LAB RECORD

23CSE111- Object Oriented Programming

Submitted by

CH.SC.U4CSE24023 – Keerthana Varapradha N B

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.U4CSE24023 – Keerthana Varapradha N B*** 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

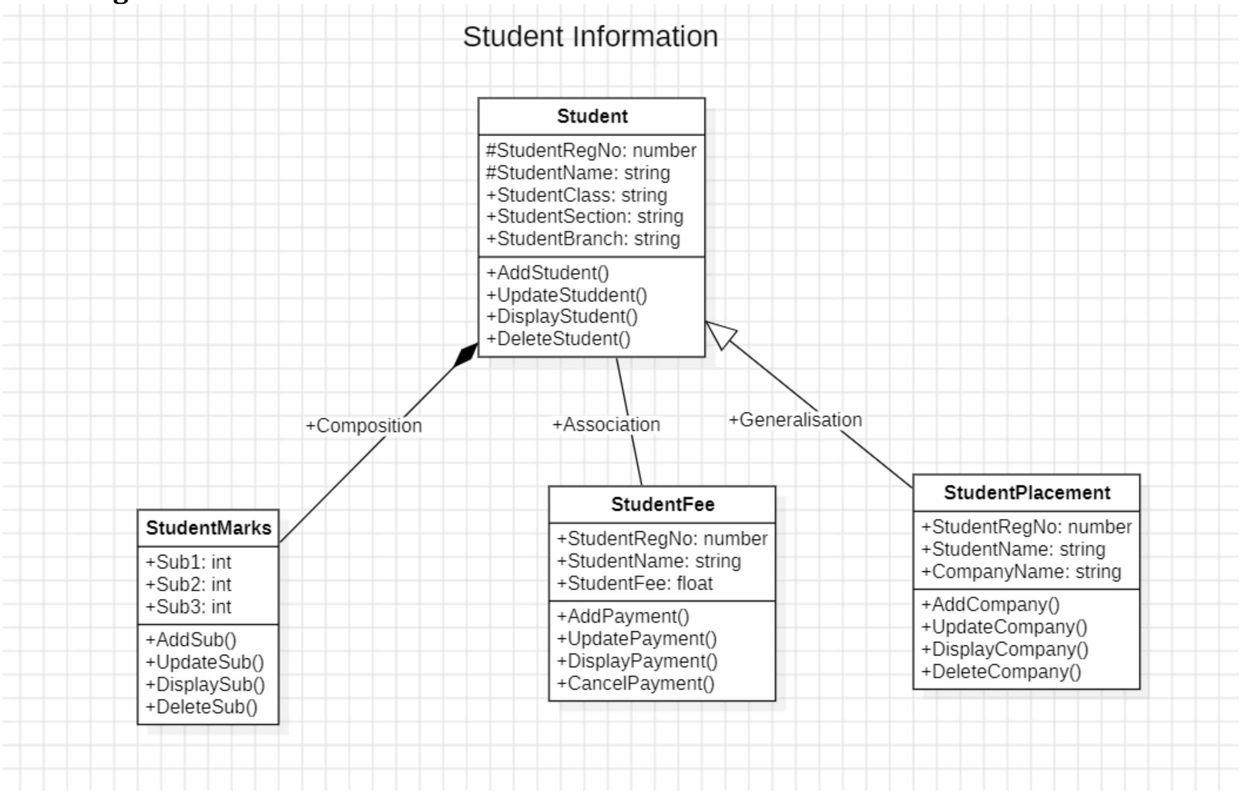
INDEX

1 UML Diagrams - 1

1) Draw a UML class diagram for a Student Information System.

AIM: To model a Student Information System managing student details, marks, fees and placements using UML class diagram concept.

UML diagram:

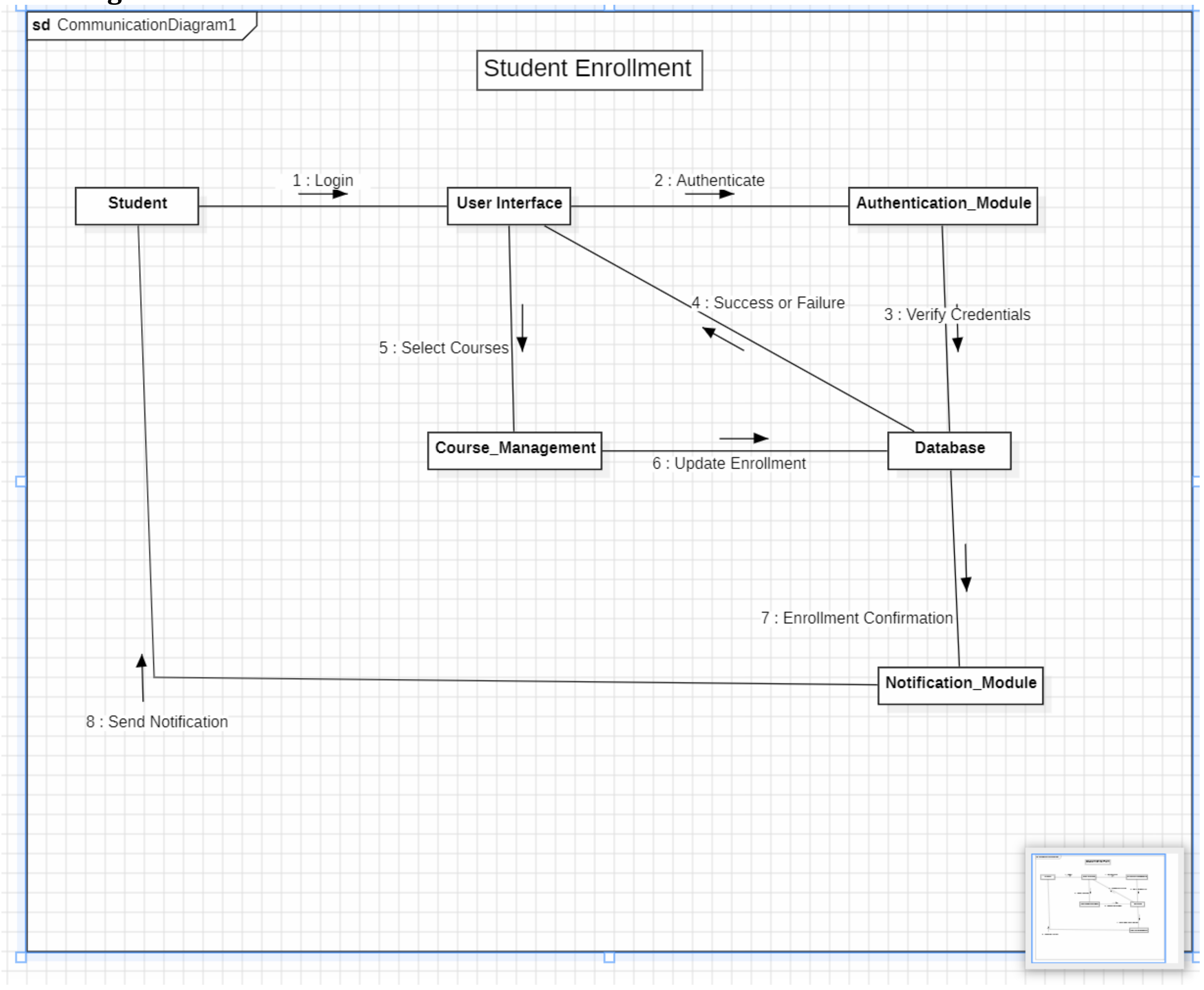


Result: The Student class stores student details, while StudentMarks (composition) depends on it for marks. StudentFee (association) tracks payment, and StudentPlacement (generalization) extends it for placements.

2) Draw a communication diagram for a Student Enrollment System.

AIM: To model the interaction between different components in a Student Enrollment System using a UML Communication Diagram.

UML diagram:

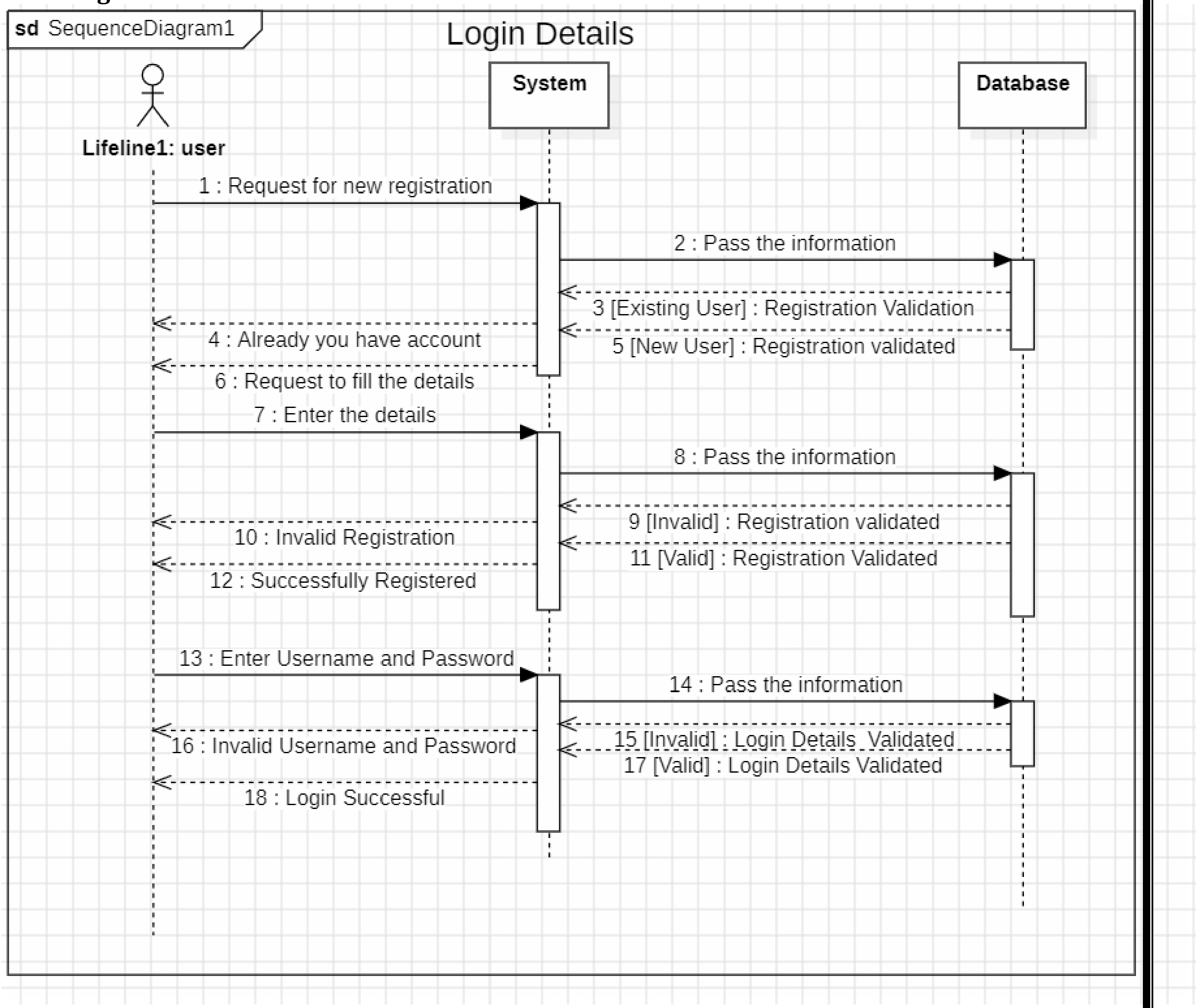


Result: The Student logs in via the User Interface, which Authenticates through the Authentication Module and Database. Upon successful login, the student selects courses, and Course Management updates enrollment in the Database. Finally, the notification module confirms enrollment and notifies the student.

3) Draw a sequence diagram for a Login System.

AIM: To illustrate the step-by-step interaction between the User, System and Database during the login and registration process using a UML sequence diagram.

UML diagram:

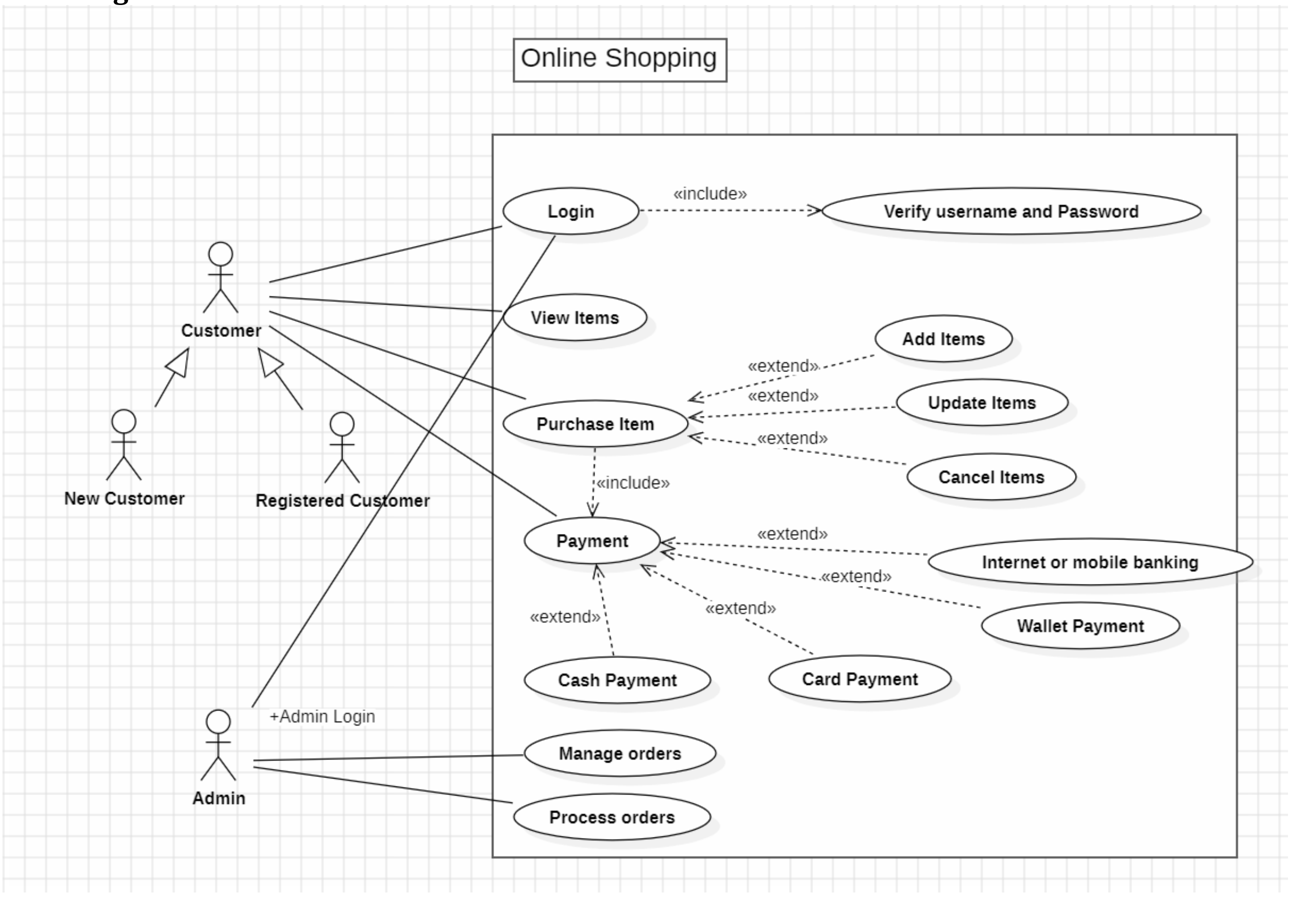

sd SequenceDiagram1
Login Details
Lifeline1: user
System
Database
1 : Request for new registration
2 : Pass the information
3 [Existing User] : Registration Validation
4 : Already you have account
5 [New User] : Registration validated
6 : Request to fill the details
7 : Enter the details
8 : Pass the information
9 [Invalid] : Registration validated
10 : Invalid Registration
11 [Valid] : Registration Validated
12 : Successfully Registered
13 : Enter Username and Password
14 : Pass the information
15 [Invalid] : Login Details Validated
16 : Invalid Username and Password
17 [Valid] : Login Details Validated
18 : Login Successful

Result: The User requests registration, and the System verifies the details with the Database. If valid, the user is registered; otherwise, an error is returned. During login, the user enters credentials, which are validated by the system. If correct, login is successful; otherwise, an error message is displayed.

