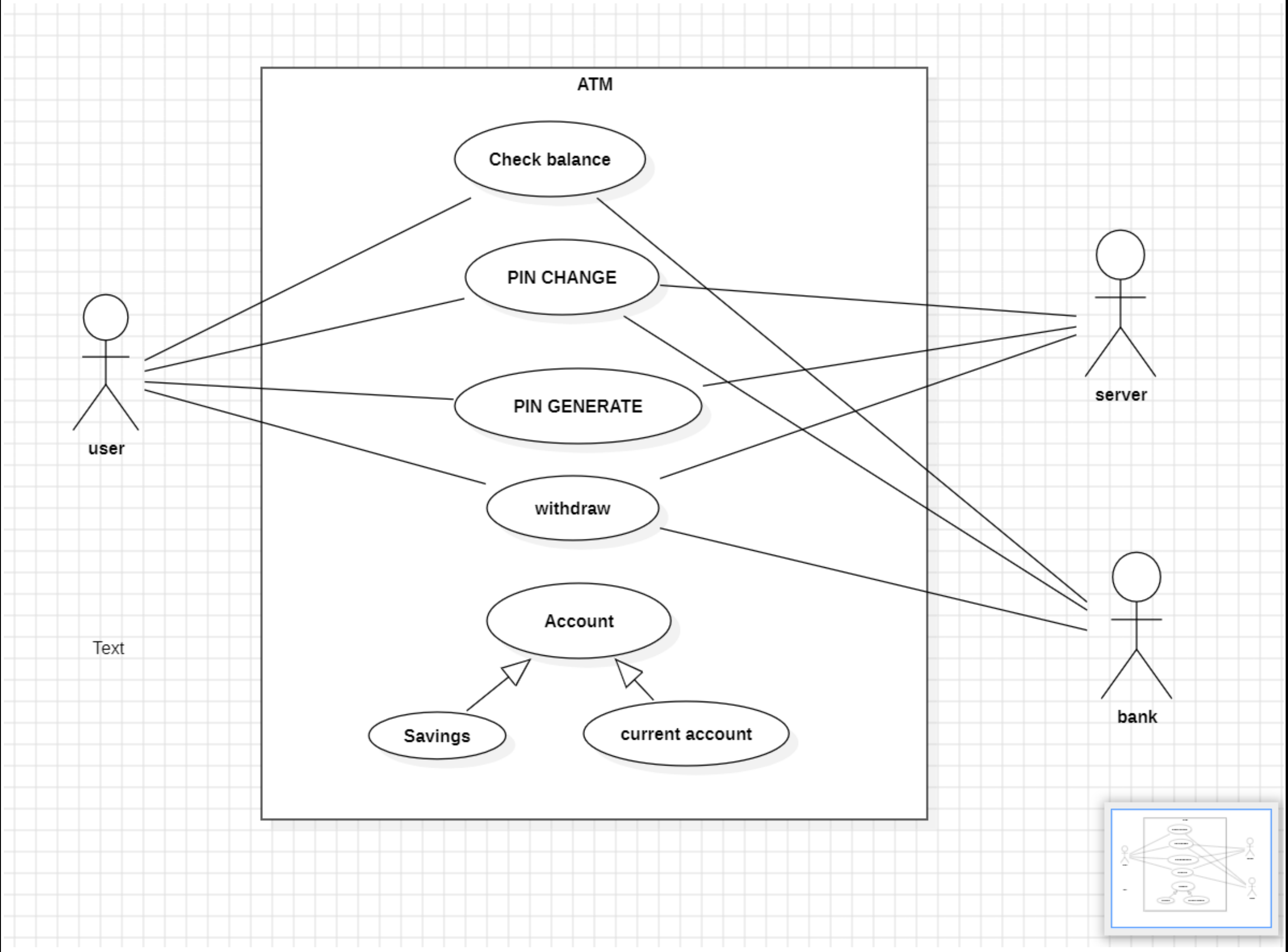
Internal Examiner 1 Internal Examiner 2

INDEX

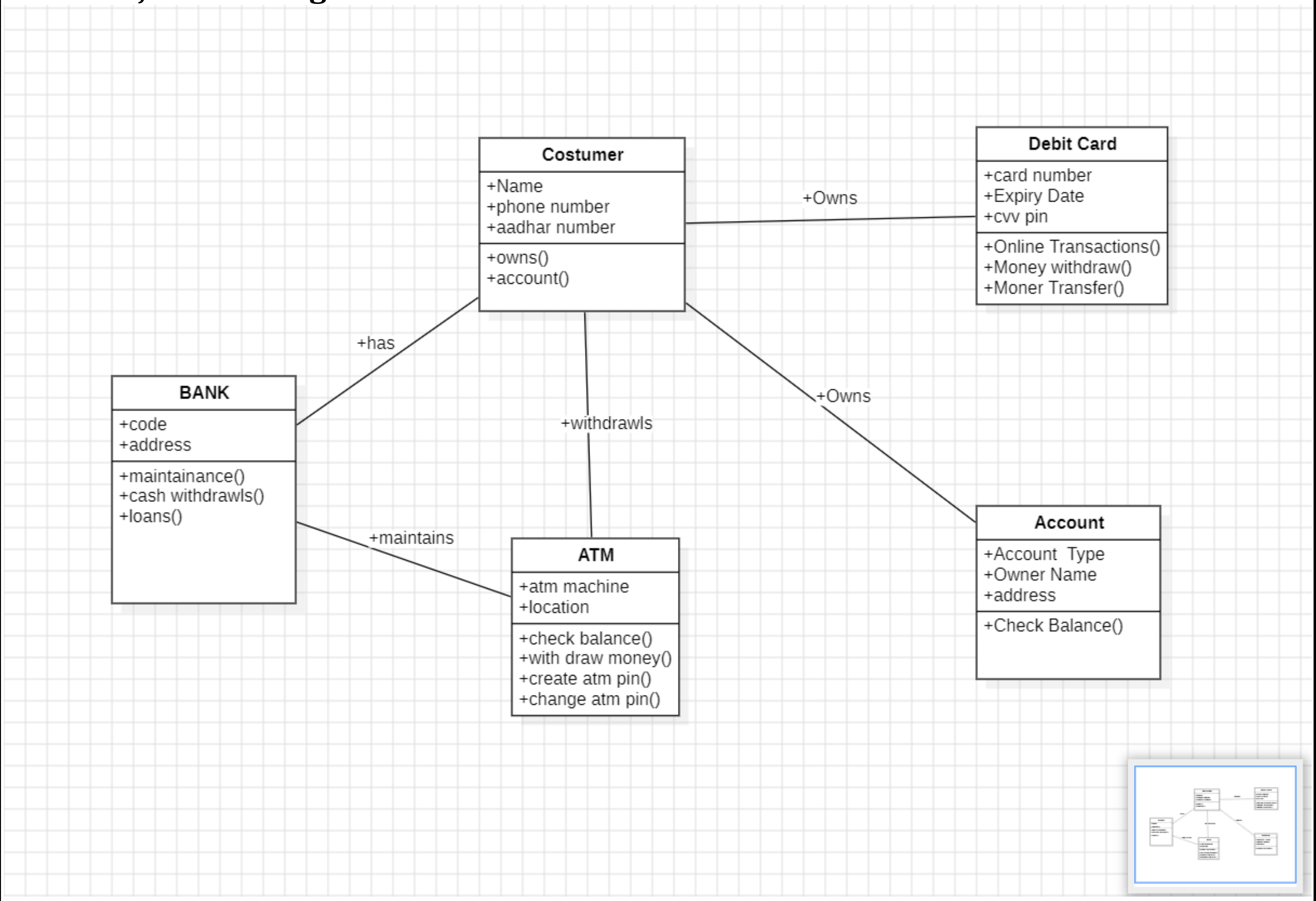
UML DIAGRAMS

1. ATM Withdraw

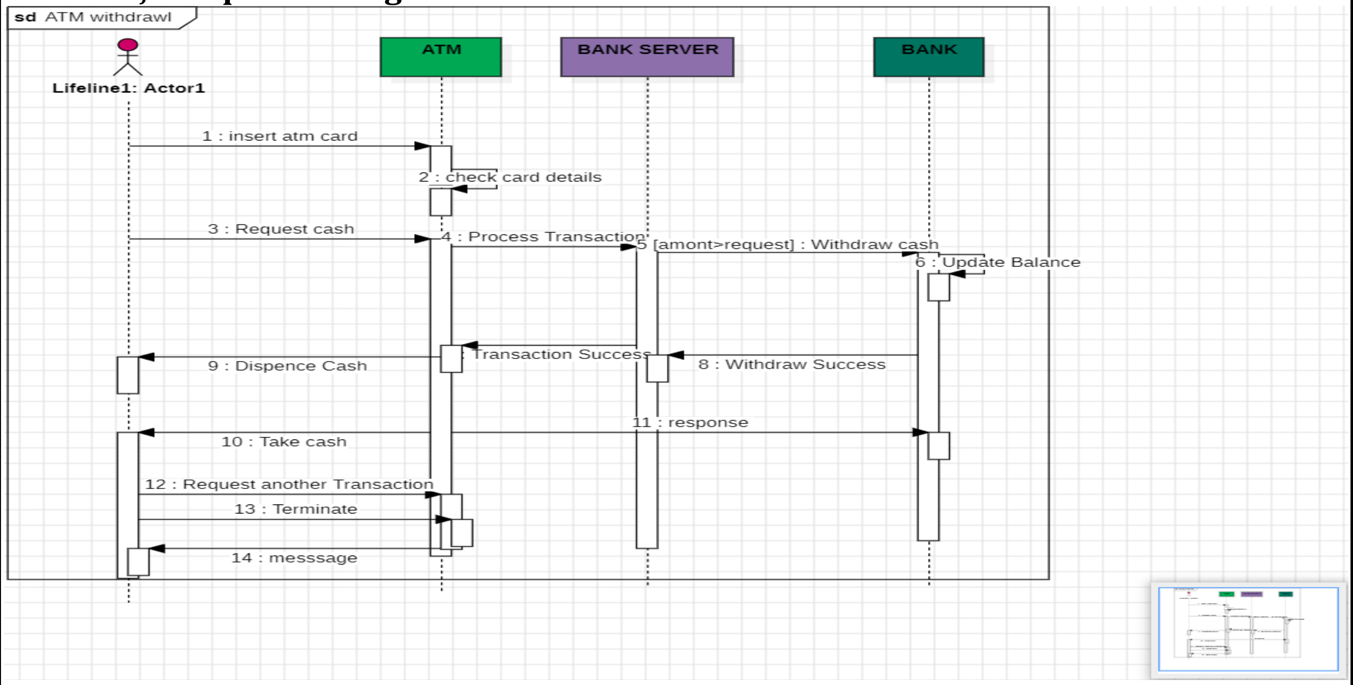
1.a) Use Case Diagram:



