



**SCHOOL OF  
COMPUTING**

# **LAB RECORD**

23CSE111 – Object Oriented Programming

*Submitted by*

**CH.SC.U4CSE24115 – Dhaksin Kaarthick**

**BACHELOR OF TECHNOLOGY**  
**IN**  
**COMPUTER SCIENCE AND ENGINEERING**

**AMRITA VISHWA VIDYAPEETHAM**  
**AMRITA SCHOOL OF COMPUTING**

**CHENNAI**



SCHOOL OF  
COMPUTING

**AMRITA VISHWA VIDYAPEETHAM**  
**AMRITA SCHOOL OF COMPUTING, CHENNAI**

**BONAFIDE CERTIFICATE**

This is to certify that the Lab Record work for 23CSE111-Object Oriented Programming Subject submitted by **CH.SC.U4CSE24115 – Dhaksin Kaarthick S.U** 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.

Internal Examiner 1

Internal Examiner 2

# Index

S.NO	Experiment Name	Page Number
1.	UML DIAGRAM	4
	Online Shopping <ul style="list-style-type: none"> <li>• Class Diagram</li> <li>• Use Case Diagram</li> <li>• Sequence Diagram</li> <li>• State Activity Diagram</li> <li>• Object Diagram</li> </ul>	4
2.	Library Management <ul style="list-style-type: none"> <li>• Class Diagram</li> <li>• Use Case Diagram</li> <li>• Sequence Diagram</li> <li>• State Activity Diagram</li> <li>• Object Diagram</li> </ul>	8
3.	Java Basic Programs	12
i)	Armstrong Number:	12
ii)	Count Digits	13
iii)	Even or Odd:	14
iv)	Factorial	15
v)	Fibonacci	16
vi)	Palindrome Check	17
vii)	Prime Check	18
viii)	Print Number	19
xi)	Reverse Number:	20
x)	Sum Of Natural Number	21

	<b>INHERITANCE:</b>	
4.	Single Inheritance	
i)	Students Details	23
ii	Bank Details	
5.	<b>MULTILEVEL INHERITANCE PROGRAMS</b>	
i)	General Details	
ii)	Employee Salary	
6.	<b>HIERARCHICAL INHERITANCE PROGRAMS</b>	
i)	Vehicle Model	
ii)	Person Details	
7.	<b>HYBRID INHERITANCE PROGRAMS</b>	
i)	Student Details	
ii)	Vehicle type	
	<b>POLYMORPHISM</b>	
8.	<b>CONSTRUCTOR PROGRAMS</b>	
i)	Book Details	
9.	<b>CONSTRUCTOR OVERLOADING PROGRAMS</b>	
i)	Employee info	
10.	<b>METHOD OVERLOADING PROGRAMS</b>	
i)	Employee details	
ii)	Shape details	
11.	<b>METHOD OVERRIDING PROGRAMS</b>	
i)	Vehicle car bike class	

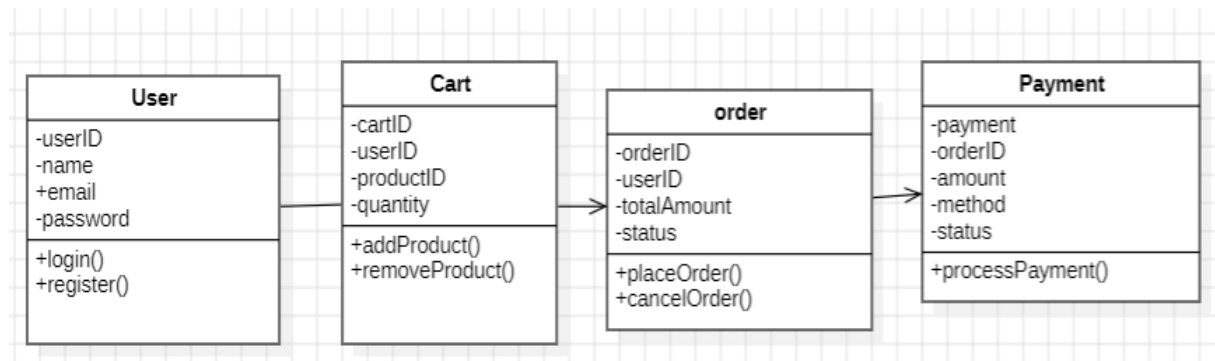
ii)	BankAccount interest Rate	
	<b>ABSTRACTION</b>	
12.	<b>INTERFACE PROGRAMS</b>	
i)	Shape Area	
ii)	Camara and Music Player class	
iii)	Payment Type	
iv)	FuelType	
13.	<b>ABSTRACT CLASS PROGRAMS</b>	
i)	Mileage Calculator	
ii)	Vehicle start and stop	
iii)	Area of shape	
iv)	Shape 2D	
	<b>ENCAPSULATION</b>	
14.	<b>ENCAPSULATION PROGRAMS</b>	
i)	Patients Records	
ii)	Product Details	
iii)	House Details	
iv)	Game Characters	
15.	<b>PACKAGES PROGRAMS</b>	
i)	Button creation	
ii)	Basic Calculation	
iii)	Area calculation	
iv)	Employee Details	
16.	<b>EXCEPTION HANDLING PROGRAMS</b>	
i)	Division by 0	
ii)	Age limit with throws keyword	