4) Draw a use case diagram for an Online Shopping System.

AIM: To illustrate the interactions and functionalities of an Online Shopping System use a UML Use Case Diagram, highlighting the roles of different actors and their relationships with core operators.

UML diagram:

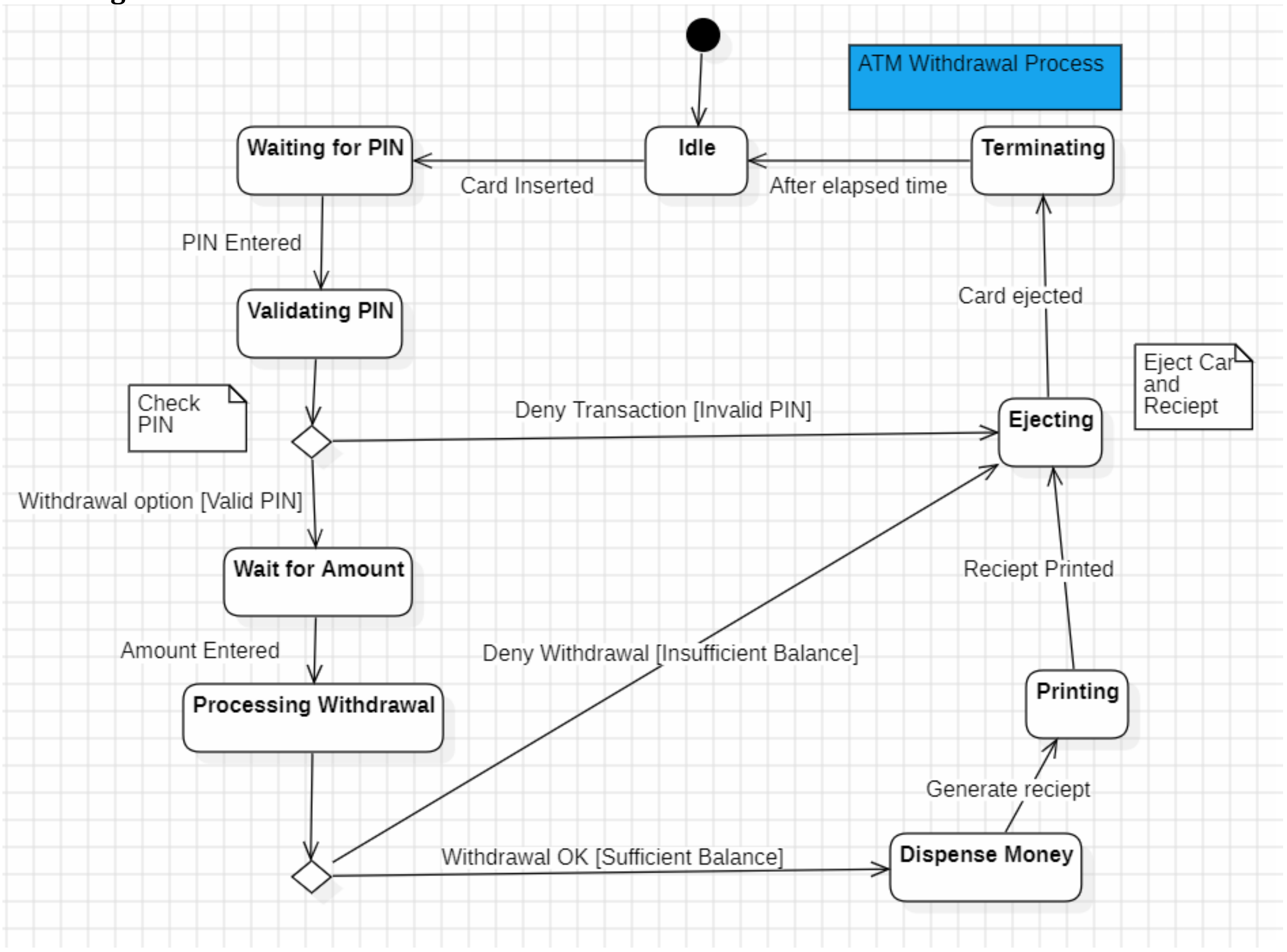


Result: The Diagram shows New Customer, Registered Customer, and Admin interacting with use cases like Login, View Items, Purchase, Payment and Order Management. It also includes Add, Update, Cancel Items, and multiple payment methods, outlining the system's key functions.

5) Draw a state diagram for an ATM Withdrawal Process, illustrate validation, decision-making, and money dispensing.

AIM: To represent the step-by-step flow of an ATM withdrawal, focusing on user actions, system validation and transaction completion.

UML diagram:



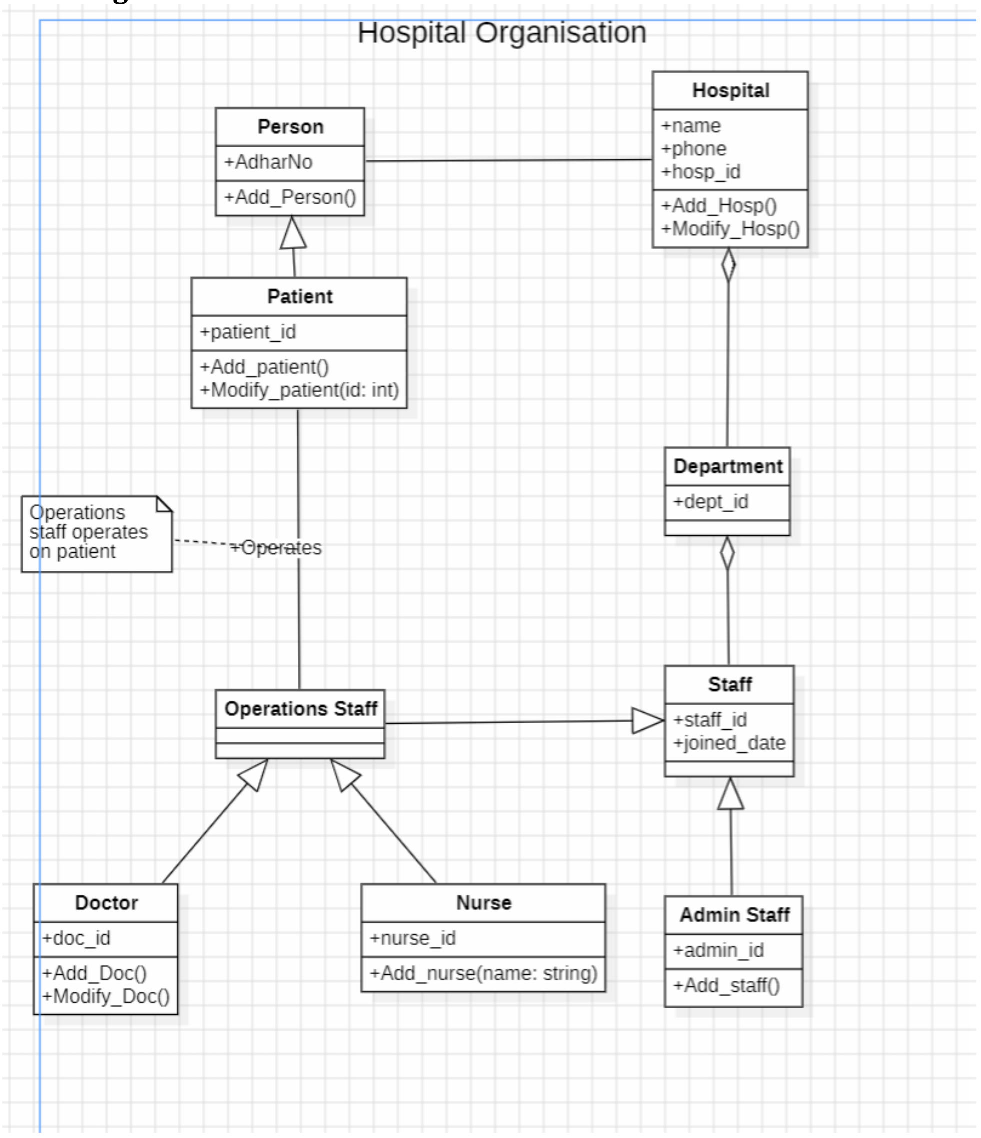
Result: The flowchart starts with card insertion, followed by PIN entry, validation and amount selection. It includes decision points for PIN verification and balance sufficiency. The process ends with money dispensing, receipt printing and card ejection, ensuring a complete transaction.

2 UML Diagrams - 2

1) Draw a UML class diagram representing the organizational structure and entities of a Hospital System.

AIM: To illustrate the hierarchy, attributes, methods, and relationships among hospital entities for effective management.

UML diagram:

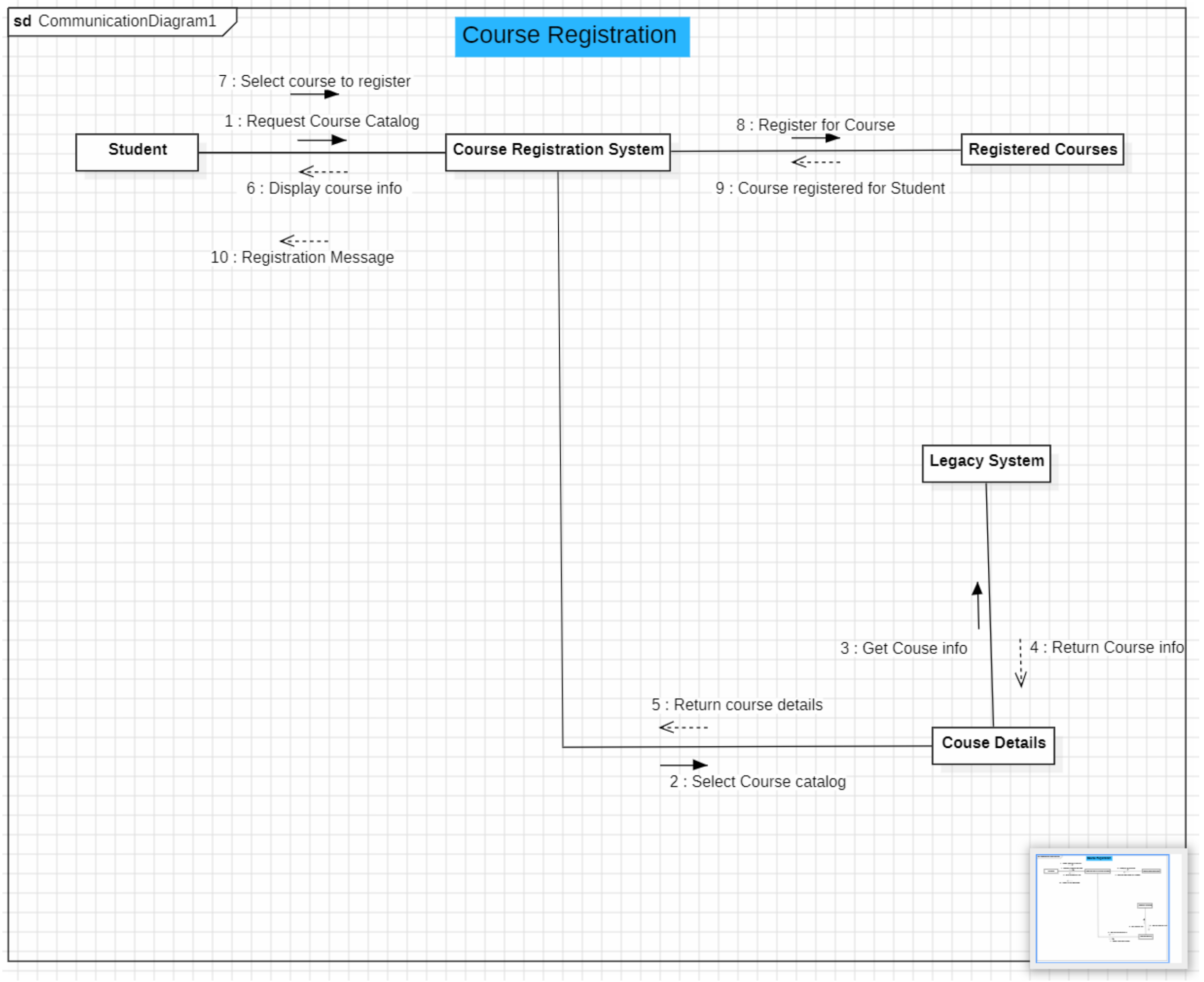


Result: The diagram includes Person, Patient, Staff, Doctor, Nurse, Admin_Staff and Hospital with relevant attributes and methods like Add_Person(), Add_Patient(), Modify_Hospital() and Add_Doctor(). It represents inheritance (eg: Staff subclasses) and operations (eg: Staff managing Patients), offering a structured view of hospital overflows.