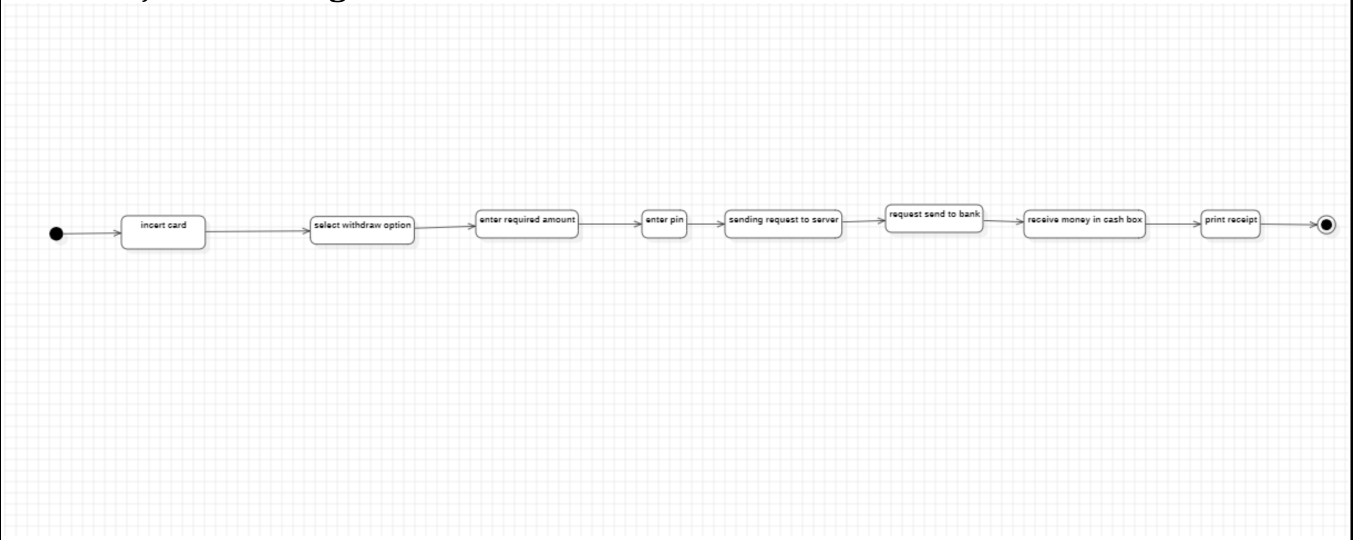
1.b) Class Diagram:



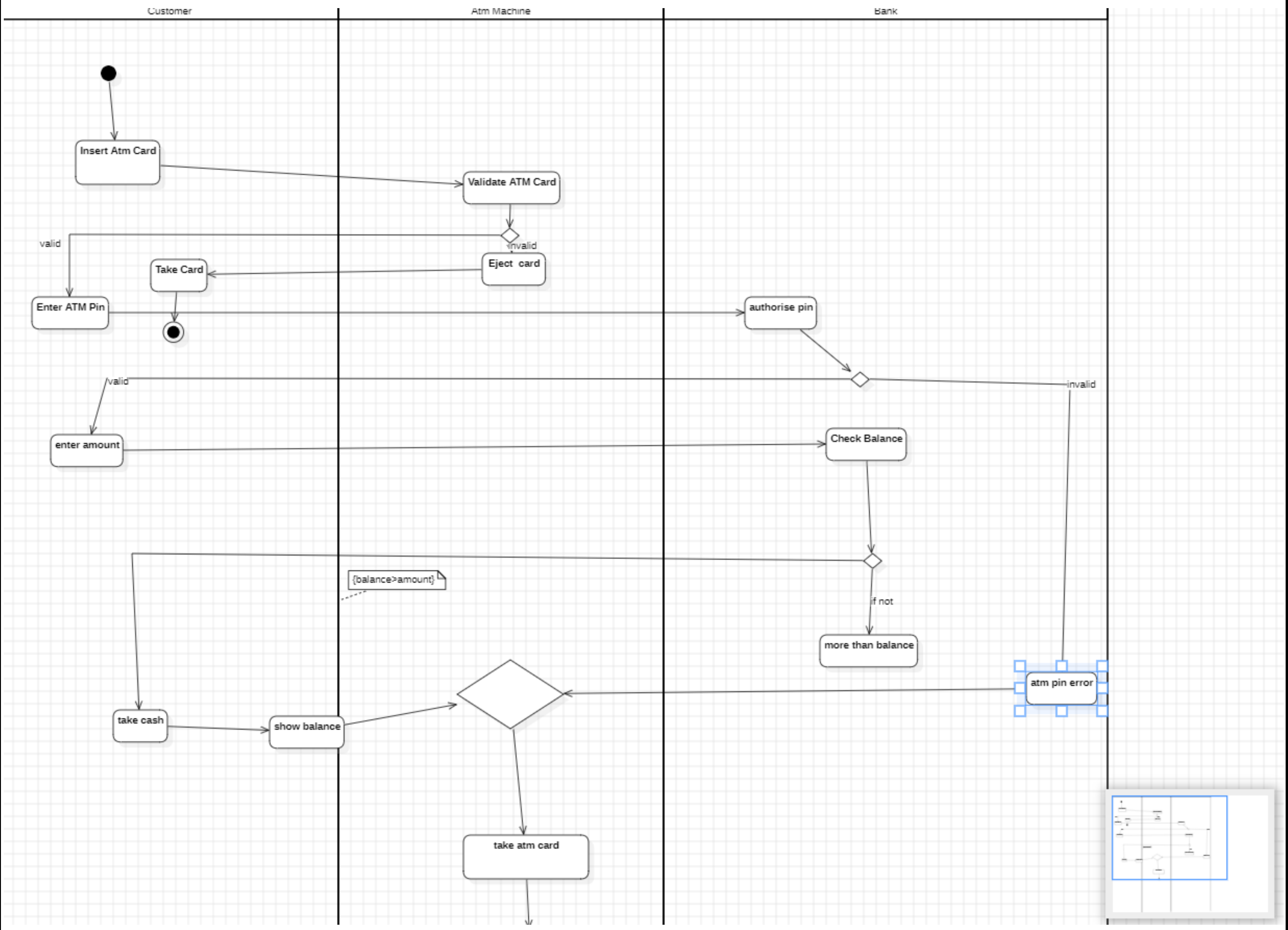
1.c) Sequence Diagram:



1.d) State- Diagram:

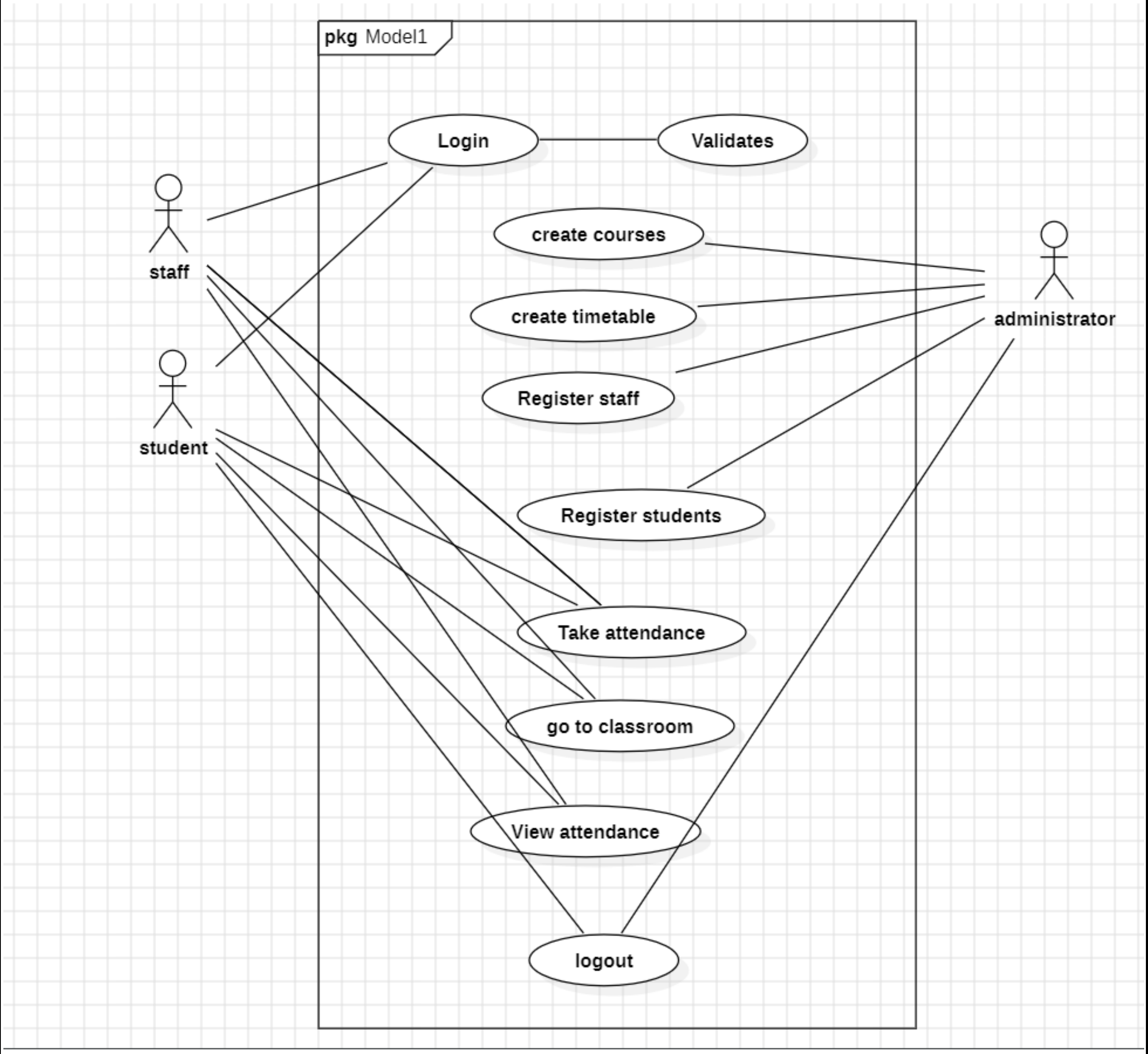


1.e) Activity Diagram:

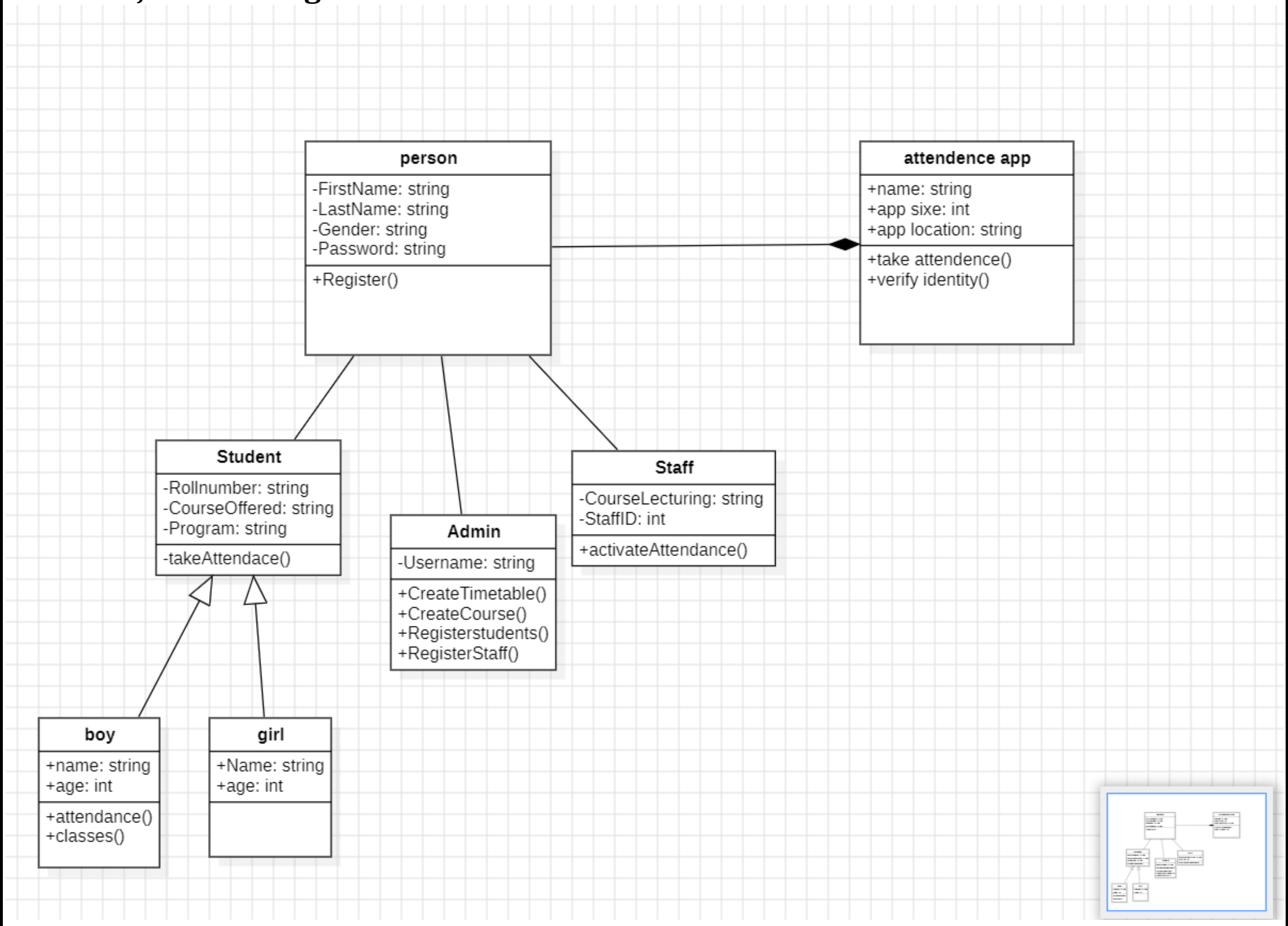


2. Online Attendence App

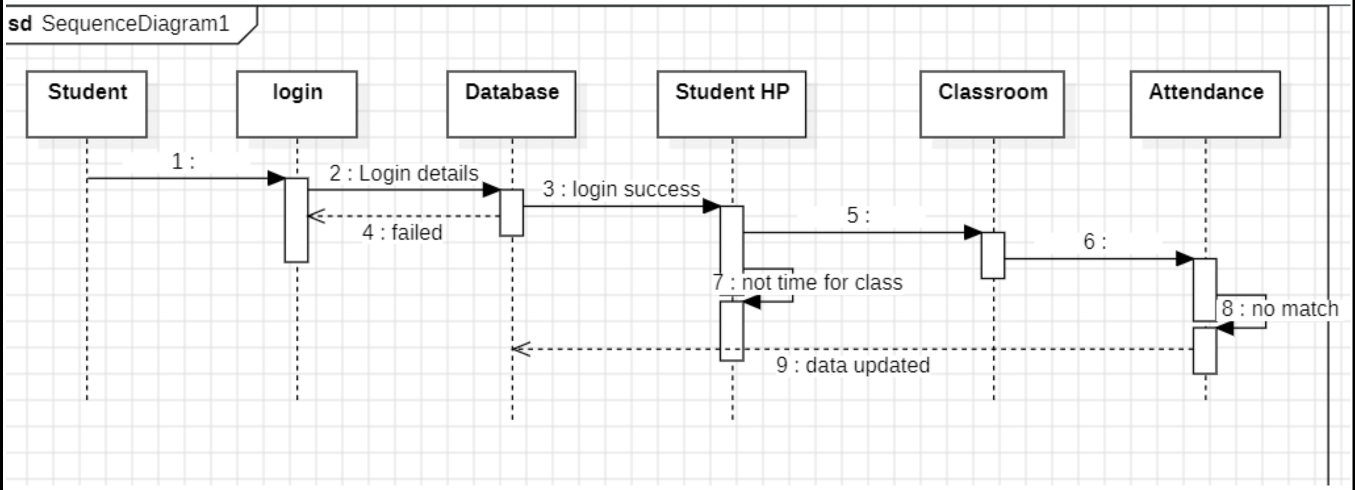
2.a) Use Case Diagram:



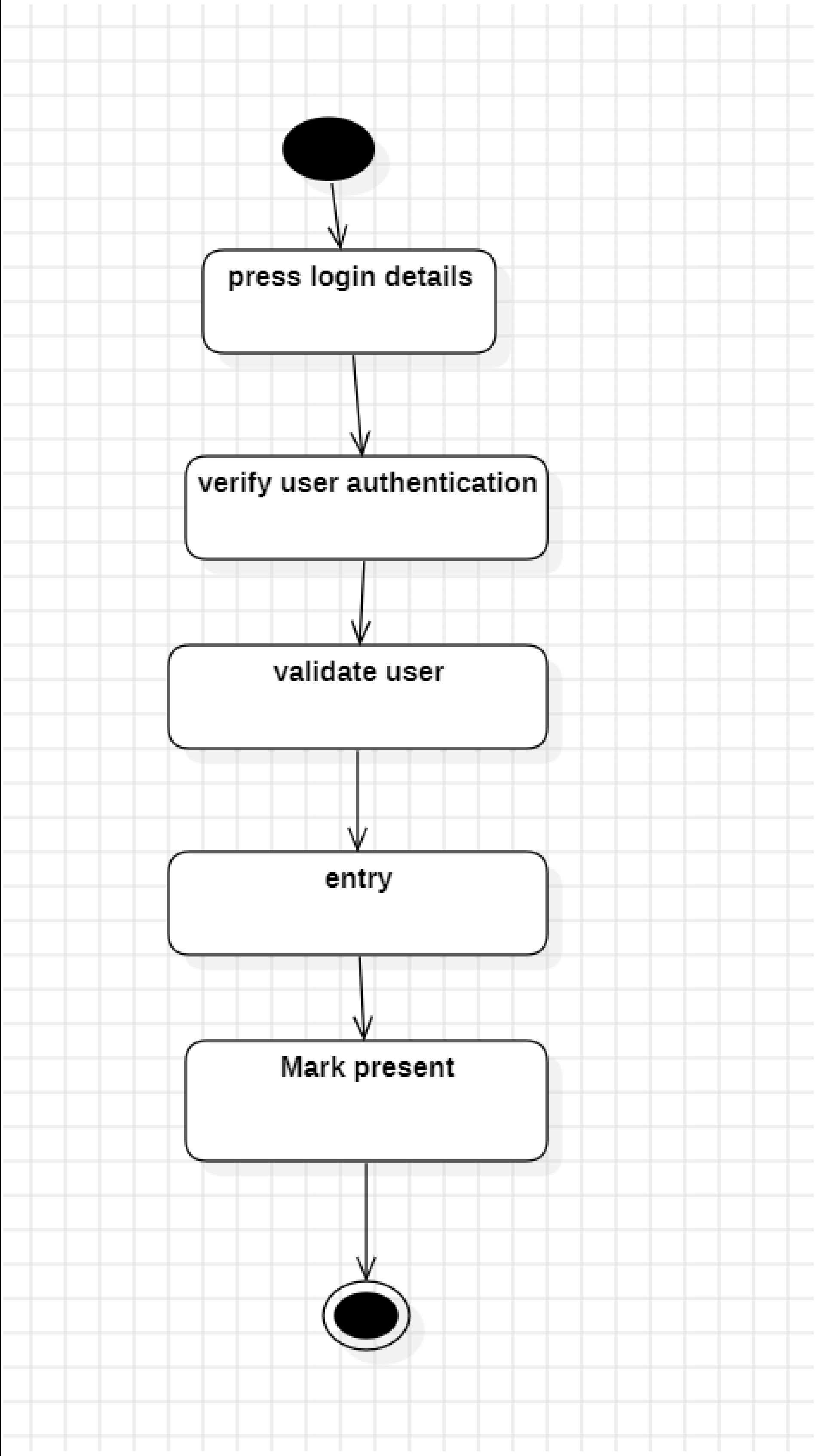
2.b) Class Diagram:



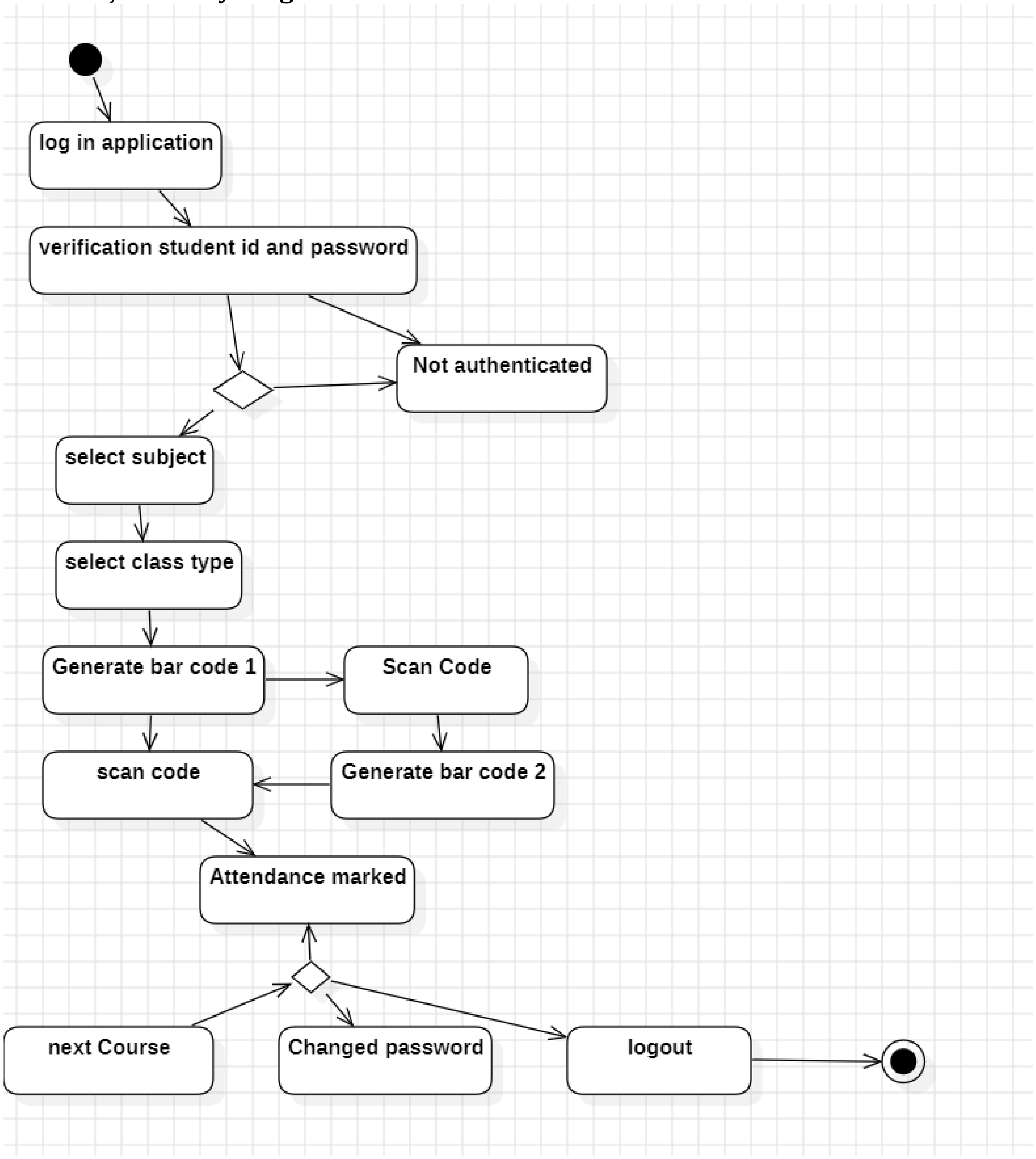
2b)Sequence Diagram:



2.c) State Diagram:



2.d) Activity Diagram:



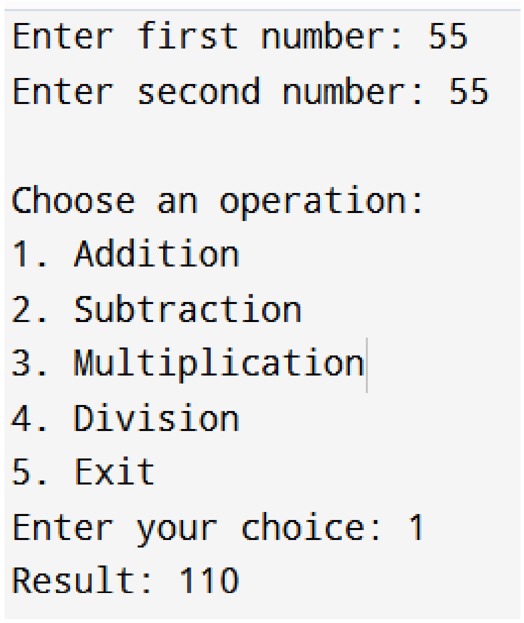
3. Basic Java Programs

3.a) Calculate two numbers:

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) {
         // Displaying the menu
         System.out.println("\nChoose an operation:");
         System.out.println("1. Addition");
         System.out.println("2. Subtraction");
         System.out.println("3. Multiplication");
         System.out.println("4. Division");
         System.out.println("5. Exit");
         System.out.print("Enter your choice: ");

         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:
               if (n != 0) {
                   div = (double) m / n;
                   System.out.println("Result: " + div);
               } else {
                   System.out.println("Division by zero is not allowed.");
               }
               break;

            case 5:
               System.out.println("Exiting program...");
               s.close(); // Close scanner before exit
               System.exit(0);

            default:
               System.out.println("Invalid option. Please try again.");
         }
      }
   }
}
```

Output:



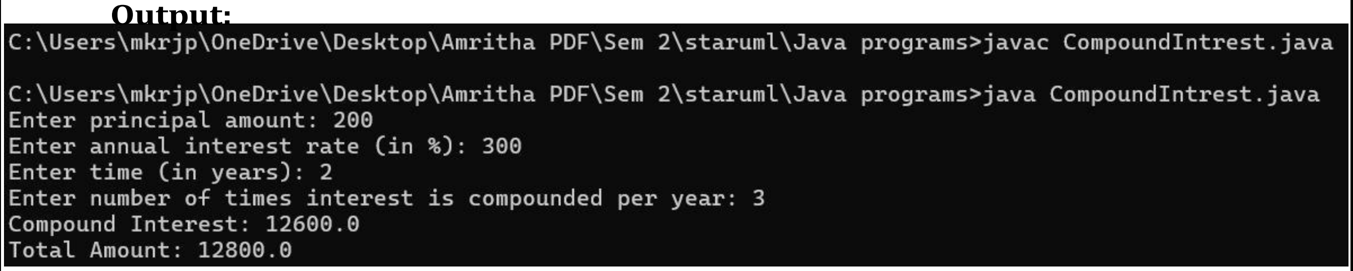
```
Enter first number: 55
Enter second number: 55

Choose an operation:
1. Addition
2. Subtraction
3. Multiplication
4. Division
5. Exit
Enter your choice: 1
Result: 110
```

3.b) CompoundInterest :

Code:

```
import java.util.Scanner;

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

        System.out.print("Enter principal amount: ");
        double principal = sc.nextDouble();

        System.out.print("Enter annual interest rate (in %): ");
        double rate = sc.nextDouble();

        System.out.print("Enter time (in years): ");
        int time = sc.nextInt();

        System.out.print("Enter number of times interest is compounded
per year: ");
        int n = sc.nextInt();

        double amount = principal * Math.pow(1 + (rate / (n * 100)), n
* time);
        double compoundInterest = amount - principal;

        System.out.println("Compound Interest: " + compoundInterest);
        System.out.println("Total Amount: " + amount);

        sc.close();
    }
}
```

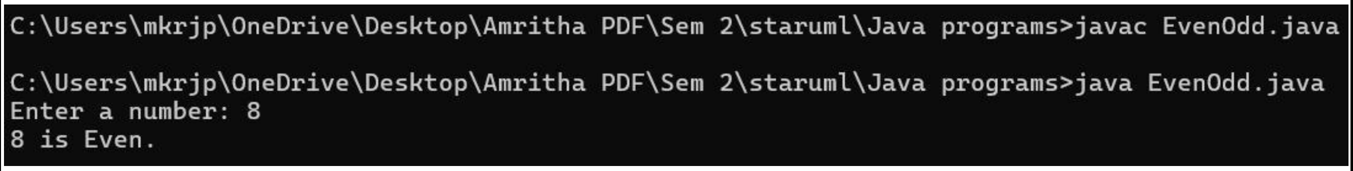
Output:



```
C:\Users\mkrjp\OneDrive\Desktop\Amritha PDF\Sem 2\staruml\Java programs>javac CompoundIntrest.java

C:\Users\mkrjp\OneDrive\Desktop\Amritha PDF\Sem 2\staruml\Java programs>java CompoundIntrest.java
Enter principal amount: 200
Enter annual interest rate (in %): 300
Enter time (in years): 2
Enter number of times interest is compounded per year: 3
Compound Interest: 12600.0
Total Amount: 12800.0
```

3.c) Even or Odd :

Code:
```
import java.util.Scanner;

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

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

        scanner.close();
    }
}
```
Output:



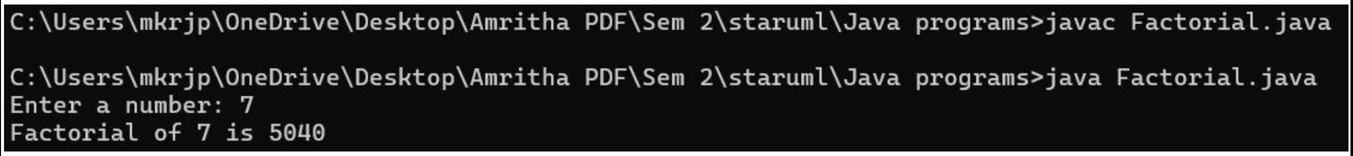
```
C:\Users\mkrjp\OneDrive\Desktop\Amritha PDF\Sem 2\staruml\Java programs>javac EvenOdd.java

C:\Users\mkrjp\OneDrive\Desktop\Amritha PDF\Sem 2\staruml\Java programs>java EvenOdd.java
Enter a number: 8
8 is Even.
```

3.d) Factorial :

Code:

```
import java.util.Scanner;

public class Factorial {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a number: ");
        int num = scanner.nextInt();
        int fact = 1;

        for (int i = 1; i <= num; i++) {
            fact *= i;
        }

        System.out.println("Factorial of " + num + " is " + fact);
        scanner.close();
    }
}
```

Output;



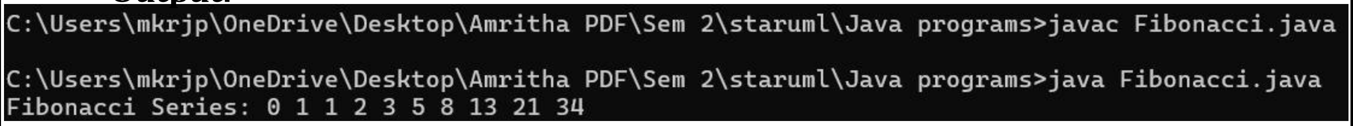
```
C:\Users\mkrjp\OneDrive\Desktop\Amritha PDF\Sem 2\staruml\Java programs>javac Factorial.java

C:\Users\mkrjp\OneDrive\Desktop\Amritha PDF\Sem 2\staruml\Java programs>java Factorial.java
Enter a number: 7
Factorial of 7 is 5040
```

3.e) Fibonacci Series :

Code:

```
public class Fibonacci {
    public static void main(String[] args) {
        int n = 10, first = 0, second = 1;

        System.out.print("Fibonacci Series: " + first + " " +
second);

        for (int i = 2; i < n; i++) {
            int next = first + second;
            System.out.print(" " + next);
            first = second;
            second = next;
        }
    }
}
```

Output:



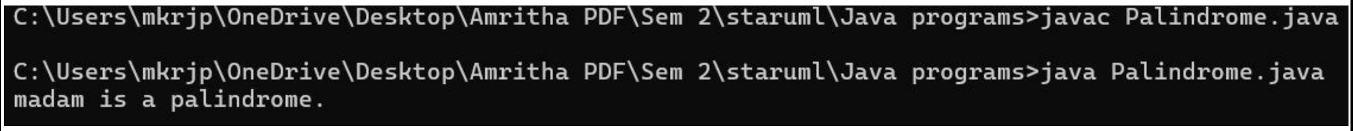
```
C:\Users\mkrjp\OneDrive\Desktop\Amritha PDF\Sem 2\staruml\Java programs>javac Fibonacci.java

C:\Users\mkrjp\OneDrive\Desktop\Amritha PDF\Sem 2\staruml\Java programs>java Fibonacci.java
Fibonacci Series: 0 1 1 2 3 5 8 13 21 34
```

3.f) Palindrone :

Code:

```
public class Palindrome
{
  public static void main(String[] args) {
  String str = "madam";    boolean isPalindrome = true;

 for (int i = 0; i < str.length() / 2; i++)
 {
  if (str.charAt(i) != str.charAt(str.length() - 1 - i))
  {
   isPalindrome = false;
   break;
  }
}
 if (isPalindrome)
 {
  System.out.println(str + " is a palindrome.");
 }
 else
 {
 System.out.println(str + " is not a palindrome.");
 }
}
}
```

Output:



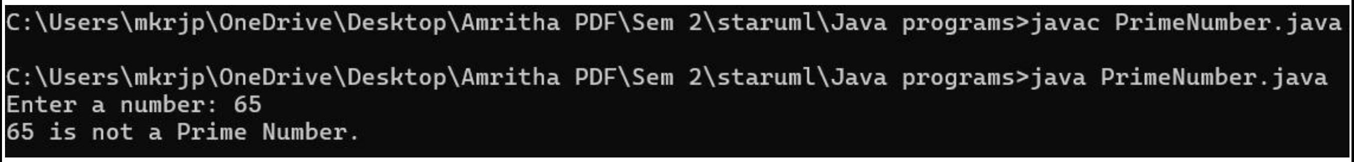
```
C:\Users\mkrjp\OneDrive\Desktop\Amritha PDF\Sem 2\staruml\Java programs>javac Palindrome.java

C:\Users\mkrjp\OneDrive\Desktop\Amritha PDF\Sem 2\staruml\Java programs>java Palindrome.java
madam is a palindrome.
```

3.g) Prime Checker:

Code:

```
import java.util.Scanner;

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

        if (num <= 1) {
            isPrime = false;
        } else {
            for (int i = 2; i <= Math.sqrt(num); i++) {
                if (num % i == 0) {
                    isPrime = false;
                    break;
                }
            }
        }

        if (isPrime) {
            System.out.println(num + " is a Prime Number.");
        } else {
            System.out.println(num + " is not a Prime Number.");
        }

        scanner.close();
    }
}
```

Output:



```
C:\Users\mkrjp\OneDrive\Desktop\Amritha PDF\Sem 2\staruml\Java programs>javac PrimeNumber.java

C:\Users\mkrjp\OneDrive\Desktop\Amritha PDF\Sem 2\staruml\Java programs>java PrimeNumber.java
Enter a number: 65
65 is not a Prime Number.
```

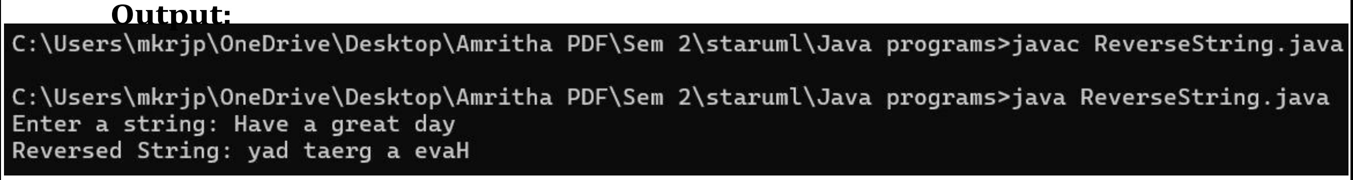
:

3h)Reverse String :

Code:

```
import java.util.Scanner;

public class ReverseString {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a string: ");
        String str = scanner.nextLine();
        String reversed = "";

        for (int i = str.length() - 1; i >= 0; i--) {
            reversed += str.charAt(i);
        }

        System.out.println("Reversed String: " + reversed);
        scanner.close();
    }
            }
```

Output:



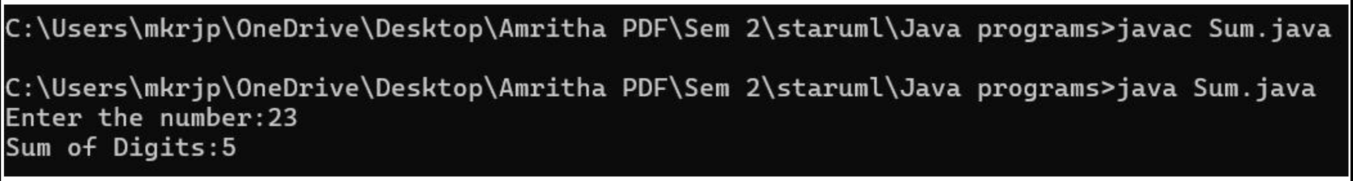
```
C:\Users\mkrjp\OneDrive\Desktop\Amritha PDF\Sem 2\staruml\Java programs>javac ReverseString.java

C:\Users\mkrjp\OneDrive\Desktop\Amritha PDF\Sem 2\staruml\Java programs>java ReverseString.java
Enter a string: Have a great day
Reversed String: yad taerg a evaH
```

3.h) Sum of digits :

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:



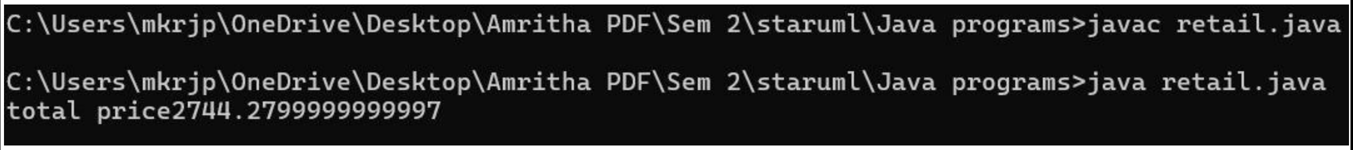
```
C:\Users\mkrjp\OneDrive\Desktop\Amritha PDF\Sem 2\staruml\Java programs>javac Sum.java

C:\Users\mkrjp\OneDrive\Desktop\Amritha PDF\Sem 2\staruml\Java programs>java Sum.java
Enter the number:23
Sum of Digits:5
```

3.i) Retail :

Code:

```
import util.java.*;

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

int  itema=100; int itemb=200; int itemc=400;
double price;

price=((itema*2)+(itemb*5)+(itemc*4));
price=price-(0.01*price);
price=price+(0.1*price);

if(price>=2000);
  price=price-(0.1*price);
System.out.println("total price" +price);
else
  System.out.primtln("not applicable for discount");
}
}
```

Output:



```
C:\Users\mkrjp\OneDrive\Desktop\Amritha PDF\Sem 2\staruml\Java programs>javac retail.java

C:\Users\mkrjp\OneDrive\Desktop\Amritha PDF\Sem 2\staruml\Java programs>java retail.java
total price2744.2799999999997
```

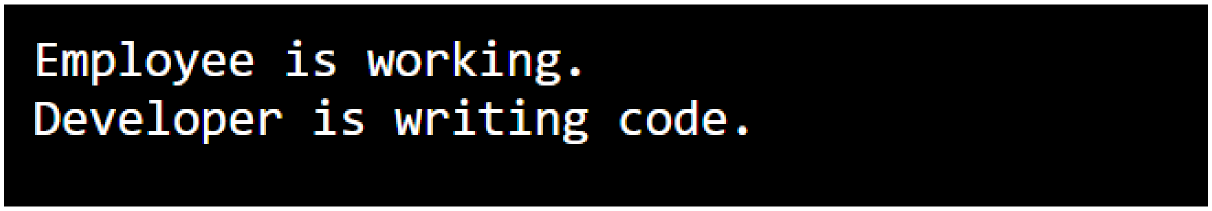
4.Single inheritance Programs

4 a) EMPLOYEE-developer

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:



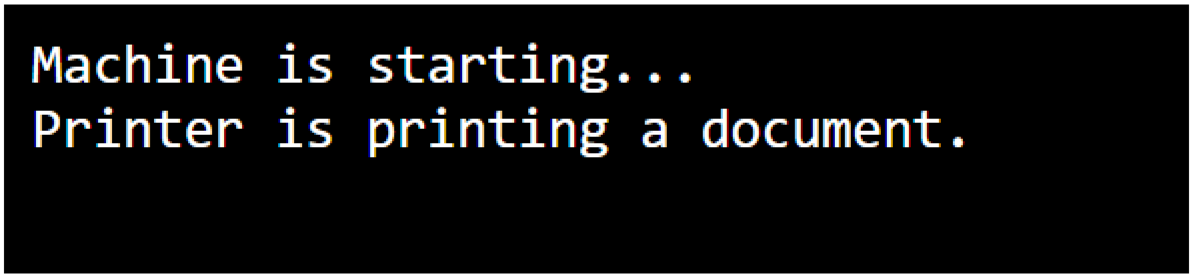
```
Employee is working.
Developer is writing code.
```

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:



```
Machine is starting...
Printer is printing a document.
```

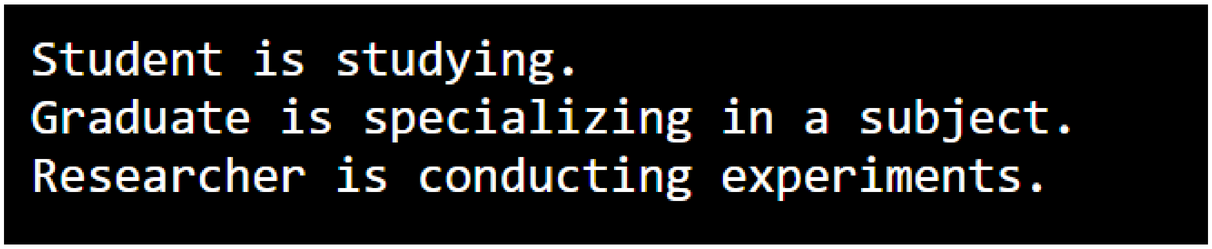
5.multilevel inheritance Programs

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:



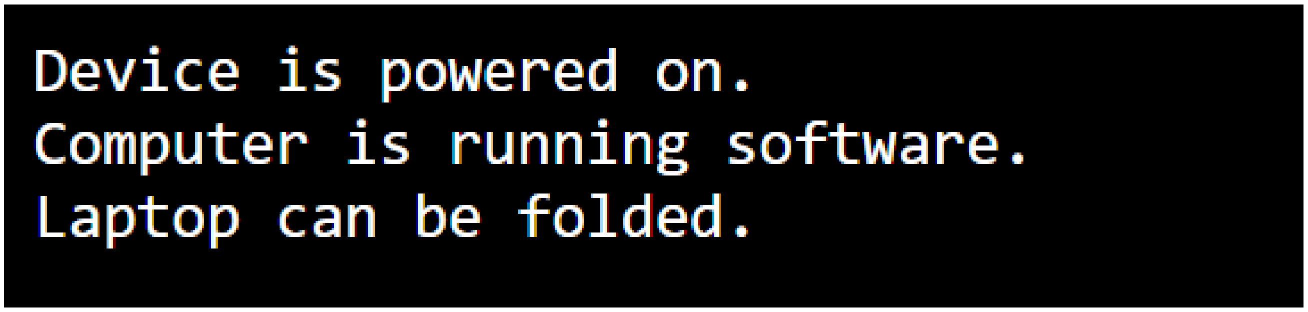
```
Student is studying.
Graduate is specializing in a subject.
Researcher is conducting experiments.
```

5 B) device-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:



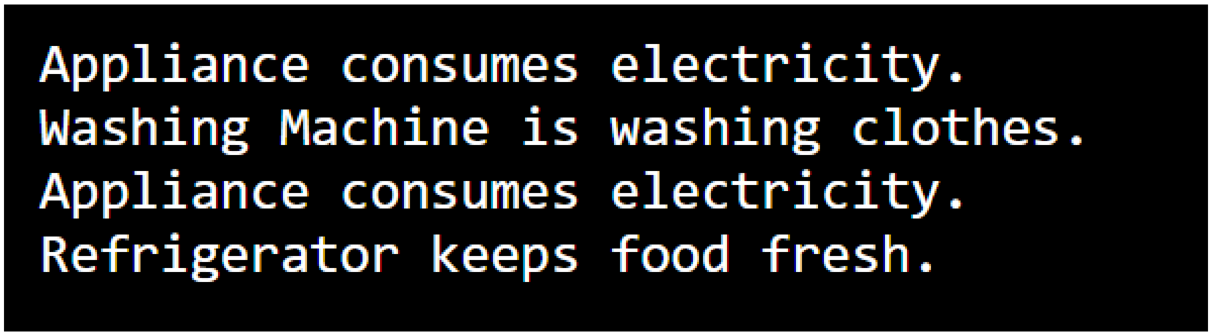
```
Device is powered on.
Computer is running software.
Laptop can be folded.
```

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:



```
Appliance consumes electricity.
Washing Machine is washing clothes.
Appliance consumes electricity.
Refrigerator keeps food fresh.
```

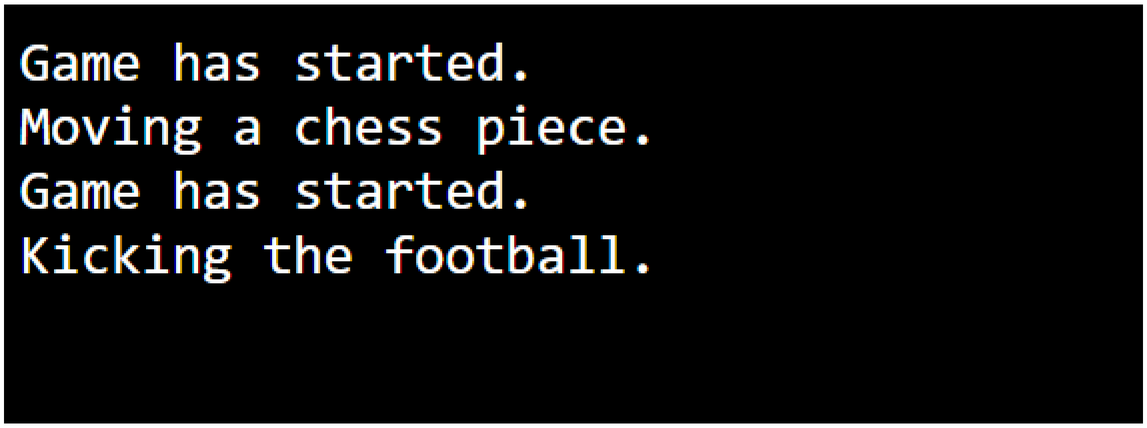
6 b) Game:

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:



```
Game has started.
Moving a chess piece.
Game has started.
Kicking the football.
```

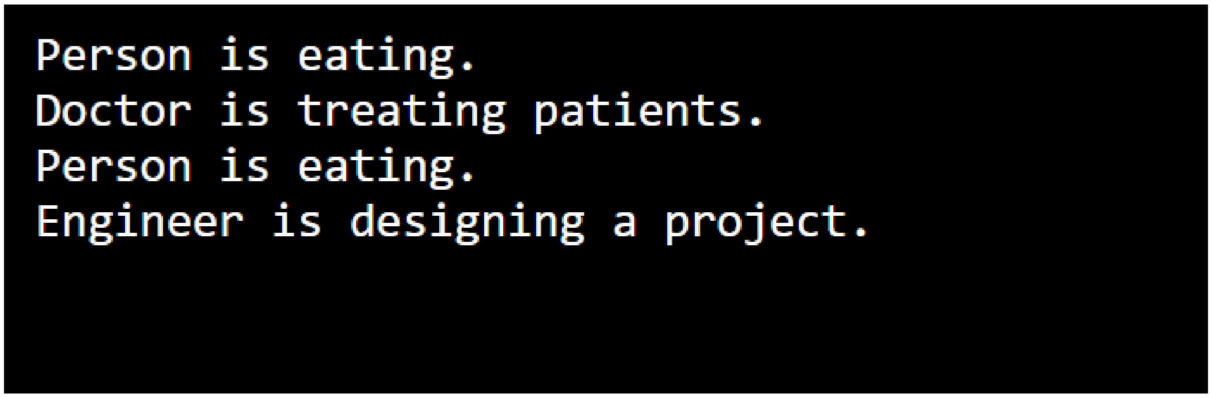
7.hybrid inheritance Programs

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:



```
Person is eating.
Doctor is treating patients.
Person is eating.
Engineer is designing a project.
```

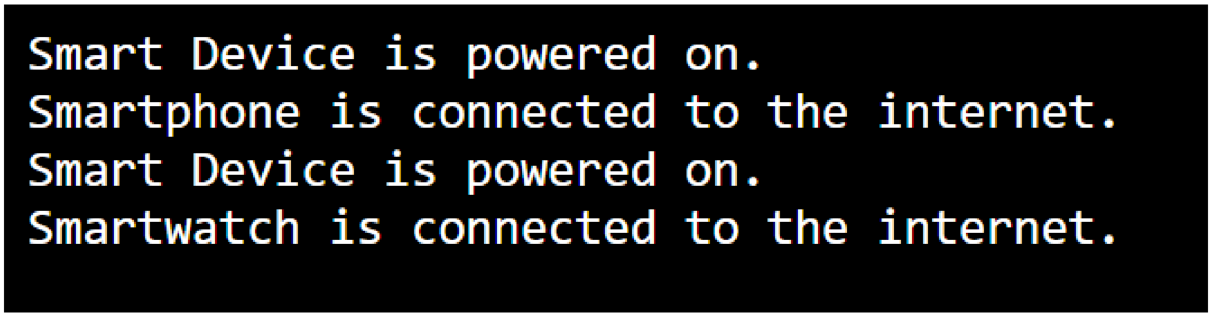
7 b) smart device

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:



```
Smart Device is powered on.
Smartphone is connected to the internet.
Smart Device is powered on.
Smartwatch is connected to the internet.
```

POLYMORPHISM

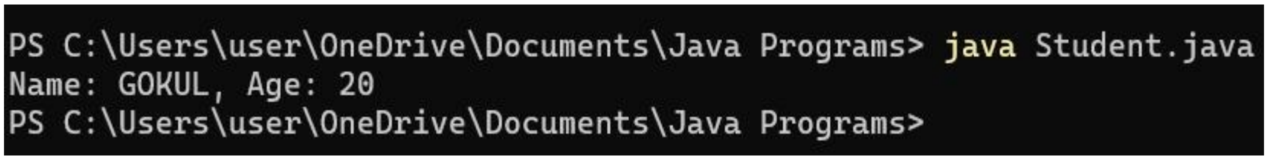
8.Constructor Programs

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:



```
PS C:\Users\user\OneDrive\Documents\Java Programs> java Student.java
Name: GOKUL, Age: 20
PS C:\Users\user\OneDrive\Documents\Java Programs>
```

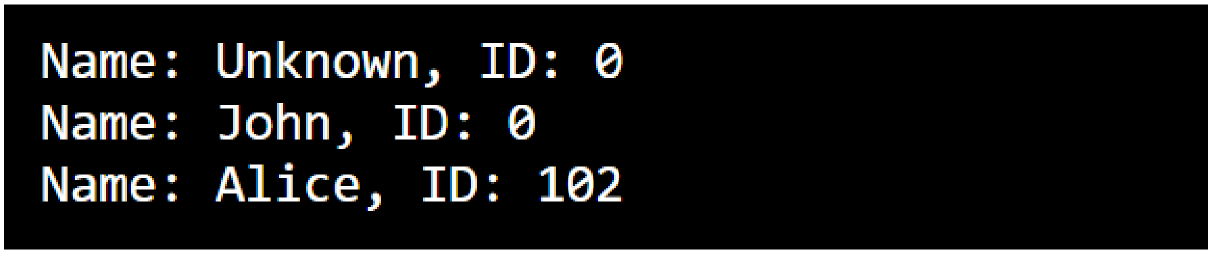
1. Constructor overloading Programs

9 a) Employee Constructor Overloading

CODE:

```
 class Employee {
  String name;
  int id;


  Employee() {
    name = "Unknown";
    id = 0;
  }


  Employee(String n) {
    name = n;
    id = 0;
  }


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

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

  public static void main(String[] args) {
    Employee e1 = new Employee();
    Employee e2 = new Employee("John");
    Employee e3 = new Employee("Alice", 102);

    e1.display();
    e2.display();
    e3.display();
  }
}
```

OUTPUT:



```
Name: Unknown, ID: 0
Name: John, ID: 0
Name: Alice, ID: 102
```

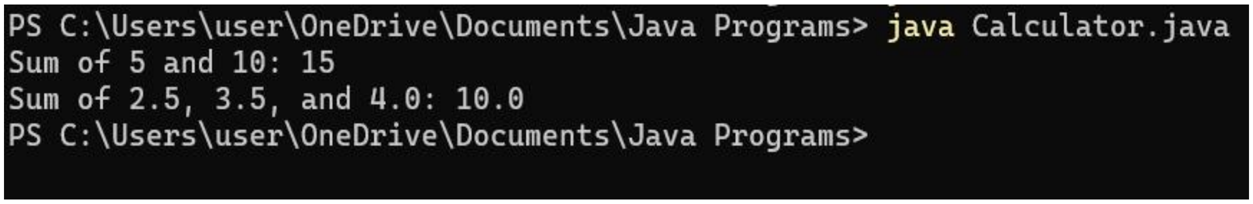
10.Method overloading Programs

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:



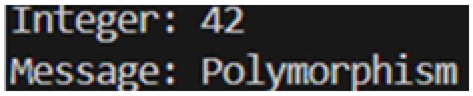
```
PS C:\Users\user\OneDrive\Documents\Java Programs> java Calculator.java
Sum of 5 and 10: 15
Sum of 2.5, 3.5, and 4.0: 10.0
PS C:\Users\user\OneDrive\Documents\Java Programs>
```

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:



```
Integer: 42
Message: Polymorphism
```

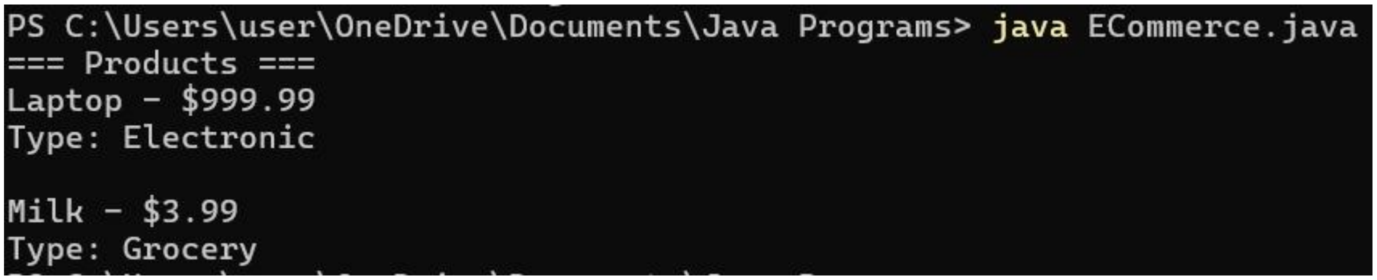
11.METHOD OVERRIDING

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:



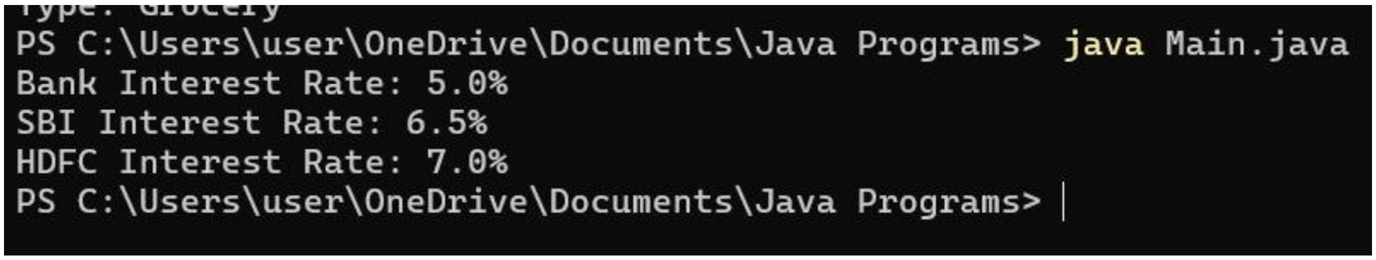
```
PS C:\Users\user\OneDrive\Documents\Java Programs> java ECommerce.java
=== Products ===
Laptop - $999.99
Type: Electronic

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:



```
Type: Grocery
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>
```

ABSTRACTION

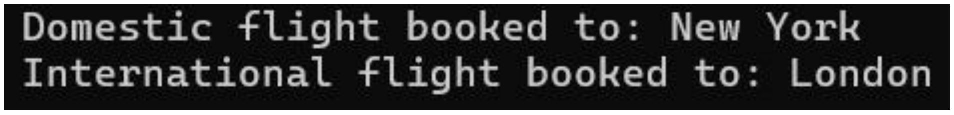
12.INTERFACE PROGRAMS

12a) FlightReservationSystem

CODE:

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

class DomesticFlight implements FlightBooking {
    public void bookTicket(String destination) {
        System.out.println("Domestic flight booked to: " + destination);
    }
}

class InternationalFlight implements FlightBooking {
    public void bookTicket(String destination) {
        System.out.println("International flight booked to: " + destination);
    }
}

public class FlightReservationSystem {
    public static void main(String[] args) {
        FlightBooking domestic = new DomesticFlight();
        FlightBooking international = new InternationalFlight();

        domestic.bookTicket("New York");
        international.bookTicket("London");
    }
}
```

OUTPUT:



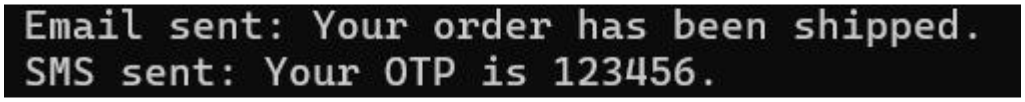
```
Domestic flight booked to: New York
International flight booked to: London
```

12b) NotificationSystem

CODE:

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

class EmailNotification implements Notification {
    public void sendNotification(String message) {
        System.out.println("Email sent: " + message);
    }
}

class SMSNotification implements Notification {
    public void sendNotification(String message) {
        System.out.println("SMS sent: " + message);
    }
}

public class NotificationSystem {
    public static void main(String[] args) {
        Notification email = new EmailNotification();
        Notification sms = new SMSNotification();

        email.sendNotification("Your order has been shipped.");
        sms.sendNotification("Your OTP is 123456.");
    }
}
```

OUTPUT:



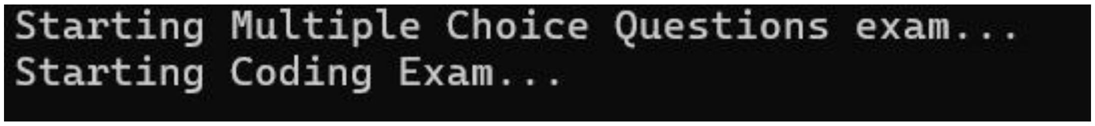
```
Email sent: Your order has been shipped.
SMS sent: Your OTP is 123456.
```

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:



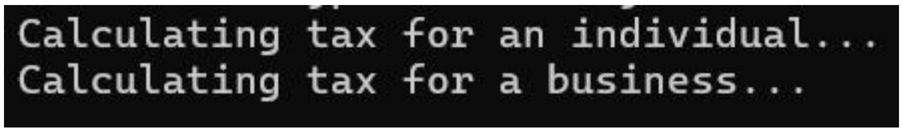
```
Starting Multiple Choice Questions exam...
Starting Coding Exam...
```

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:



```
Calculating tax for an individual...
Calculating tax for a business...
```

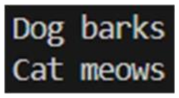
13. ABSTRACT CLASS PROGRAMS

13a)Animal

CODE:

```
Animal {
  abstract void makeSound();
 }

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

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

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

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

OUTPUT:



```
Dog barks
Cat meows
```

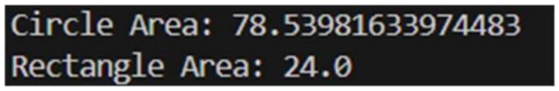
13.b) Shape

CODE:

```
1.a)  abstract class
Shape {
   abstract double calculateArea();
}

Class Circle extends Shape {
   double radius;

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

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

class Rectangle extends Shape {
   double length, breadth;

   public Rectangle(double length, double breadth) {
      this.length = length;
      this.breadth = breadth;
   }

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

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

      System.out.println("Circle Area: " + circle.calculateArea());
```

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

OUTPUT:



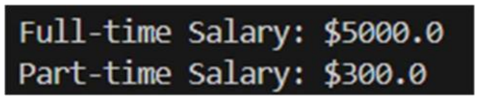
```
Circle Area: 78.53981633974483
Rectangle Area: 24.0
```

13.c) Employee-Salary:

CODE:

```
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());
  }
}
```

OUTPUT:



```
Full-time Salary: $5000.0
Part-time Salary: $300.0
```

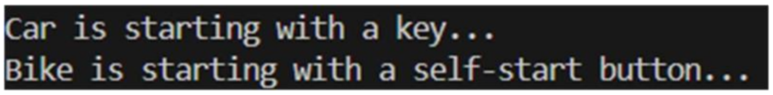
13.d)Vehicle:

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();
   }
}
```

CODE:



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

ENCAPSULATION

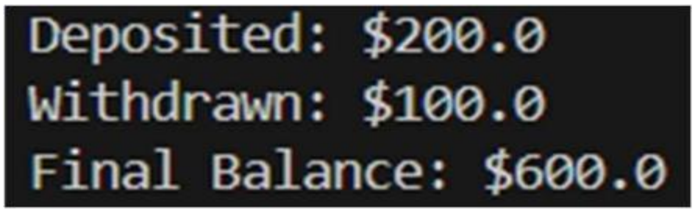
14.a) BanK Account

CODE:

```
class BankAccount {
   private String accountNumber;
   private double balance;

   public BankAccount(String accountNumber, double balance) {
      this.accountNumber = accountNumber;
      if (balance >= 0) {
         this.balance = balance;
      } else {
         this.balance = 0;
         System.out.println("Balance cannot be negative. Setting to 0.");
      }
   }

   public void deposit(double amount) { if
      (amount > 0) {
         balance += amount;
         System.out.println("Deposited: $" + amount);
      } else {
         System.out.println("Invalid deposit amount.");
      }
   }

   public void withdraw(double amount) {
      if (amount > 0 && amount <= balance) {
         balance -= amount;
         System.out.println("Withdrawn: $" + amount);
      } else {
         System.out.println("Invalid withdrawal amount or insufficient balance.");
      }
   }
```

```
    public double getBalance() {
        return balance;
    }

    public String getAccountNumber() {
        return accountNumber;
    }
    public static void main(String[] args) {
        BankAccount account = new BankAccount("123456789", 500);
        account.deposit(200);
        account.withdraw(100);
        System.out.println("Final Balance: $" + account.getBalance());
    }
}
```

OUTPUT:



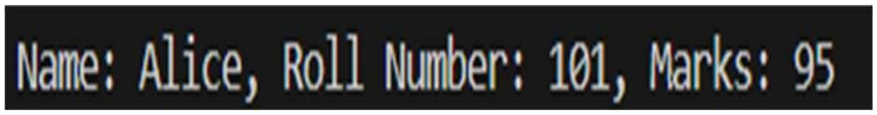
```
Deposited: $200.0
Withdrawn: $100.0
Final Balance: $600.0
```

14.b)Student:

CODE:

```
class Student {
    private String name; private
    int rollNumber; private int
    marks;

    public Student(String name, int rollNumber, int marks) {
        this.name = name;
        this.rollNumber = rollNumber;
        setMarks(marks);
    }

    public void setMarks(int marks) {
        if (marks >= 0 && marks <= 100) {
            this.marks = marks;
        } else {
            System.out.println("Invalid marks! Setting marks to 0.");
            this.marks = 0;
        }
    }

    public int getMarks() {
        return marks;
    }

    public void display() {
        System.out.println("Name: " + name + ", Roll Number: " + rollNumber + ", Marks: "
+ marks);
    }
    public static void main(String[] args) {
        Student student = new Student("Alice", 101, 95);
        student.display();
    }
}
```

OUTPUT:



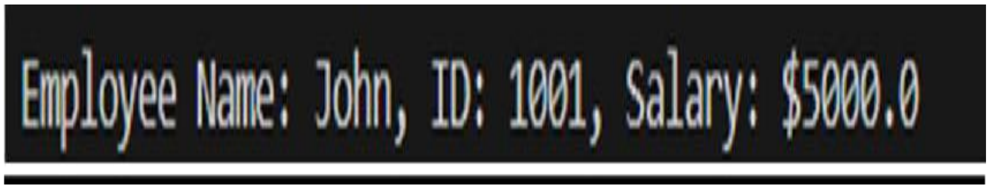
```
Name: Alice, Roll Number: 101, Marks: 95
```