iii)	Age limit with throw	
iv)	File missing error	
17.	<b>FILE HANDLING PROGRAMS</b>	
i)	Reading the file	
ii)	Writing the file	
iii)	Reading N writing to a file	
iv)	writing to file multiple try catch	

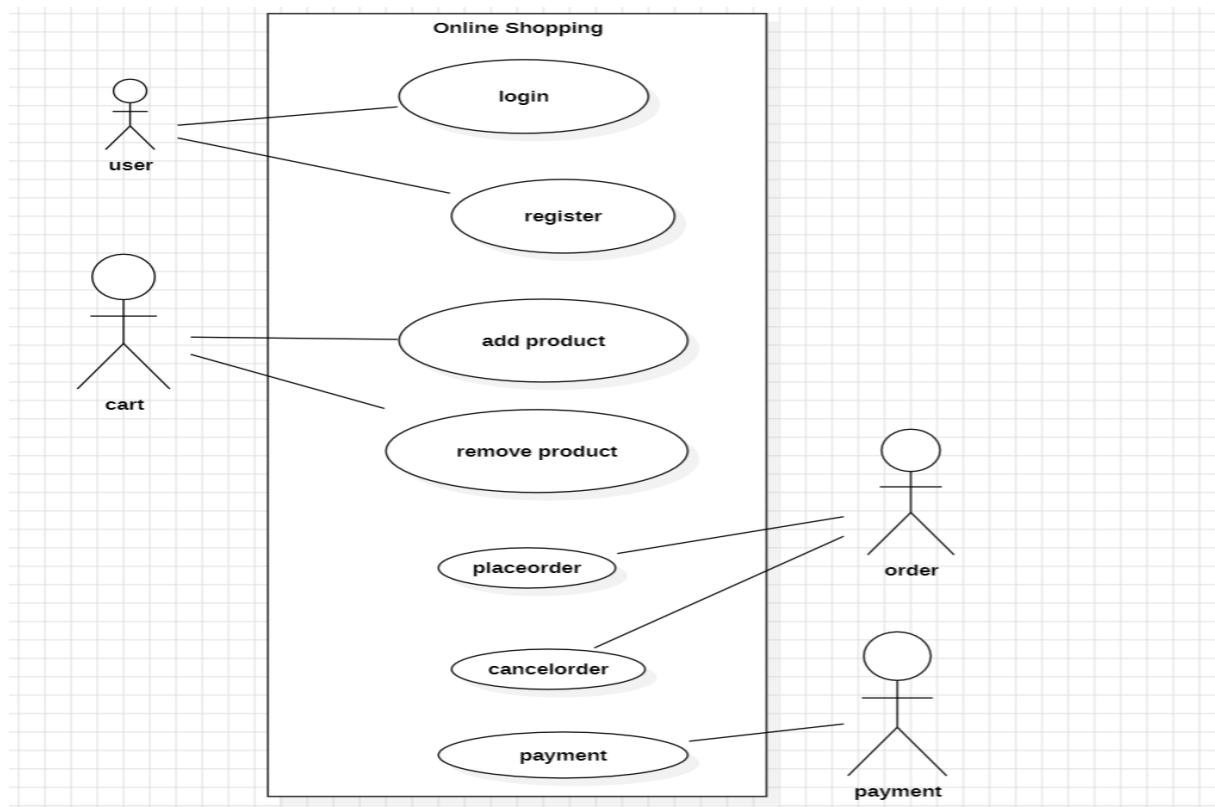
# UML DIAGRAM

## 1. Online Shopping

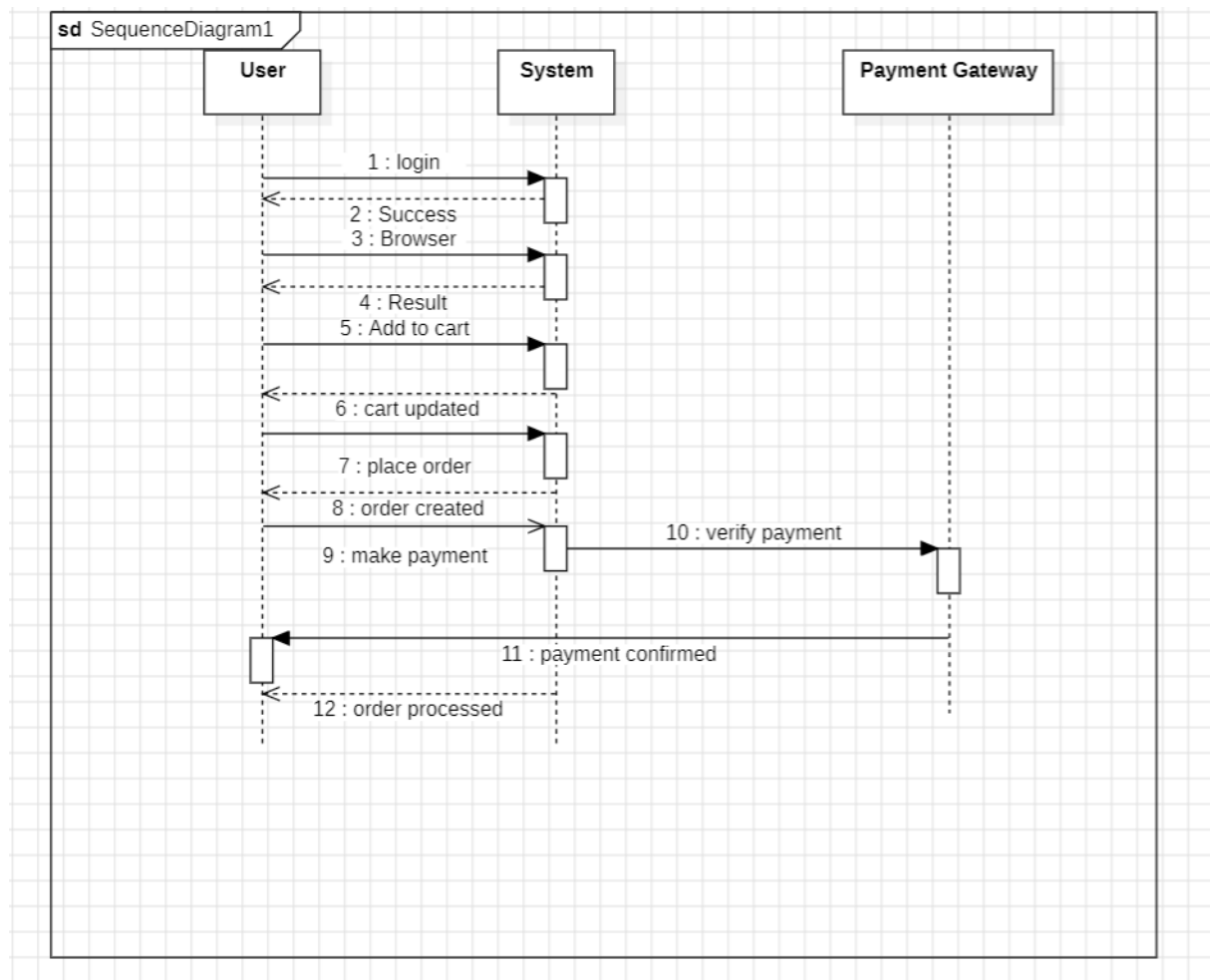
### i. Class Diagram:



### ii. Use Case Diagram:

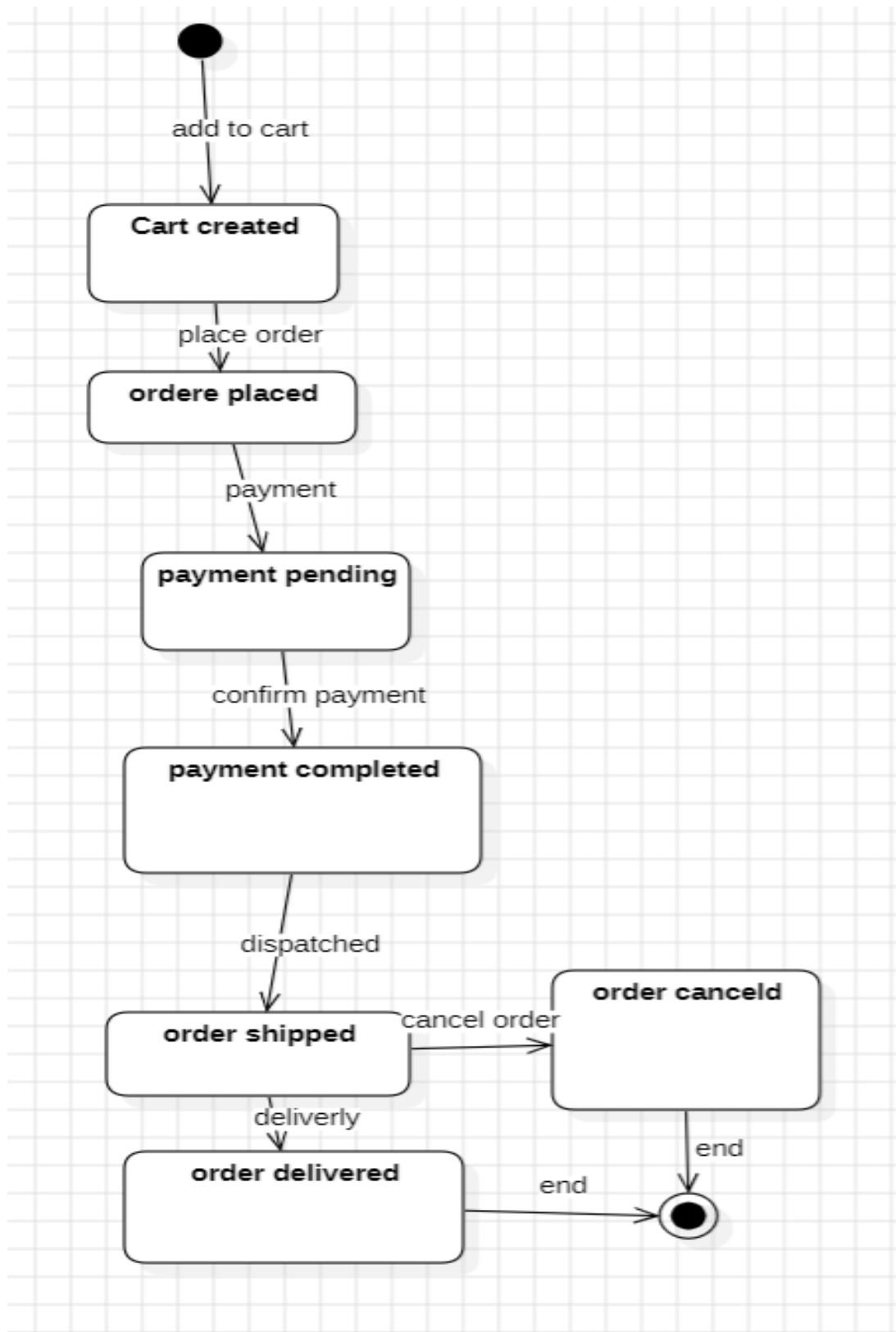


### iii. Sequence