2) Draw a communication diagram for a Course Registration System.

AIM: To model the interactions between entities in a course registration system using a UML communication diagram.

UML diagram:

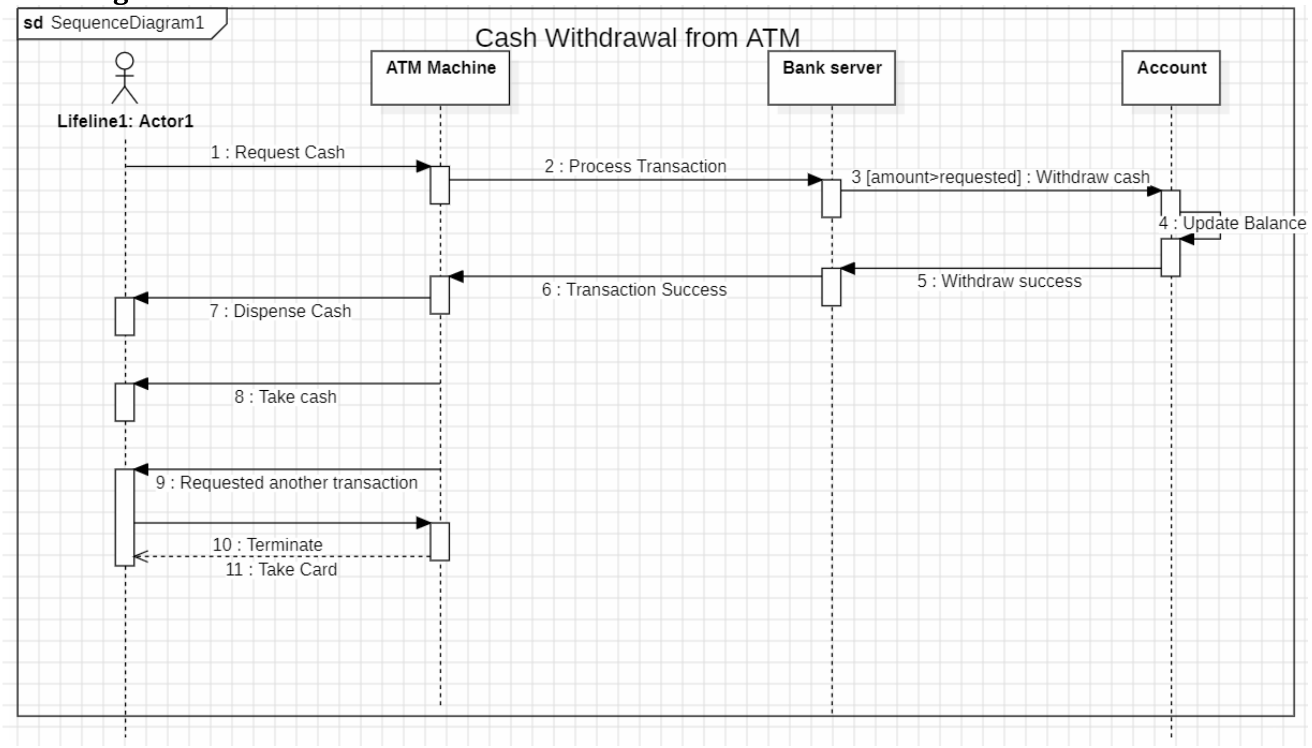

sd CommunicationDiagram1
Course Registration
7 : Select course to register
1 : Request Course Catalog
Student
Course Registration System
8 : Register for Course
Registered Courses
6 : Display course info
9 : Course registered for Student
10 : Registration Message
Legacy System
3 : Get Couse info
4 : Return Course info
5 : Return course details
Couse Details
2 : Select Course catalog

Result: The diagram shows interactions among Student, Course Registration System, Registered Courses, Legacy System and Course Details. The Student requests the Course Catalog, retrieved from Course Details via the Legacy System. The selected course is registered, and Registered Courses confirm enrollment, visually representing the course registration flow.

3) Draw a sequence diagram for ATM Cash withdrawal process.

AIM: To illustrate the interaction between the user, ATM machine, bank server and account during a cash withdrawal transaction using a UML sequence diagram.

UML diagram:

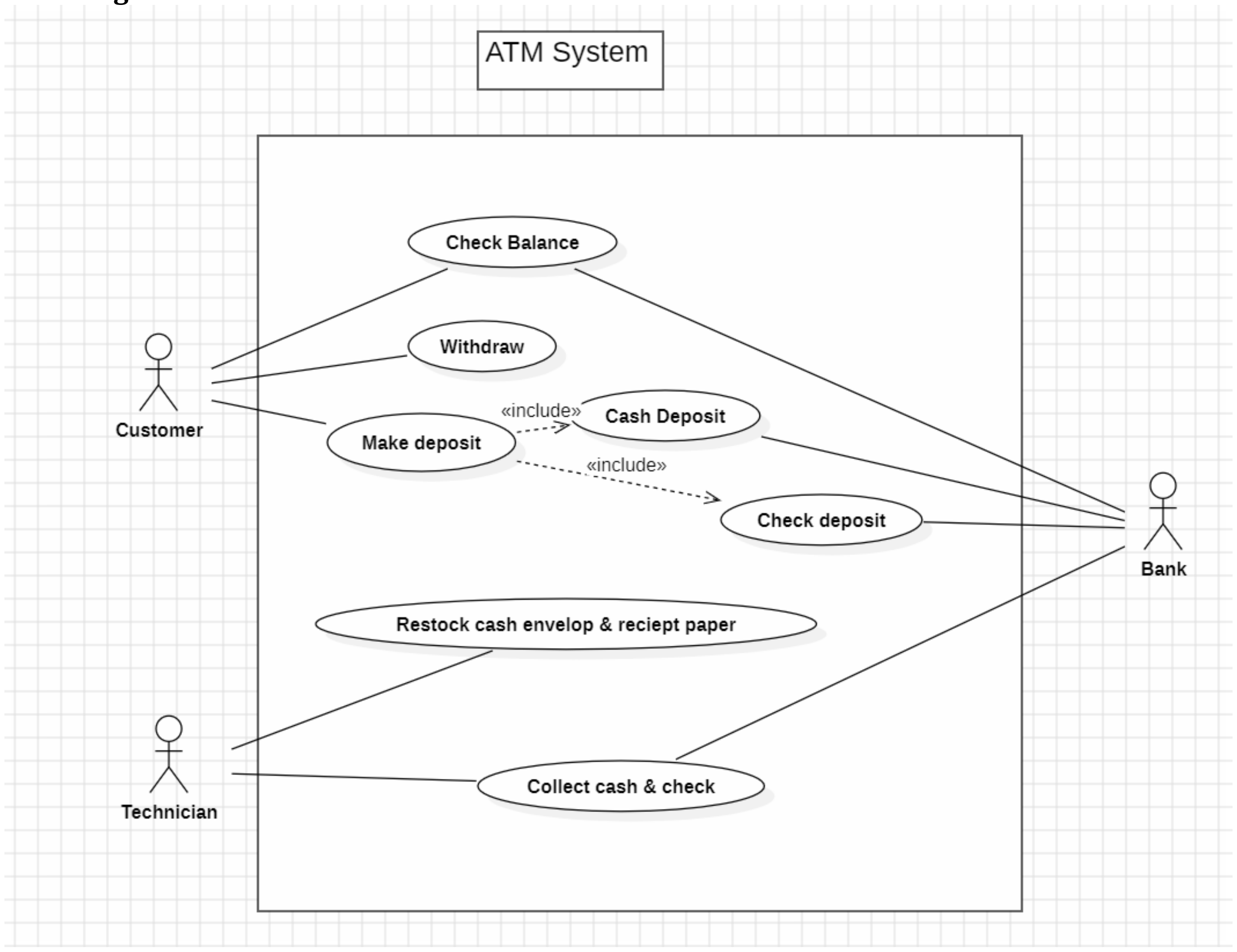


Result: The user requests cash from the ATM, which processes the transaction and communicates with the Bank Server. The Bank Server updates the Account Balance and confirms the transaction. The ATM then dispenses cash, and the session ends unless another transaction is initiated.

4) Draw an use case diagram for an ATM System, showcasing its functionalities and interactions.

AIM: To visually represent the actors and their interactions with the ATM system for a better understanding of its functionalities.

UML diagram:

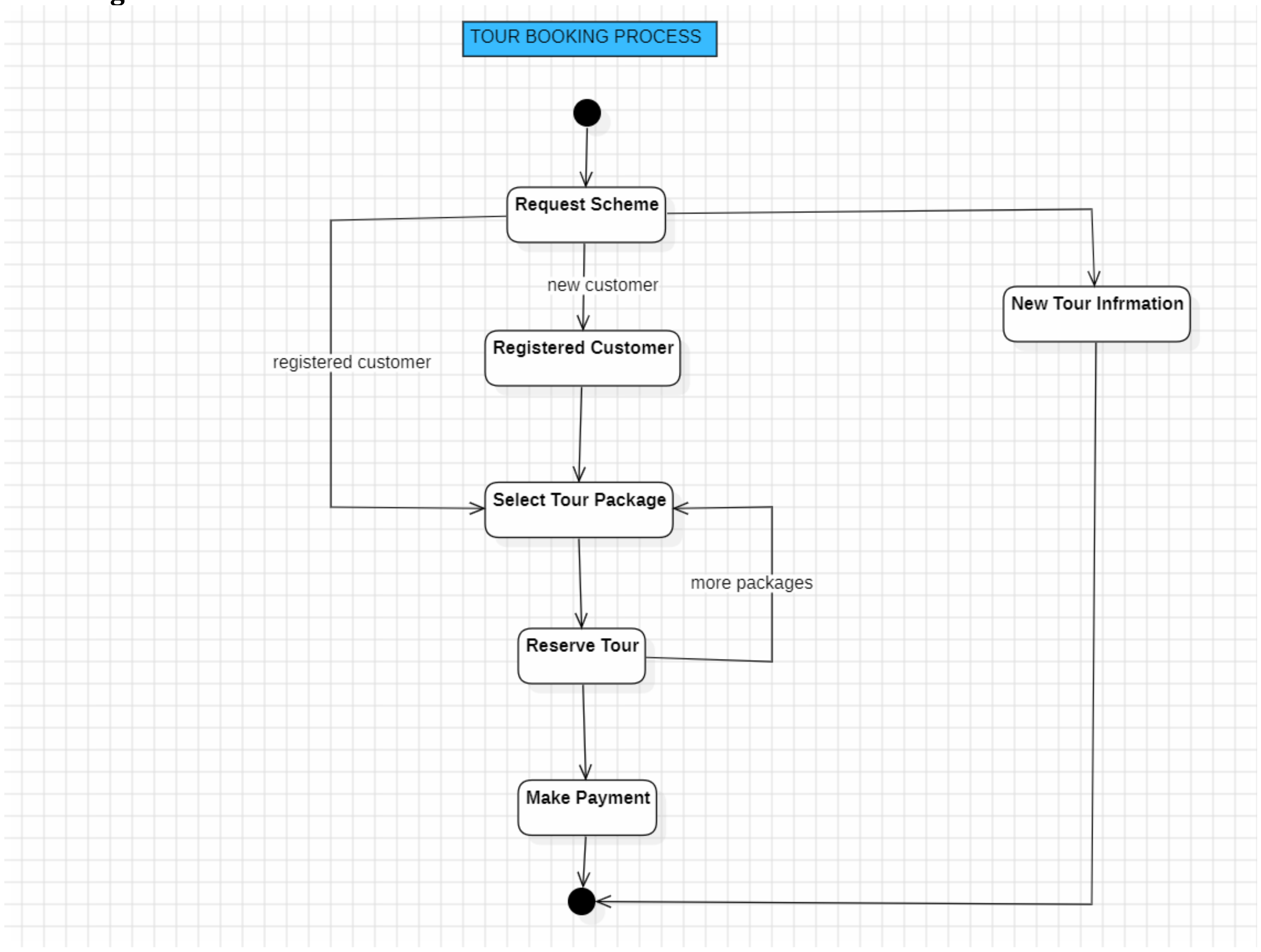


Result: The Use Case Diagram shows a Customer using the ATM for balance checks, withdrawals and deposits, a Technician handling maintenance, and the Bank supporting deposit operations.

5) Draw an state diagram for the tour package booking process.

AIM: To visually represent the booking process, highlighting the roles of new and registered customers and their required actions.

UML diagram:



Result: The flowchart starts with a request for a scheme, followed by registered customers selecting and reserving a tour package. New tour information is provided before reaching the payment step. Black circles mark the start and end points, ensuring a clear view of the booking process.

3 Basic Java Programs

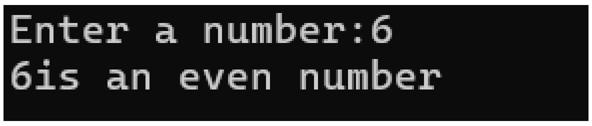
1) Write java program to check if a number is even or odd.