14.c)Employee:

CODE:

```
class Employee { private
    String name; private
    int id;
    private double salary;

    public Employee(String name, int id, double salary) {
        this.name = name;
        this.id = id;
        setSalary(salary);
    }

    public void setSalary(double salary) { if
        (salary >= 0) {
            this.salary = salary;
        } else {
            System.out.println("Salary cannot be negative. Setting to 0.");
            this.salary = 0;
        }
    }

    public double getSalary() {
        return salary;
    }

    public void display() {
        System.out.println("Employee Name: " + name + ", ID: " + id + ", Salary: $" +
  salary);
    }

    public static void main(String[] args) {
        Employee emp = new Employee("John", 1001, 5000);
        emp.display();
        }
}
```

OUTPUT:



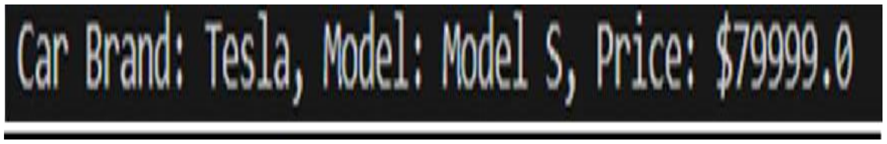
```
Employee Name: John, ID: 1001, Salary: $5000.0
```

14.d) Car:

CODE:

```
class Car {
   private  String  brand;
   private  String  model;
   private double price;

   public Car(String brand, String model, double price) {
      this.brand = brand;
      this.model = model;
      setPrice(price);
   }

   public void setPrice(double price) { if
      (price > 0) {
         this.price = price;
      } else {
         System.out.println("Price cannot be zero or negative. Setting to default
$10,000.");
         this.price = 10000;
      }
   }

   public double getPrice() {
      return price;
   }

   public void display() {
      System.out.println("Car Brand: " + brand + ", Model: " + model + ", Price: $" +
price);
   }

   public static void main(String[] args) {
      Car car = new Car("Tesla", "Model S", 79999);
      car.display();
   }
}
```

OUTPUT :



```
Car Brand: Tesla, Model: Model S, Price: $79999.0
```

15.PACKAGES PROGRAMS

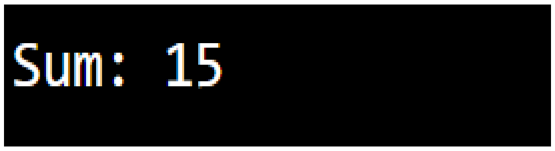
15 a) user defined package

Package file:

CODE:

```
package mathoperations;

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
```

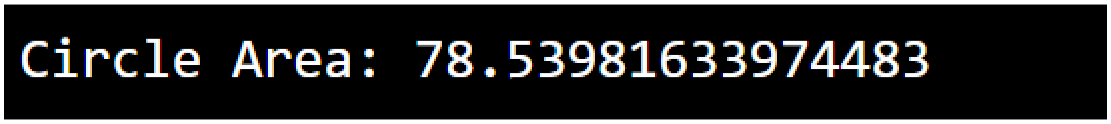
15 b) user defined package

Package file:

CODE:

```
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
```

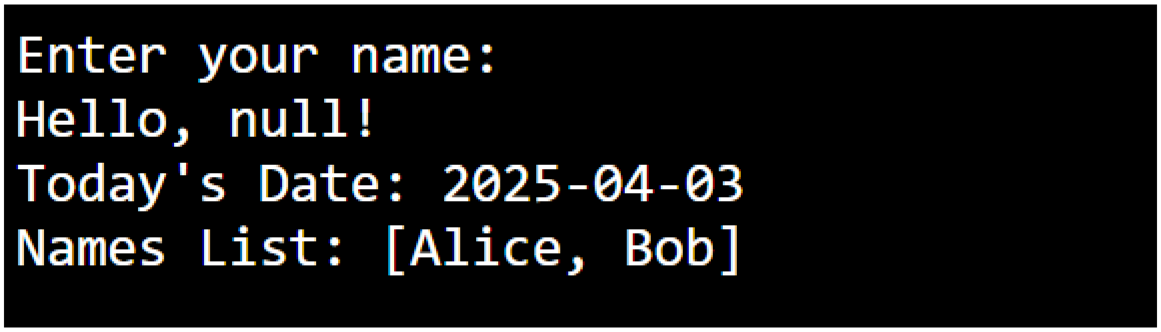
Built-In Packages

16 a) built in packages

CODE:

```
import java.util.ArrayList;
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.time.LocalDate;

public class BuiltInPackageExample1 {
    public static void main(String[] args) throws Exception {
        // Using java.util.ArrayList
        ArrayList<String> names = new ArrayList<>();
        names.add("Alice");
        names.add("Bob");

        // Using java.io.BufferedReader
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        System.out.println("Enter your name: ");
        String userName = br.readLine();

        // Using java.time.LocalDate
        LocalDate today = LocalDate.now();

        System.out.println("Hello, " + userName + "!");
        System.out.println("Today's Date: " + today);
        System.out.println("Names List: " + names);
    }
}
```

OUTPUT:



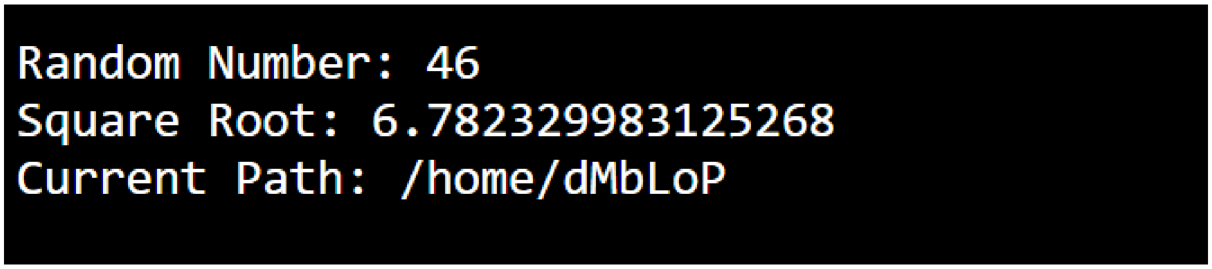
```
Enter your name:
Hello, null!
Today's Date: 2025-04-03
Names List: [Alice, Bob]
```

16 b) built in packages

CODE:

```
import java.util.Random;
import java.lang.Math;
import java.nio.file.Paths;

public class BuiltInPackageExample2 {
    public static void main(String[] args) {
        // Using java.util.Random
        Random rand = new Random();
        int randomNum = rand.nextInt(100);
        System.out.println("Random Number: " + randomNum);

        // Using java.lang.Math
        double squareRoot = Math.sqrt(randomNum);
        System.out.println("Square Root: " + squareRoot);

        System.out.println("Current Path: " + Paths.get("").toAbsolutePath());
    }
}
```

OUTPUT:



```
Random Number: 46
Square Root: 6.782329983125268
Current Path: /home/dMbLoP
```

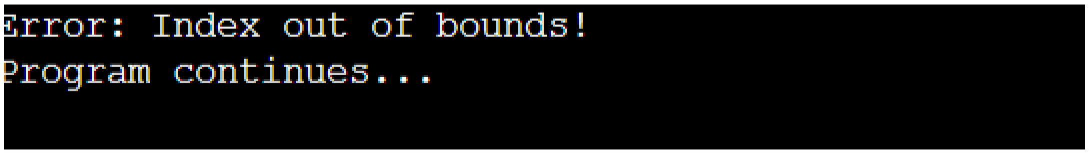
17. Exception Handling Programs

17.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:



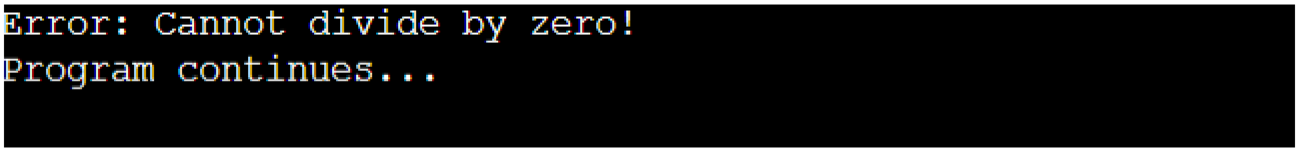
```
rror: Index out of bounds!
rogram continues...
```

17.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 :



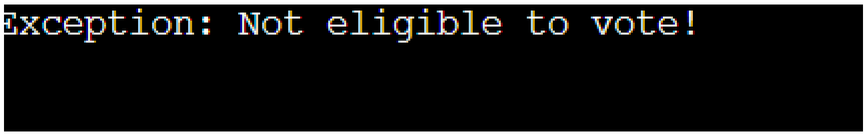
```
Error: Cannot divide by zero!
Program continues...
```

17.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:



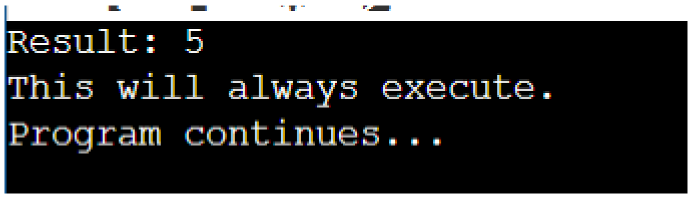
```
Exception: Not eligible to vote!
```

17.d) 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:



```
Result: 5
This will always execute.
Program continues...
```

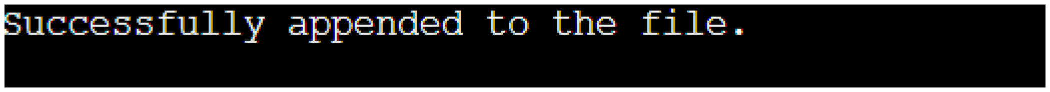
18.File Handling Programs

18.a) Append a Text File :

CODE :

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

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

OUTPUT:



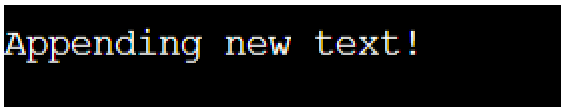
```
Successfully appended to the file.
```

18.b) Read File

CODE:

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

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

OUTPUT:



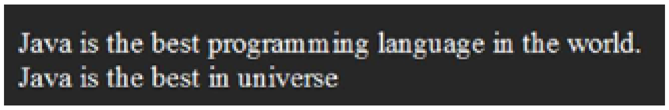
```
Appending new text!
```

18.c)Replace word

CODE:

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

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

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

OUTPUT:



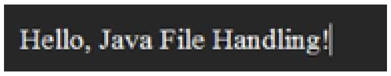
```
Java is the best programming language in the world.
Java is the best in universe
```

18.d) Writing to a Text File

CODE:

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

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

OUTPUT:



```
Hello, Java File Handling!
```