AIM: To write java program to check if a number is 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 an even number");
    }else{
      System.out.println(num + "is an odd number");
    }
    scanner.close();
  }
}
```

Output:



```
Enter a number:6
6is an even number
```

Result: The Java program successfully determines whether the given number is even or odd based on user input and displays the appropriate result.

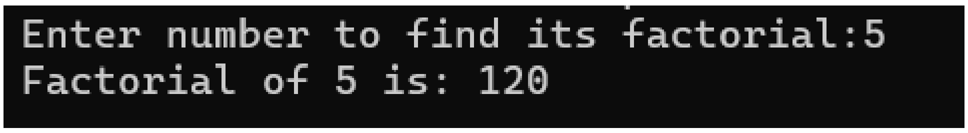
2) Write java program to find the factorial of a number.

AIM: To write java program to find the factorial of a number.

Code:

```
import java.util.Scanner;

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

    int fact = 1;
    int i = 1;

    while (i<=num){
      fact *= i;
      i++;
    }
    System.out.println("Factorial of " + num + " is: " + fact);
  }
}
```

Output:



```
Enter number to find its factorial:5
Factorial of 5 is: 120
```

Result: The Java program successfully calculates and displays the factorial of the given number based on user input.

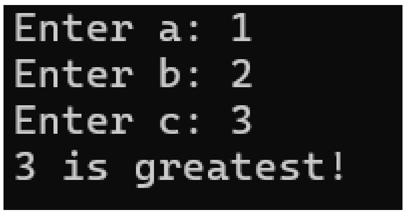
3) Write java program to find the Largest number.

AIM: To write java program to find the largest number.

Code:

```
import java.util.Scanner;

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

    if (a>b && a>c){
      System.out.println(a+" is greatest!");
    } else if (b>a && b>c){
      System.out.println(b+" is greatest!");
    }else if (c>a && c>b){
      System.out.println(c+" is greatest!");
    }else{
      System.out.println("Two or more elements are equal!");
    }
  }
}
```

Output:



```
Enter a: 1
Enter b: 2
Enter c: 3
3 is greatest!
```

Result: The Java program successfully determines and displays the largest number from the given input values.

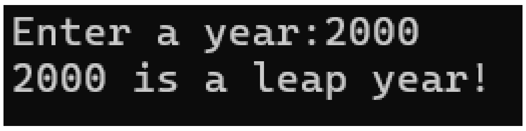
4) Write java program to find if the given year is a leap year or not.

AIM: To write java program to find if the given year is a leap year or not.

Code:

```
import java.util.Scanner;

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

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

Output:



```
Enter a year:2000
2000 is a leap year!
```

Result: The Java program successfully determines whether the given year is a leap year or not and displays the appropriate result

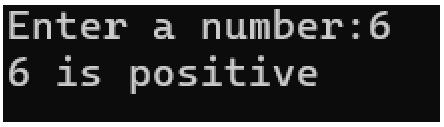
5) Write java program to find if the given number is positive or negative.

AIM: To write java program to find if the given number is positive or negative.

Code:

```
import java.util.Scanner;

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

    if(num>0){
      System.out.println(num + " is positive");
    }else if (num == 0){
      System.out.println(num + " is zero");
    }else{
      System.out.println(num + " is negative");
    }
  }
}
```

Output:



```
Enter a number:6
6 is positive
```

Result: The Java program successfully determines whether the given number is positive, negative, or zero and displays the appropriate result.

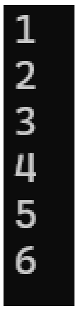
6) Write java program to print the elements of arrays.

AIM: To write java program to print the elements of arrays.

Code:

```
public class PrintingArrays
{
  public static void main(String[] args)
  {
    int[] numbers = {1, 2, 3, 4, 5, 6};

    for (int number: numbers)
    {
      System.out.println(number);
    }
  }
}
```

Output:



```
1
2
3
4
5
6
```

Result: The Java program successfully prints the elements of the given array based on user input.

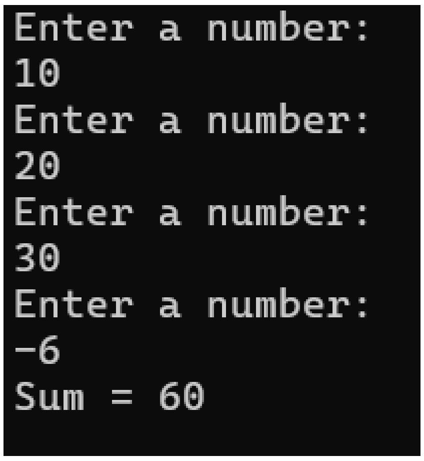
7) Write java program to print the sum of only positive number and not the negative numbers.

AIM: To write java program to print the sum of only the positive numbers and not the negative numbers.

Code:

```
//Sum of positive numbers
import java.util.Scanner;

class Sum
{
 public static void main(String[] args)
 {
  int sum = 0;

  Scanner input = new Scanner(System.in);
  System.out.println("Enter a number:");
  int number = input.nextInt();

  while (number>=0){
   sum += number;
   System.out.println("Enter a number:");
   number = input.nextInt();
  }

  System.out.println("Sum = " + sum);
  input.close();
 }
}
```

Output:



```
Enter a number:
10
Enter a number:
20
Enter a number:
30
Enter a number:
-6
Sum = 60
```

Result: The Java program successfully calculates and displays the sum of only the positive numbers while ignoring the negative numbers from the given input.

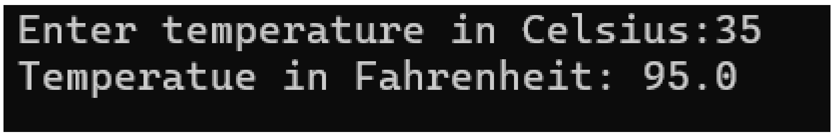
8) Write java program to calculate the temperature in Fahrenheit.

AIM: To write java program to calculate the temperature in Fahrenheit.

Code:

```
import java.util.Scanner;

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

   double f = (c * 9/5)+32;
   System.out.println("Temperatue in Fahrenheit: " + f);
  }
}
```

Output:



```
Enter temperature in Celsius:35
Temperatue in Fahrenheit: 95.0
```

Result: The Java program successfully converts the given temperature from Celsius to Fahrenheit and displays the result.

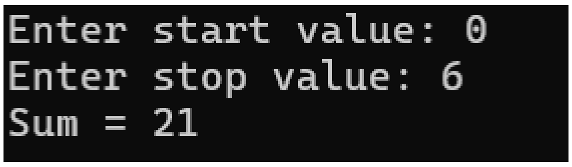
9) Write java program to print the sum of numbers by getting input from the user for the upper and lower limit.

AIM: To write java program to print the sum of numbers by getting input from the user for the upper and lower limit.

Code:

```
import java.util.Scanner;

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

    System.out.print("Enter stop value: ");
    int n = scanner.nextInt();


    for (int i = sum + 1; i<=n; ++i)
    {
      sum+=i;
    }
    System.out.println("Sum = " + sum);
  }
}
```

Output:



```
Enter start value: 0
Enter stop value: 6
Sum = 21
```

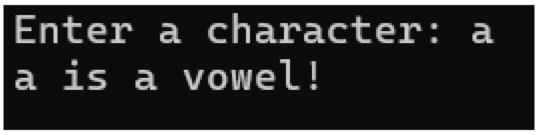
Result: The Java program successfully calculates and displays the sum of numbers within the given lower and upper limits based on user input.

10) Write java program to check if the given number is a vowel or a consonant.

AIM: To write java program to check if the given number is a vowel or a consonant.

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

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

    if (ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u' || ch == 'A' || ch == 'E' || ch == 'I' || ch == 'O' ||
ch == 'U'){
      System.out.println(ch + " is a vowel!");
    }else{
      System.out.println(ch + " is a consonant!");
    }
  }
}
```

Output:


```
Enter a character: a
a is a vowel!
```

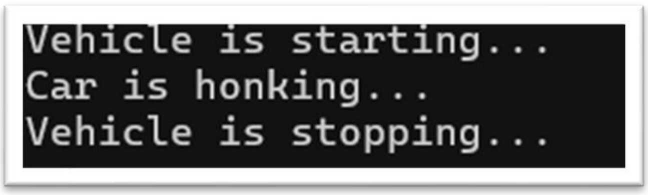
Result: The Java program successfully determines whether the given character is a vowel or a consonant and displays the appropriate result.

4 Single Inheritance Programs

a) **Code:**

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

        void stop(){
                System.out.println("Vehicle is stopping...");
        }
}

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

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

Output:

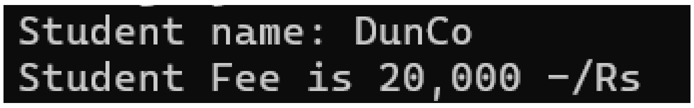


```
Vehicle is starting...
Car is honking...
Vehicle is stopping...
```

b) Code:

```
class Student {
   void Fee() {
      System.out.println("Student Fee is 20,000 -/Rs");
   }
}

class Student_Name extends Student {
   String name;

   Student_Name(String name) {
      this.name = name;
      System.out.println("Student name: " + name);
   }
}

class College {
   public static void main(String[] args) {
      Student_Name p = new Student_Name("DunCo");
      p.Fee();
   }
}
```

Output:



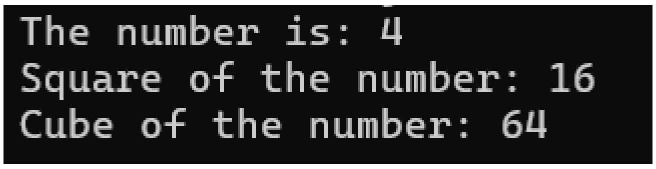
```
Student name: DunCo
Student Fee is 20,000 -/Rs
```

5 Multi-level Inheritance Programs

a) Code:

```
class Number{
        int value;

        void setValue(int value){
                this.value = value;
        }

        void displayValue(){
                System.out.println("The number is: " + value);
        }
}

class Square extends Number{
        void calculateSquare(){
                System.out.println("Square of the number: " + (value * value));
        }
}

class Cube extends Square{
        void calculateCube(){
                System.out.println("Cube of the number: " + (value * value * value));
        }
}

class Arithmetic{
        public static void main(String[] args){
                Cube number = new Cube();
                number.setValue(4);
                number.displayValue();
                number.calculateSquare();
                number.calculateCube();
        }
}
```

Output:

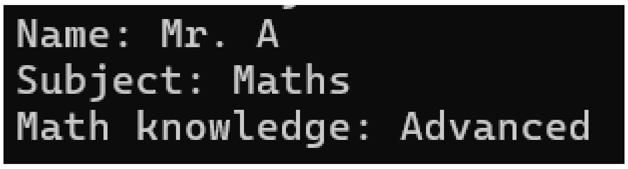


```
The number is: 4
Square of the number: 16
Cube of the number: 64
```

b) Code:

```
class Person{
        void name(String Name){
                System.out.println("Name: " + Name);
        }
}

class Teacher extends Person{
        void subject(String subject){
                System.out.println("Subject: " + subject);
        }
}

class MathTeacher extends Teacher{
        void mathKnowledge(String knowledge){
                System.out.println("Math knowledge: " + knowledge);
        }
}

class School{
        public static void main(String[] args){
                MathTeacher teacher = new MathTeacher();
                teacher.name("Mr. A");
                teacher.subject("Maths");
                teacher.mathKnowledge("Advanced");
        }
}
```

Output:



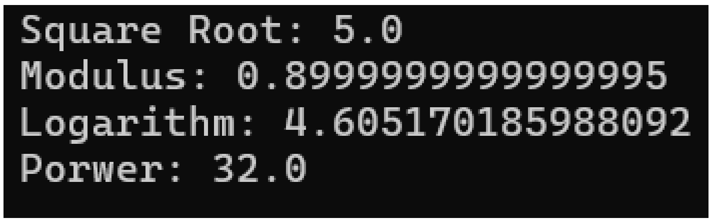
```
Name: Mr. A
Subject: Maths
Math knowledge: Advanced
```

6 Hierarchical Inheritance Programs

a) **Code:**

```
class Operations{
        void mod(double a, double b){
                System.out.println("Modulus: " + (a%b));
        }

        void power(double base, double exponent){
                System.out.println("Porwer: " + Math.pow(base, exponent));
        }
}

class SquareRoot extends Operations{
        void squareRoot(double a){
                System.out.println("Square Root: " + Math.sqrt(a));
        }
}

class Logarithm extends Operations {
        void logarithm(double a){
                if(a>0){
                        System.out.println("Logarithm: " + Math.log(a));
                }else{
                        System.out.println("Logarithm of non-positive numbers is undefined.");
                }
        }
}

public class Function{
        public static void main(String[] args){
                SquareRoot sqrt = new SquareRoot();
                Logarithm log = new Logarithm();

                sqrt.squareRoot(25);
                sqrt.mod(10.5, 3.2);

                log.logarithm(100);
                log.power(2, 5);
        }
}
```

Output:

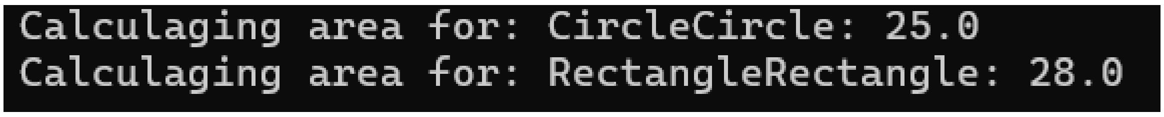


```
Square Root: 5.0
Modulus: 0.8999999999999995
Logarithm: 4.605170185988092
Porwer: 32.0
```

b) Code:

```
class Shape{
	void area(String shapeType){
		System.out.print("Calculaging area for: " + shapeType);
	}
}

class Circle extends Shape{
	void circleArea(double radius){
		System.out.println("Circle: " + (radius * radius));
	}
}

class Rectangle extends Shape{
	void rectangleArea(double length, double breadth){
		System.out.println("Rectangle: " + (length * breadth));
	}
}

class Geometry{
	public static void main(String[] args){
		Circle C = new Circle();
		C.area("Circle");
		C.circleArea(5);

		Rectangle R = new Rectangle();
		R.area("Rectangle");
		R.rectangleArea(4,7);
	}
}
```

Output:



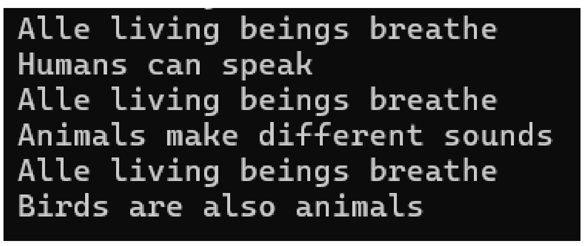
```
Calculaging area for: CircleCircle: 25.0
Calculaging area for: RectangleRectangle: 28.0
```

7 Multi-level Inheritance Programs

a) **Code:**

```
class LivingBeing{
	void breathe(){
		System.out.println("Alle living beings breathe");
	}
}

class Human extends LivingBeing{
	void speak(){
		System.out.println("Humans can speak");
	}
}

class Animal extends LivingBeing{
	void sound(){
		System.out.println("Animals make different sounds");
	}
}

class Bird extends Animal{
	void fly(){
		System.out.println("Birds are also animals");
	}
}

public class Creatures{
	public static void main(String[] args){
		Human h = new Human();
		Animal a = new Animal();
		Bird b = new Bird();

		h.breathe();
		h.speak();

		a.breathe();
		a.sound();

		b.breathe();
		b.fly();
	}
}
```

Output:

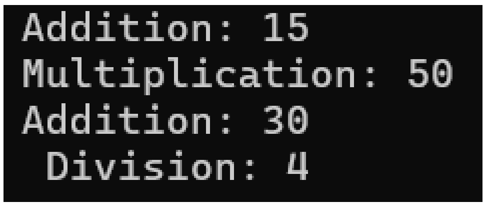


```
Alle living beings breathe
Humans can speak
Alle living beings breathe
Animals make different sounds
Alle living beings breathe
Birds are also animals
```

b) Code:

```
class Arithmetic{
        void add(int a, int b){
                System.out.println("Addition: " + (a+b));
        }
}

class Multiply extends Arithmetic{
        void Multiply(int a, int b){
                System.out.println("Multiplication: "+ (a*b));
        }
}

class Divide extends Arithmetic{
        void divide(int a, int b){
                if(b!=0){
                        System.out.println(" Division: " + (a/b));
                }else{
                        System.out.println("Division by zero is not allowed");
                }
        }
}

public class Main{
        public static void main(String[] args){
                Multiply m = new Multiply();
                Divide d = new Divide();

                m.add(10, 5);
                m.Multiply(10, 5);

                d.add(20, 10);
                d.divide(20, 5);
        }
}
```

Output:



```
Addition: 15
Multiplication: 50
Addition: 30
 Division: 4
```

8 Constructor Programs

a) **Code:**

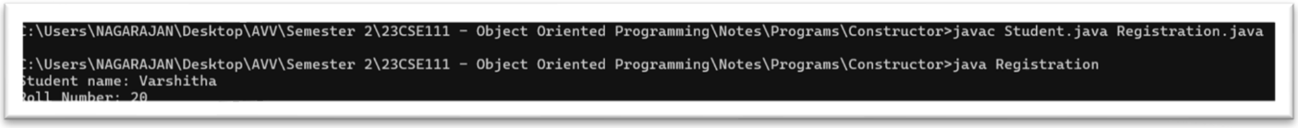
Student.java:

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

    Student(int N) {
        name = "Varshitha";
        num = N;
    }

    void displayDetails() {
        System.out.println("Student name: " + name + "\nRoll Number: " + num);
    }
}
```

Registration.java:

```
public class Registration {
    public static void main(String[] args) {
        Student s = new Student(20);
        s.displayDetails();
    }
}
```

Output:



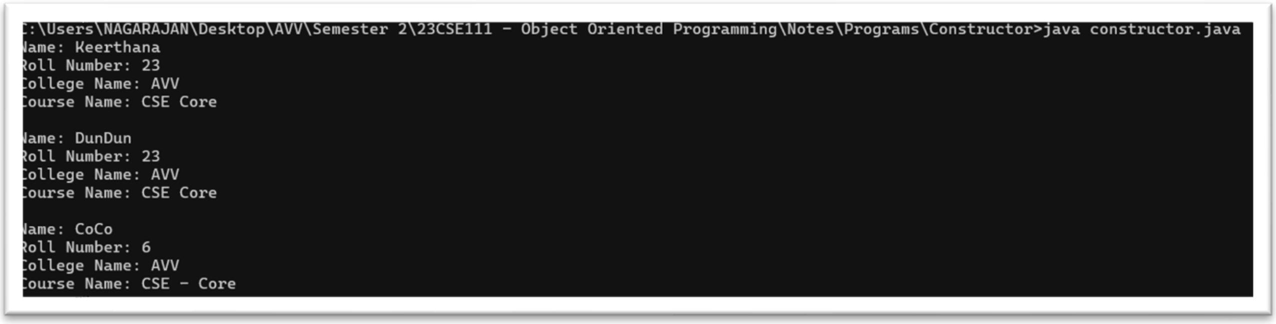
```
:\Users\NAGARAJAN\Desktop\AVV\Semester 2\23CSE111 - Object Oriented Programming\Notes\Programs\Constructor>javac Student.java Registration.java

:\Users\NAGARAJAN\Desktop\AVV\Semester 2\23CSE111 - Object Oriented Programming\Notes\Programs\Constructor>java Registration
tudent name: Varshitha
oll Number: 20
```

9 Constructor Overloading Programs

a) Code:

```
class Information{
        String Name;
        int RollNum;
        String Clg;
        String Course;
        Information(String Name){
                this.Name = "Keerthana";
                this.RollNum = 23;
                this.Clg = "AVV";
                this.Course = "CSE Core";
        }
        Information(String Name, String Clg){
                this.Name = Name;
                this.RollNum = 23;
                this.Clg = Clg;
                this.Course = "CSE Core";
        }
        Information(String Name, String Clg, int RollNum, String Course){
                this.Name = Name;
                this.RollNum = RollNum;
                this.Clg = Clg;
                this.Course = Course;
        }

        void Display(){
                System.out.println("Name: "+Name);
                System.out.println("Roll Number: "+RollNum);
                System.out.println("College Name: " + Clg);
                System.out.println("Course Name: "+Course +"\n");
        }
}

public class constructor{
        public static void main(String[] args){
                Information s1 = new Information("Varapradha");
                Information s2 = new Information("DunDun", "AVV");
                Information s3 = new Information("CoCo", "AVV", 006, "CSE - Core");

                s1.Display();
                s2.Display();
                s3.Display();
        }
}
```

Output:



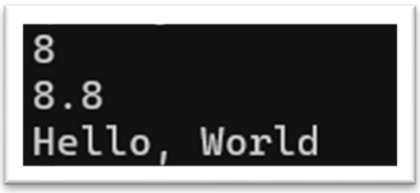
```
:\Users\NAGARAJAN\Desktop\AVV\Semester 2\23CSE111 - Object Oriented Programming\Notes\Programs\Constructor>java constructor.java
lame: Keerthana
Roll Number: 23
College Name: AVV
Course Name: CSE Core

lame: DunDun
Roll Number: 23
College Name: AVV
Course Name: CSE Core

lame: CoCo
Roll Number: 6
College Name: AVV
Course Name: CSE - Core
```

10 Method Overloading Programs

a) **Code:**

```
class Calculator{
	int add(int a, int b){
		return a+b;
	}
	double add(double a, double b){
		return a+b;
	}
	String add(String a, String b){
		return a+b;
	}
}

public class Main{
	public static void main(String[] args){
		Calculator calc = new Calculator();
		System.out.println(calc.add(5,3));
		System.out.println(calc.add(5.5, 3.3));
		System.out.println(calc.add("Hello, ", "World"));
	}
}
```

Output:

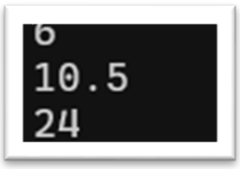


```
8
8.8
Hello, World
```

b) Code:

```
class Multiply{
        int multiply(int a, int b){
                return a*b;
        }
        double multiply(double a, double b){
                return a*b;
        }
        int multiply(int a, int b, int c){
                return a*b*c;
        }
}
public class Main1{
        public static void main(String[] args){
                Multiply m = new Multiply();
                System.out.println(m.multiply(2,3));
                System.out.println(m.multiply(2.5,4.2));
                System.out.println(m.multiply(2,3,4));
        }
}
```

Output:



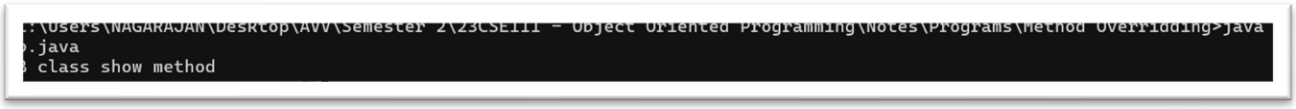
```
6
10.5
24
```

11 Method Over-ridding Programs

a) **Code:**

```
class Base{
        void show(){
                System.out.println("Base class show method.");
        }
}

class B extends Base{
        void show(){
                System.out.println("B class show method");
        }
}

public class b{
        public static void main(String[] args){
                Base b1 = new B();
                b1.show();
        }
}
```

Output:

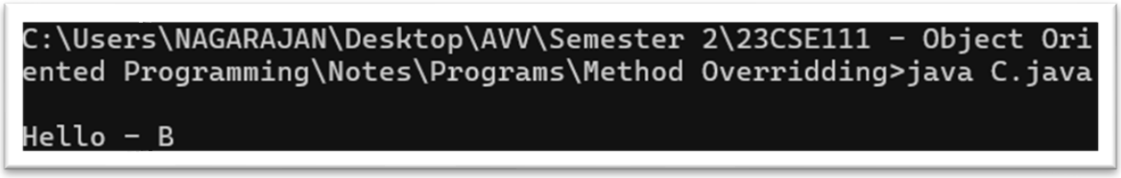


```
:\Users\NAGARAJAN\Desktop\AVV\Semester 2\23CSE111 - Object Oriented Programming\Notes\Programs\Method Overridding>java
.java
 class show method
```

b) Code:

```
class A{
	void hello(){
		System.out.println("Hello - A");
	}
}

class B extends A{
	void hello(){
		System.out.println("Hello - B");
	}
}

public class C{
	public static void main(String[] args){
		A c = new B();
		c.hello();
	}
}
```

Output:



```
C:\Users\NAGARAJAN\Desktop\AVV\Semester 2\23CSE111 - Object Ori
ented Programming\Notes\Programs\Method Overridding>java C.java

Hello - B
```

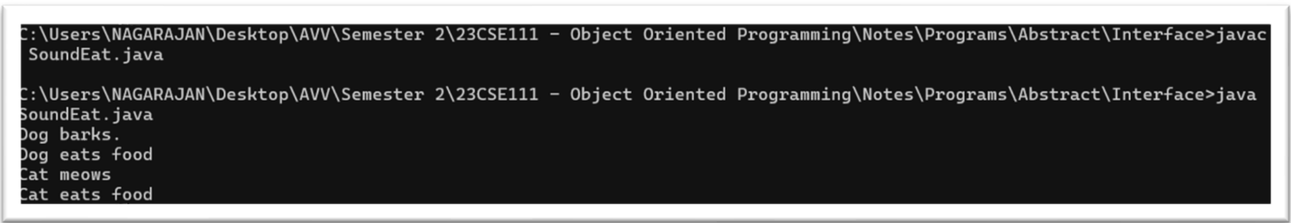
ABSTRACT

12 Interface Programs

a) **Code:**

```
interface Animal{
	void sound();
	void eat();
}

class Dog implements Animal{
	public void sound(){
		System.out.println("Dog barks.");
	}
	public void eat(){
		System.out.println("Dog eats food");
	}
}

class Cat implements Animal{
	public void sound(){
		System.out.println("Cat meows");
	}
	public void eat(){
		System.out.println("Cat eats food");
	}
}

public class SoundEat{
	public static void main(String[] args){
		Animal dog = new Dog();
		dog.sound();
		dog.eat();

		Animal cat = new Cat();
		cat.sound();
		cat.eat();
	}
}
```

Output:

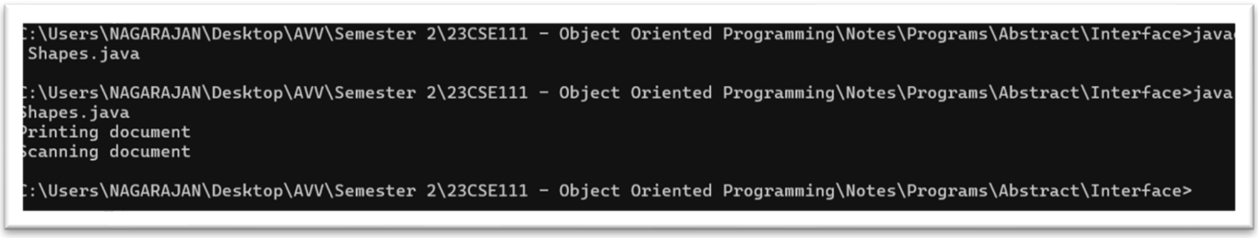


```
C:\Users\NAGARAJAN\Desktop\AVV\Semester 2\23CSE111 - Object Oriented Programming\Notes\Programs\Abstract\Interface>javac
 SoundEat.java

C:\Users\NAGARAJAN\Desktop\AVV\Semester 2\23CSE111 - Object Oriented Programming\Notes\Programs\Abstract\Interface>java
SoundEat.java
Dog barks.
Dog eats food
Cat meows
Cat eats food
```

b) Code:

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

interface Scanner{
        void scan();
}

class APrinter implements Printer, Scanner{
        public void print(){
                System.out.println("Printing document");
        }
        public void scan(){
                System.out.println("Scanning document");
        }
}

public class Shapes{
        public static void main(String[] args){
                APrinter p = new APrinter();
                p.print();
                p.scan();
        }
}
```

Output:

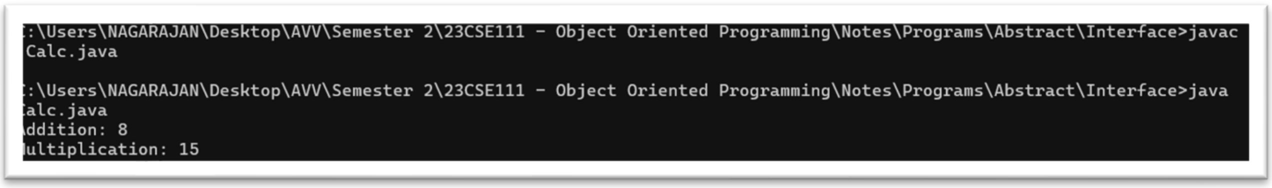


```
:\Users\NAGARAJAN\Desktop\AVV\Semester 2\23CSE111 - Object Oriented Programming\Notes\Programs\Abstract\Interface>java
 Shapes.java

:\Users\NAGARAJAN\Desktop\AVV\Semester 2\23CSE111 - Object Oriented Programming\Notes\Programs\Abstract\Interface>java
hapes.java
rinting document
canning document

:\Users\NAGARAJAN\Desktop\AVV\Semester 2\23CSE111 - Object Oriented Programming\Notes\Programs\Abstract\Interface>
```

c) Code:

```
interface Calculator{
	int add(int a, int b);
	default int multiply(int a, int b){
		return a*b;
	}
}

class SimpleCalc implements Calculator{
	public int add(int a, int b){
		return a+b;
	}
}

public class Calc{
	public static void main(String[] args){
		Calculator c = new SimpleCalc();
		System.out.println("Addition: " + c.add(5, 3));
		System.out.println("Multiplication: " + c.multiply(5,3));
	}
}
```

Output:

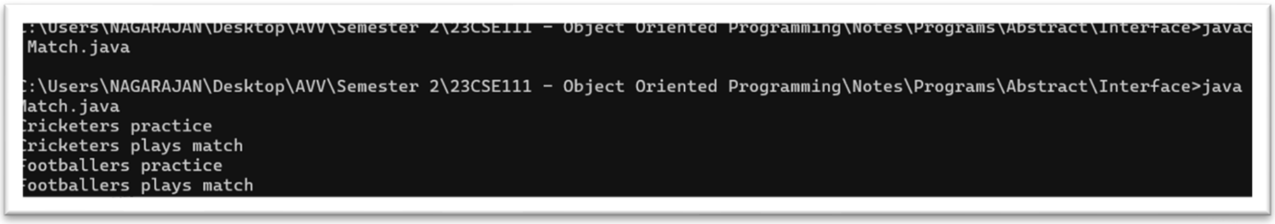


```
:\Users\NAGARAJAN\Desktop\AVV\Semester 2\23CSE111 - Object Oriented Programming\Notes\Programs\Abstract\Interface>javac
Calc.java

:\Users\NAGARAJAN\Desktop\AVV\Semester 2\23CSE111 - Object Oriented Programming\Notes\Programs\Abstract\Interface>java
alc.java
ddition: 8
ultiplication: 15
```

d) Code:

```
interface Player{
	void practice();
	void playMatch();
}

class Cricketer implements Player{
	public void practice(){
		System.out.println("Cricketers practice");
	}
	public void playMatch(){
		System.out.println("Cricketers plays match");
	}
}

class FootBaller implements Player{
	public void practice(){
		System.out.println("Footballers practice");
	}
	public void playMatch(){
		System.out.println("Footballers plays match");
	}
}

public class Match{
	public static void main(String[] args){
		Player p1 = new Cricketer();
		p1.practice();
		p1.playMatch();

		Player p2 = new FootBaller();
		p2.practice();
		p2.playMatch();
	}
}
```

Output:



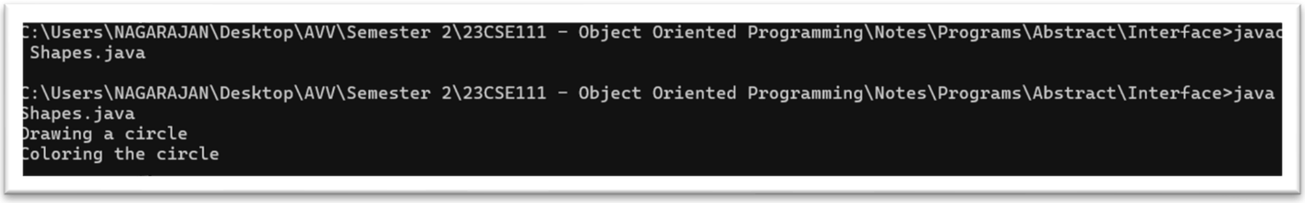
```
:\Users\NAGARAJAN\Desktop\AVV\Semester 2\23CSE111 - Object Oriented Programming\Notes\Programs\Abstract\Interface>javac
 Match.java

:\Users\NAGARAJAN\Desktop\AVV\Semester 2\23CSE111 - Object Oriented Programming\Notes\Programs\Abstract\Interface>java
latch.java
ricketers practice
ricketers plays match
ootballers practice
ootballers plays match
```

13 Interface Programs

a) **Code:**

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

interface Colorable{
	void color();
}

class Circle extends Shape implements Colorable{
	void draw(){
		System.out.println("Drawing a circle");
	}
	public void color(){
		System.out.println("Coloring the circle");
	}
}

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

Output:

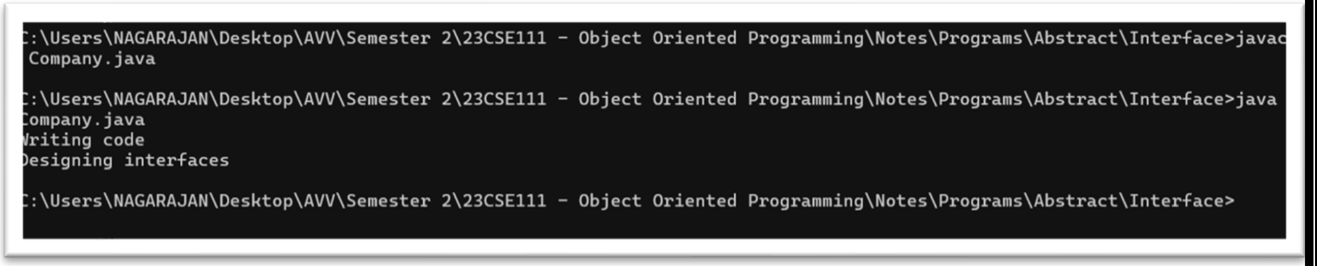


```
:\Users\NAGARAJAN\Desktop\AVV\Semester 2\23CSE111 - Object Oriented Programming\Notes\Programs\Abstract\Interface>javac
 Shapes.java

:\Users\NAGARAJAN\Desktop\AVV\Semester 2\23CSE111 - Object Oriented Programming\Notes\Programs\Abstract\Interface>java
hapes.java
rawing a circle
oloring the circle
```

b) Code:

```
abstract class Employee{
        abstract void work();
}

class Developer extends Employee{
        void work(){
                System.out.println("Writing code");
        }
}

class Designer extends Employee{
        void work(){
                System.out.println("Designing interfaces");
        }
}

public class Company{
        public static void main(String[] args){
                Employee e1 = new Developer();
                Employee e2 = new Designer();
                e1.work();
                e2.work();
        }
}
```

Output:

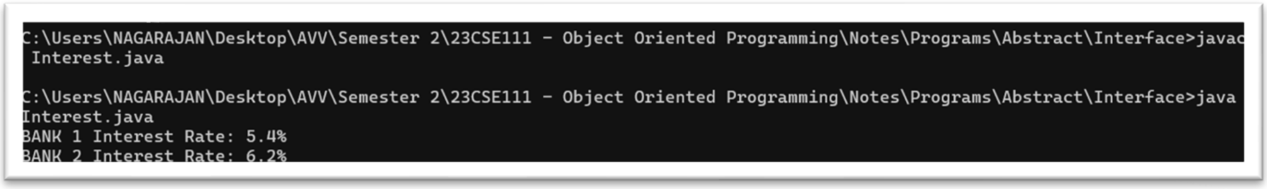


```
:\Users\NAGARAJAN\Desktop\AVV\Semester 2\23CSE111 - Object Oriented Programming\Notes\Programs\Abstract\Interface>javac
 Company.java

:\Users\NAGARAJAN\Desktop\AVV\Semester 2\23CSE111 - Object Oriented Programming\Notes\Programs\Abstract\Interface>java
ompany.java
riting code
esigning interfaces

:\Users\NAGARAJAN\Desktop\AVV\Semester 2\23CSE111 - Object Oriented Programming\Notes\Programs\Abstract\Interface>
```

c) Code:

```
abstract class Bank{
        String bankName;
        Bank(String bankName){
                this.bankName = bankName;
        }
        abstract void IR();
}

class bank1 extends Bank{
        bank1(){
                super("BANK 1");
        }

        void IR(){
                System.out.println(bankName + " Interest Rate: 5.4%");
        }
}

class bank2 extends Bank{
        bank2(){
                super("BANK 2");
        }

        void IR(){
                System.out.println(bankName + " Interest Rate: 6.2%");
        }
}


public class Interest{
        public static void main(String[] args){
                Bank b1 = new bank1();
                Bank b2 = new bank2();
                b1.IR();
                b2.IR();
        }
}
```

Output:

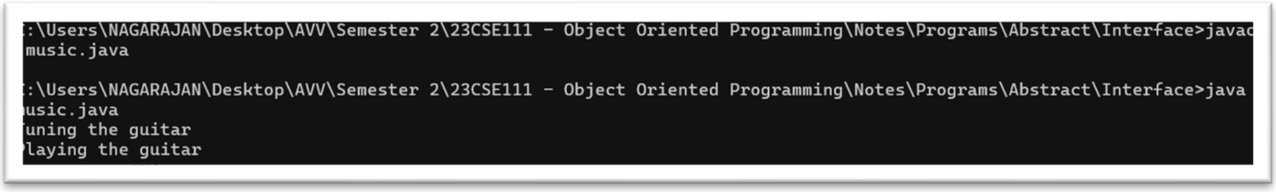


```
C:\Users\NAGARAJAN\Desktop\AVV\Semester 2\23CSE111 - Object Oriented Programming\Notes\Programs\Abstract\Interface>javac
 Interest.java

C:\Users\NAGARAJAN\Desktop\AVV\Semester 2\23CSE111 - Object Oriented Programming\Notes\Programs\Abstract\Interface>java
Interest.java
BANK 1 Interest Rate: 5.4%
BANK 2 Interest Rate: 6.2%
```

d) Code:

```
interface Playable {
   void play();
}

abstract class Instrument {
   abstract void tune();
}

class Guitar extends Instrument implements Playable {
   void tune() {
      System.out.println("Tuning the guitar");
   }

   public void play() {
      System.out.println("Playing the guitar");
   }
}

public class music {
   public static void main(String[] args) {
      Guitar g = new Guitar();
      g.tune();
      g.play();
   }
}
```

Output:



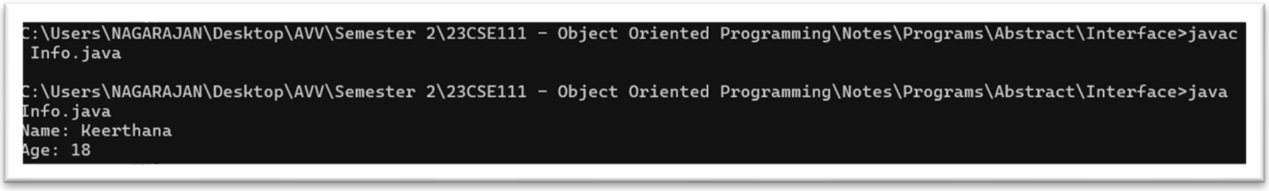
```
:\Users\NAGARAJAN\Desktop\AVV\Semester 2\23CSE111 - Object Oriented Programming\Notes\Programs\Abstract\Interface>javac
music.java

:\Users\NAGARAJAN\Desktop\AVV\Semester 2\23CSE111 - Object Oriented Programming\Notes\Programs\Abstract\Interface>java
usic.java
uning the guitar
laying the guitar
```

14 Encapsulation Programs

a) **Code:**

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

        public String getName(){
                return name;
        }

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

        public int getAge(){
                return age;
        }

        public void setAge(int age){
                this.age = age;
        }
}

public class Info{
        public static void main(String[] args){
                Student s = new Student();
                s.setName("Keerthana");
                s.setAge(18);
                System.out.println("Name: " + s.getName());
                System.out.println("Age: " + s.getAge());
        }
}
```

Output:

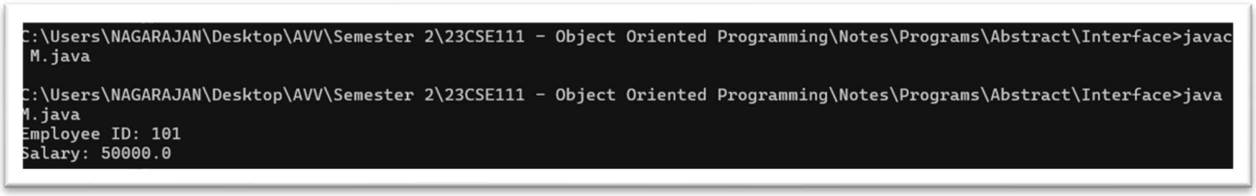


```
C:\Users\NAGARAJAN\Desktop\AVV\Semester 2\23CSE111 - Object Oriented Programming\Notes\Programs\Abstract\Interface>javac
 Info.java

C:\Users\NAGARAJAN\Desktop\AVV\Semester 2\23CSE111 - Object Oriented Programming\Notes\Programs\Abstract\Interface>java
Info.java
Name: Keerthana
Age: 18
```

b) Code:

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

        public int getId(){
                return id;
        }

        public void setId(int id){
                this.id=id;
        }

        public double getSalary(){
                return salary;
        }

        public void setSalary(double salary){
                this.salary = salary;
        }
}

public class M{
        public static void main(String[] args){
                Employee emp = new Employee();
                emp.setId(101);
                emp.setSalary(50000.0);
                System.out.println("Employee ID: " + emp.getId());
                System.out.println("Salary: " + emp.getSalary());
        }
}
```

Output:

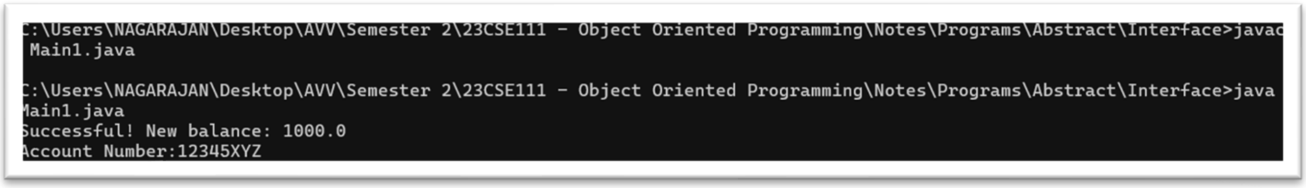


```
:\Users\NAGARAJAN\Desktop\AVV\Semester 2\23CSE111 - Object Oriented Programming\Notes\Programs\Abstract\Interface>javac
 M.java

:\Users\NAGARAJAN\Desktop\AVV\Semester 2\23CSE111 - Object Oriented Programming\Notes\Programs\Abstract\Interface>java
M.java
mployee ID: 101
alary: 50000.0
```

c) Code:

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

        public String getNum(){
                return Num;
        }

        public void setNum(String Num){
                this.Num = Num;
        }

        public double getBalance(){
                return balance;
        }

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

public class Main1{
        public static void main(String[] args){
                BankAccount a = new BankAccount();
                a.setNum("12345XYZ");
                a.deposit(1000);
                System.out.println("Account Number:" + a.getNum());
        }
}
```

Output:

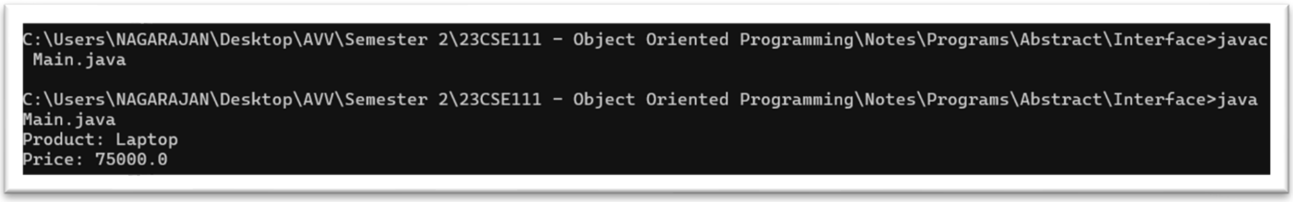


```
:\Users\NAGARAJAN\Desktop\AVV\Semester 2\23CSE111 - Object Oriented Programming\Notes\Programs\Abstract\Interface>javac
 Main1.java

:\Users\NAGARAJAN\Desktop\AVV\Semester 2\23CSE111 - Object Oriented Programming\Notes\Programs\Abstract\Interface>java
Main1.java
uccessful! New balance: 1000.0
ccount Number:12345XYZ
```

d) Code:

```
class Product {
  private String productName;
  private double price;

  public String getProductName() {
    return productName;
  }

  public void setProductName(String productName) {
    this.productName = productName;
  }

  public double getPrice() {
    return price;
  }

  public void setPrice(double price) {
    if (price > 0) {
      this.price = price;
    } else {
      System.out.println("Price cannot be negative.");
    }
  }
}

public class Main {
  public static void main(String[] args) {
    Product product = new Product();
    product.setProductName("Laptop");
    product.setPrice(75000);
    System.out.println("Product: " + product.getProductName());
    System.out.println("Price: " + product.getPrice());
  }
}
```

Output:



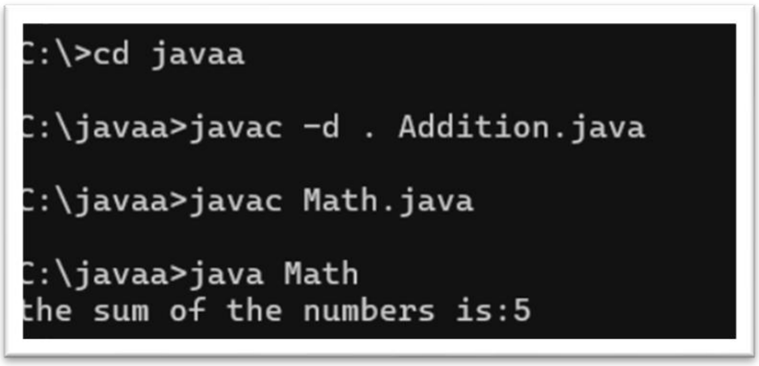
```
C:\Users\NAGARAJAN\Desktop\AVV\Semester 2\23CSE111 - Object Oriented Programming\Notes\Programs\Abstract\Interface>javac
 Main.java

C:\Users\NAGARAJAN\Desktop\AVV\Semester 2\23CSE111 - Object Oriented Programming\Notes\Programs\Abstract\Interface>java
Main.java
Product: Laptop
Price: 75000.0
```

15 Packages Programs

a) Code:

```
 package mathoperations;
public class Addition{ public int add(int n1,int n2){
return n1+n2;
}
 }
Main Code:
import mathoperations.Addition;
 public class Math
{
public static void main(String[] args)
{ Addition a=new Addition();
int sum=a.add(2,3);
System.out.println("the sum of the numbers is:"+sum);
}
 }
```

Output:



```
:\>cd javaa

:\javaa>javac -d . Addition.java

:\javaa>javac Math.java

:\javaa>java Math
the sum of the numbers is:5
```

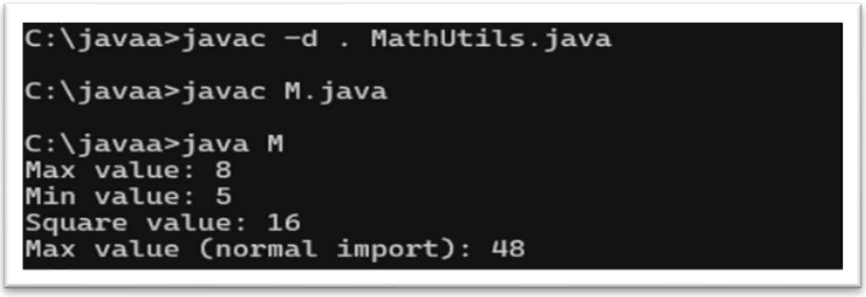
b) Code:

```
package utilities;
public class MathUtils {
public static int max(int a, int b) {
return (a > b) ? a : b;
}
public static int min(int a, int b) {
return (a < b) ? a : b;
}
public static int square(int a) {
return a * a;
}
}
```

Main code:

```
import utilities.MathUtils;
import static utilities.MathUtils.max;
import static utilities.MathUtils.min;
import static utilities.MathUtils.square;
public class M {
public static void main(String[] args) {
int maxValue = max(5, 8);
int minValue = min(5, 8);
int squareValue = square(4);
System.out.println("Max value: " + maxValue);
System.out.println("Min value: " + minValue);
System.out.println("Square value: " + squareValue);
int maxValueNormalImport = MathUtils.max(23, 48);
System.out.println("Max value (normal import): " + maxValueNormalImport);
}
}
```

Output:



```
C:\javaa>javac -d . MathUtils.java

C:\javaa>javac M.java

C:\javaa>java M
Max value: 8
Min value: 5
Square value: 16
Max value (normal import): 48
```

c) Code:

```
Package company.hr;
public class Employee {
private String name;
private double baseSalary;
private int hoursWorked;
public Employee(String name, double baseSalary, int hoursWorked) {
this.name = name;
this.baseSalary = baseSalary;
this.hoursWorked = hoursWorked;
}
public String getName() {
return name;
}
public double getBaseSalary() {
return baseSalary;
}
public int getHoursWorked() {
return hoursWorked;
}
}
```

Package code:

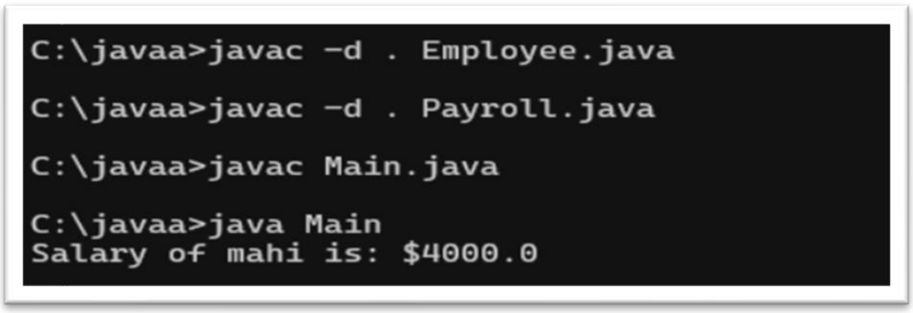
```
package company.finance;
import company.hr.Employee;


public class Payroll {
public double calculateSalary(Employee employee) {
double baseSalary = employee.getBaseSalary();
int hoursWorked = employee.getHoursWorked();
double hourlyRate = baseSalary / 160; // Assuming 160 working hours in
a month
return hourlyRate * hoursWorked;
}
}
```

Main Code:

```
import company.hr.Employee;
import company.finance.Payroll;
public class Main {
public static void main(String[] args) {
```

```
Employee employee = new Employee("mahi", 4000.0, 160);
Payroll payroll = new Payroll();
double salary = payroll.calculateSalary(employee);
System.out.println("Salary of " + employee.getName() + " is: $" +
salary);}
}
```

Output:



```
C:\javaa>javac -d . Employee.java

C:\javaa>javac -d . Payroll.java

C:\javaa>javac Main.java

C:\javaa>java Main
Salary of mahi is: $4000.0
```

d) Code:

package code:

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

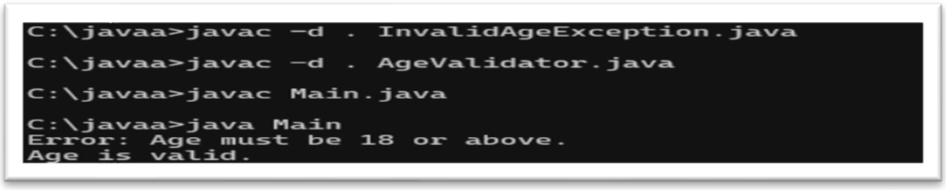
Package code:

```
package validation;
import exceptions.InvalidAgeException;

public class AgeValidator {
 public void validateAge(int age) throws InvalidAgeException {
 if (age < 18) {
 throw new InvalidAgeException("Age must be 18 or above.");
 }
 System.out.println("Age is valid.");
 }
}
```

Main code:

```
import validation.AgeValidator;
import exceptions.InvalidAgeException;
public class Main {
 public static void main(String[] args) {
 AgeValidator validator = new AgeValidator();
 try {
 validator.validateAge(15); // This will throw an exception
 } catch (InvalidAgeException e) {
 System.out.println("Error: " + e.getMessage());
} try { validator.validateAge(20);
} catch (InvalidAgeException e) { System.out.println("Error: " + e.getMessage());
}
}
```

Output:



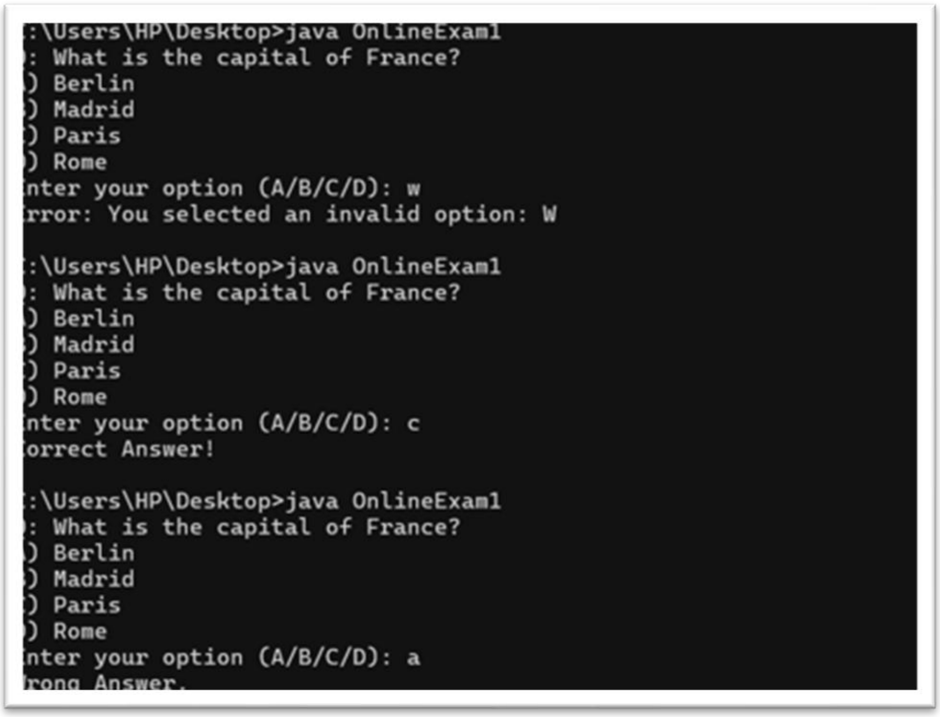
```
C:\javaa>javac -d . InvalidAgeException.java

C:\javaa>javac -d . AgeValidator.java

C:\javaa>javac Main.java

C:\javaa>java Main
Error: Age must be 18 or above.
Age is valid.
```

16 Exception Handling Programs

a) Code:

```
 import java.util.Scanner;

        class InvalidOptionException extends Exception   {
   public InvalidOptionException(String message) {
     super(message);
    }
}
public class OnlineExam1 {
   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);
      String[] options = {"A", "B", "C", "D"};
      try {
         System.out.println("Q: What is the capital of France?");
         System.out.println("A) Berlin\nB) Madrid\nC) Paris\nD) Rome");
         System.out.print("Enter your option (A/B/C/D): ");
         String input = scanner.next().toUpperCase();
                     boolean isValid = false;
         for (String opt : options) {
            if (opt.equals(input)) {
               isValid = true;
               break;
            }
         }

         if (!isValid) {
            throw new InvalidOptionException("You selected an invalid option: " + input);
         }
         if (input.equals("C")) {
            System.out.println("Correct Answer!");
         } else {
            System.out.println("Wrong Answer.");
         }

      } catch (InvalidOptionException e) {
         System.out.println("Error: " + e.getMessage());
      } finally {
         scanner.close();
      } }
   }
```

Output:

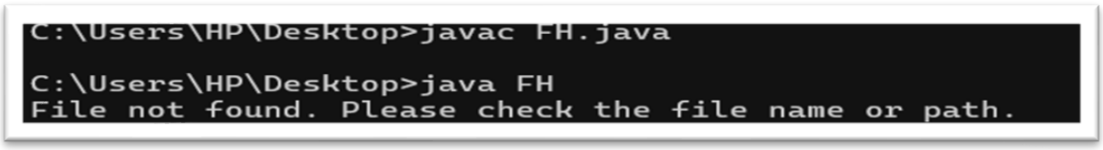


```
:\Users\HP\Desktop>java OnlineExam1
): What is the capital of France?
) Berlin
) Madrid
) Paris
) Rome
nter your option (A/B/C/D): w
rror: You selected an invalid option: W

:\Users\HP\Desktop>java OnlineExam1
): What is the capital of France?
) Berlin
) Madrid
) Paris
) Rome
nter your option (A/B/C/D): c
orrect Answer!

:\Users\HP\Desktop>java OnlineExam1
): What is the capital of France?
) Berlin
) Madrid
) Paris
) Rome
nter your option (A/B/C/D): a
rong Answer.
```

b) Code:

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

public class FH {
  public static void main(String[] args) {
    try {
      File file = new File("FH.java");
      Scanner scanner = new Scanner(file); // may throw FileNotFoundException
      while (scanner.hasNextLine()) {
        System.out.println(scanner.nextLine());
      }
      scanner.close();
    } catch (FileNotFoundException e) {
      System.out.println("File not found. Please check the file name or path.");
    }
  }
}
```

Output:

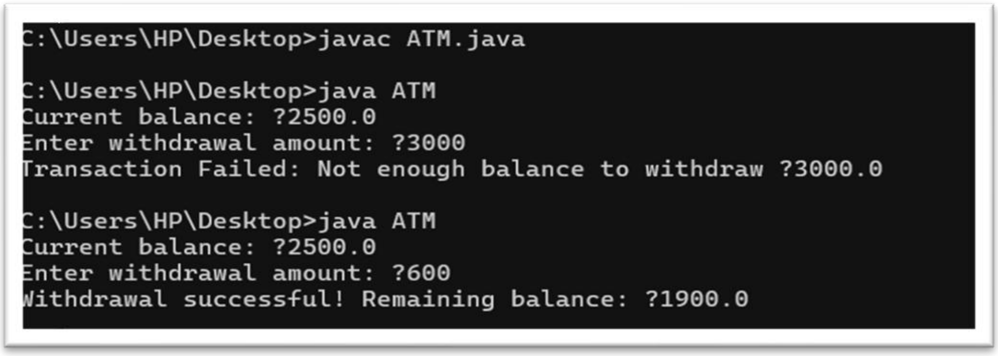


```
C:\Users\HP\Desktop>javac FH.java

C:\Users\HP\Desktop>java FH
File not found. Please check the file name or path.
```

c) Code:

```
import java.util.Scanner;
class InsufficientFundsException extends Exception {
  public InsufficientFundsException(String message) {
    super(message);
  }
}
public class ATM {
  public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    double balance = 2500.00;
    try {
      System.out.println("Current balance: ₹" + balance);
      System.out.print("Enter withdrawal amount: ₹");
      double amount = scanner.nextDouble();

      if (amount <= 0) {
        throw new IllegalArgumentException("Amount must be greater than zero.");
      }
      if (amount > balance) {
        throw new InsufficientFundsException("Not enough balance to withdraw ₹" + amount);
      }
      balance -= amount;
      System.out.println("Withdrawal successful! Remaining balance: ₹" + balance);
    } catch (InsufficientFundsException | IllegalArgumentException e) {
      System.out.println("Transaction Failed: " + e.getMessage());
    } finally {
      scanner.close();
    }
  }
}
```

Output:

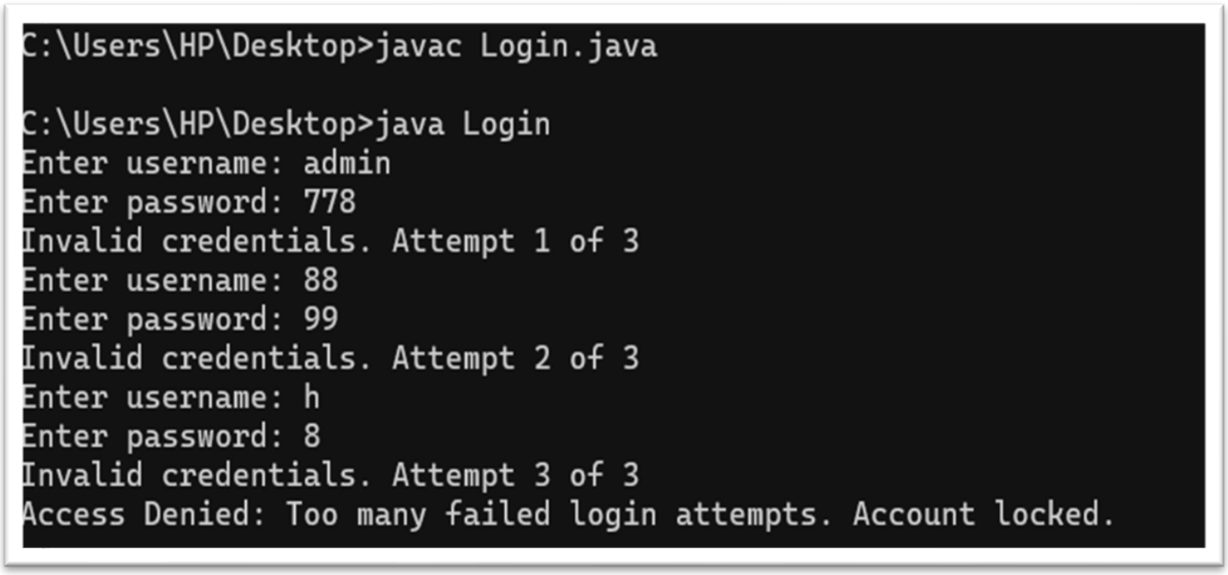


```
C:\Users\HP\Desktop>javac ATM.java

C:\Users\HP\Desktop>java ATM
Current balance: ?2500.0
Enter withdrawal amount: ?3000
Transaction Failed: Not enough balance to withdraw ?3000.0

C:\Users\HP\Desktop>java ATM
Current balance: ?2500.0
Enter withdrawal amount: ?600
Withdrawal successful! Remaining balance: ?1900.0
```

d) Code:

```
import java.util.Scanner;
class MaxLoginAttemptsExceededException extends Exception {
  public MaxLoginAttemptsExceededException(String message) {
    super(message);
  }
}
public class Login {
  public static void main(String[] args) {
    final String USERNAME = "admin";
    final String PASSWORD = "123";
    int attempts = 0;
    final int MAX_ATTEMPTS = 3;
    Scanner scanner = new Scanner(System.in);
    try {
      while (attempts < MAX_ATTEMPTS) {
        System.out.print("Enter username: ");
        String user = scanner.nextLine();
        System.out.print("Enter password: ");
        String pass = scanner.nextLine();
        if (user.equals(USERNAME) && pass.equals(PASSWORD)) {
          System.out.println("Login successful!");
          return;
        } else {
          attempts++;
          System.out.println("Invalid credentials. Attempt " + attempts + " of " + MAX_ATTEMPTS);
        }
      }
      throw new MaxLoginAttemptsExceededException("Too many failed login attempts. Account
       locked.");
    } catch (MaxLoginAttemptsExceededException e) {
      System.out.println("Access Denied: " + e.getMessage());
    } finally {
      scanner.close();
    }
  }
}
```

Output:



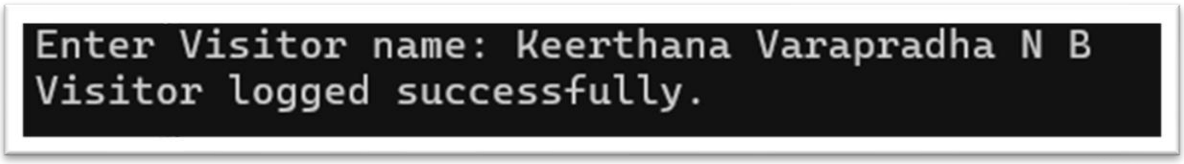
```
C:\Users\HP\Desktop>javac Login.java

C:\Users\HP\Desktop>java Login
Enter username: admin
Enter password: 778
Invalid credentials. Attempt 1 of 3
Enter username: 88
Enter password: 99
Invalid credentials. Attempt 2 of 3
Enter username: h
Enter password: 8
Invalid credentials. Attempt 3 of 3
Access Denied: Too many failed login attempts. Account locked.
```

17 File Handling Programs

a) Code:

```
 import java.io.FileWriter;
import java.io.IOException;
import java.util.Scanner;
public class V {
   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);
      System.out.print("Enter visitor name: ");
      String name = scanner.nextLine();
      try {
         FileWriter fw = new FileWriter("visitors.txt", true); // append mode
         fw.write(name + "\n");
         fw.close();
         System.out.println("Visitor logged successfully.");
      } catch (IOException e) {
         System.out.println("An error occurred while logging the visitor.");
      }
      scanner.close();
   }
}
```

Output:

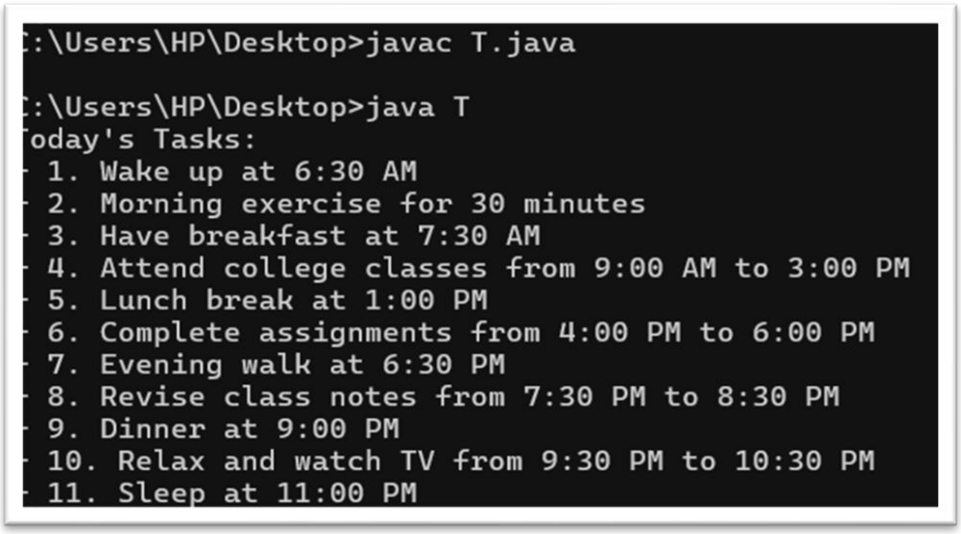


```
Enter Visitor name: Keerthana Varapradha N B
Visitor logged successfully.
```

b) Code:

```
import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;
public class T{
   public static void main(String[] args) throws FileNotFoundException {
      File file = new File("tasks.txt");
      Scanner scanner = new Scanner(file);
      System.out.println("Today's Tasks:");
      while (scanner.hasNextLine()) {
         System.out.println("- " + scanner.nextLine());
      }
      scanner.close();
   }
}
```

Output:

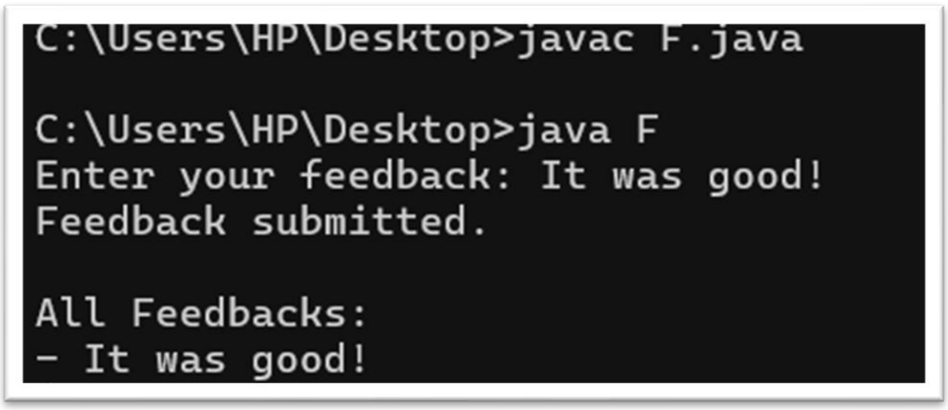


```
:\Users\HP\Desktop>javac T.java

:\Users\HP\Desktop>java T
oday's Tasks:
 1. Wake up at 6:30 AM
 2. Morning exercise for 30 minutes
 3. Have breakfast at 7:30 AM
 4. Attend college classes from 9:00 AM to 3:00 PM
 5. Lunch break at 1:00 PM
 6. Complete assignments from 4:00 PM to 6:00 PM
 7. Evening walk at 6:30 PM
 8. Revise class notes from 7:30 PM to 8:30 PM
 9. Dinner at 9:00 PM
 10. Relax and watch TV from 9:30 PM to 10:30 PM
 11. Sleep at 11:00 PM
```

c) Code:

```
import java.io.*;
import java.util.Scanner;

public class F {
    public static void main(String[] args) throws IOException {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter your feedback: ");
        String feedback = scanner.nextLine();

        FileWriter fw = new FileWriter("feedback.txt", true);
        fw.write(feedback + "\n");
        fw.close();
        System.out.println("Feedback submitted.");

        System.out.println("\nAll Feedbacks:");
        BufferedReader br = new BufferedReader(new FileReader("feedback.txt"));
        String line;
        while ((line = br.readLine()) != null) {
            System.out.println("- " + line);
        }
        br.close();

        scanner.close();
    }
}
```

Output:

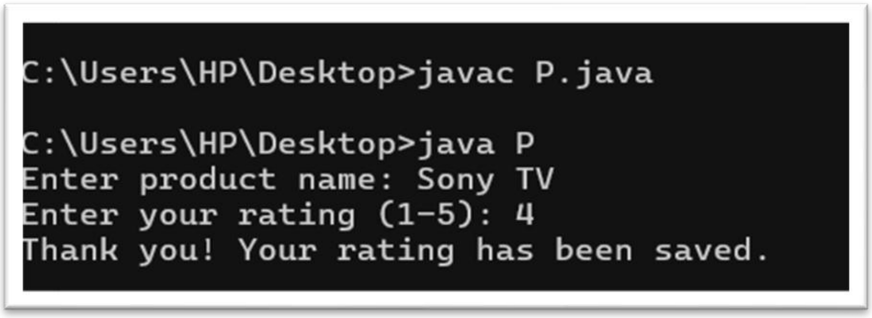


```
C:\Users\HP\Desktop>javac F.java

C:\Users\HP\Desktop>java F
Enter your feedback: It was good!
Feedback submitted.

All Feedbacks:
- It was good!
```

d) Code:

```
import java.io.FileWriter;
import java.util.Scanner;
public class P {
  public static void main(String[] args) Exception {
    Scanner scanner = new Scanner(System.in);
    System.out.print("Enter product name: ");
    String product = scanner.nextLine();
    System.out.print("Enter your rating (1-5): ");
    int rating = scanner.nextInt();
    FileWriter fw = new FileWriter("ratings.txt", true);
    fw.write(product + " - " + rating + " stars\n");
    fw.close();
    System.out.println("Thank you! Your rating has been saved.");
    scanner.close();
  }
}
```

Output:



```
C:\Users\HP\Desktop>javac P.java

C:\Users\HP\Desktop>java P
Enter product name: Sony TV
Enter your rating (1-5): 4
Thank you! Your rating has been saved.
```