Diagram:

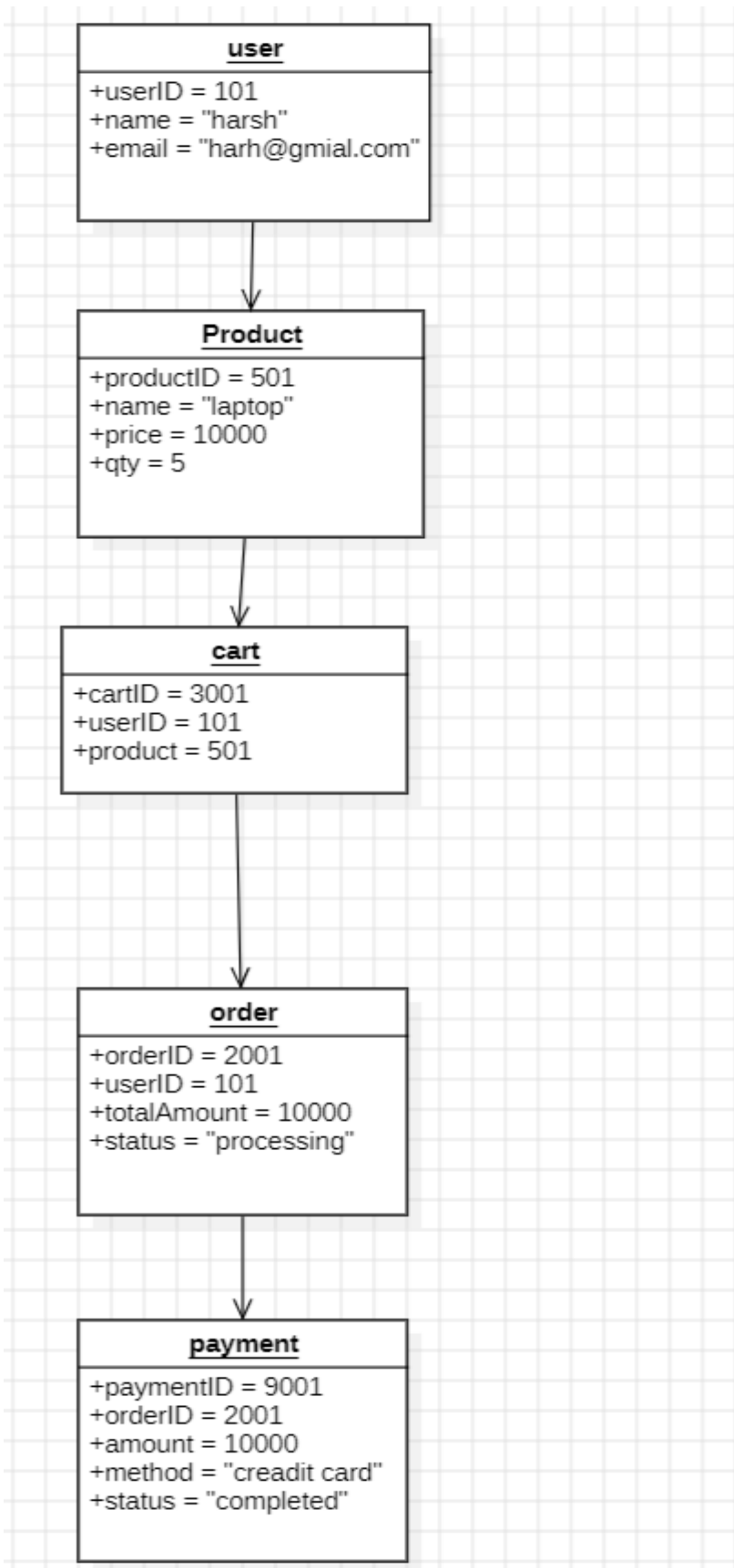




iv. State Activity Diagram:

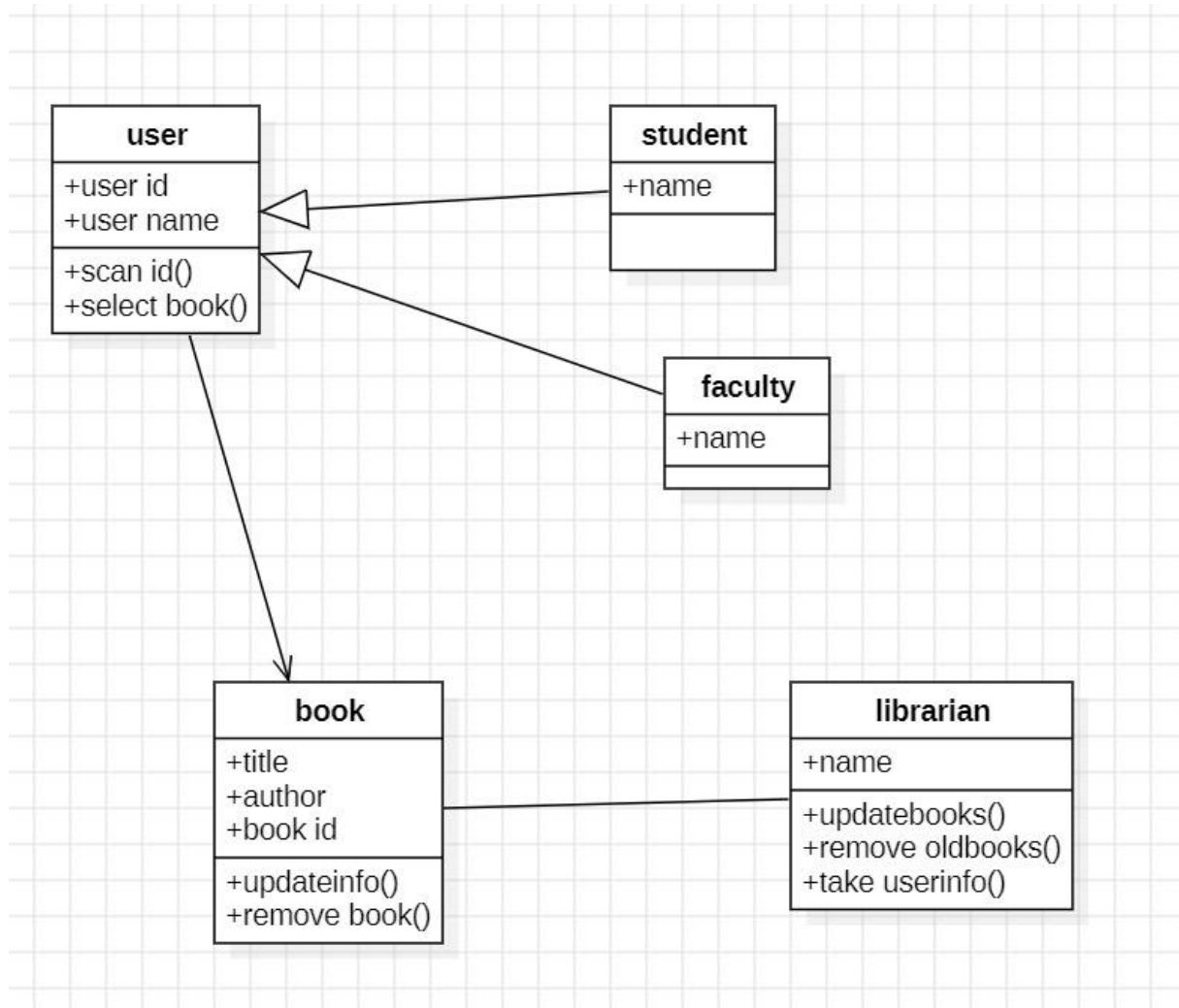


## v. Object Diagram:

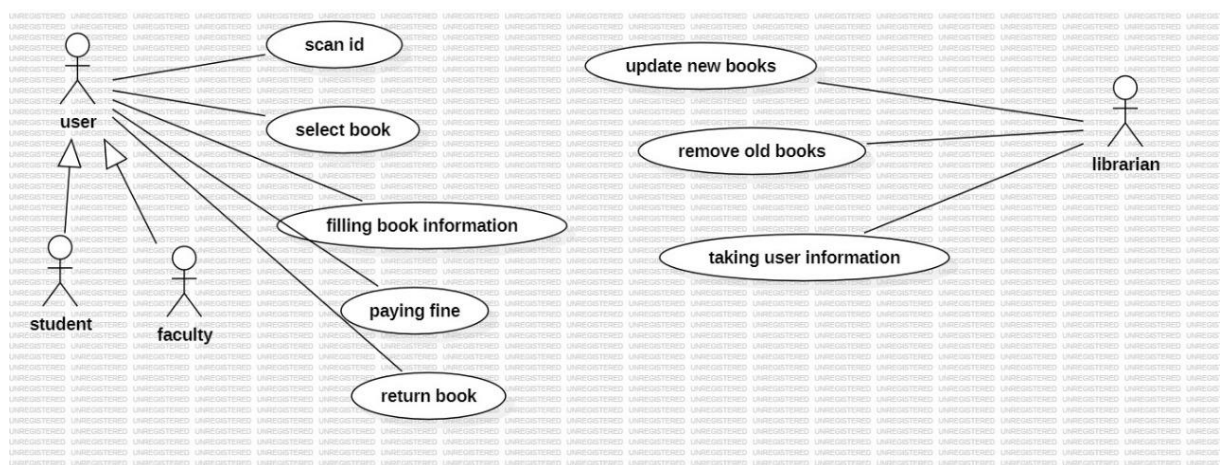


## 2. Library Management:

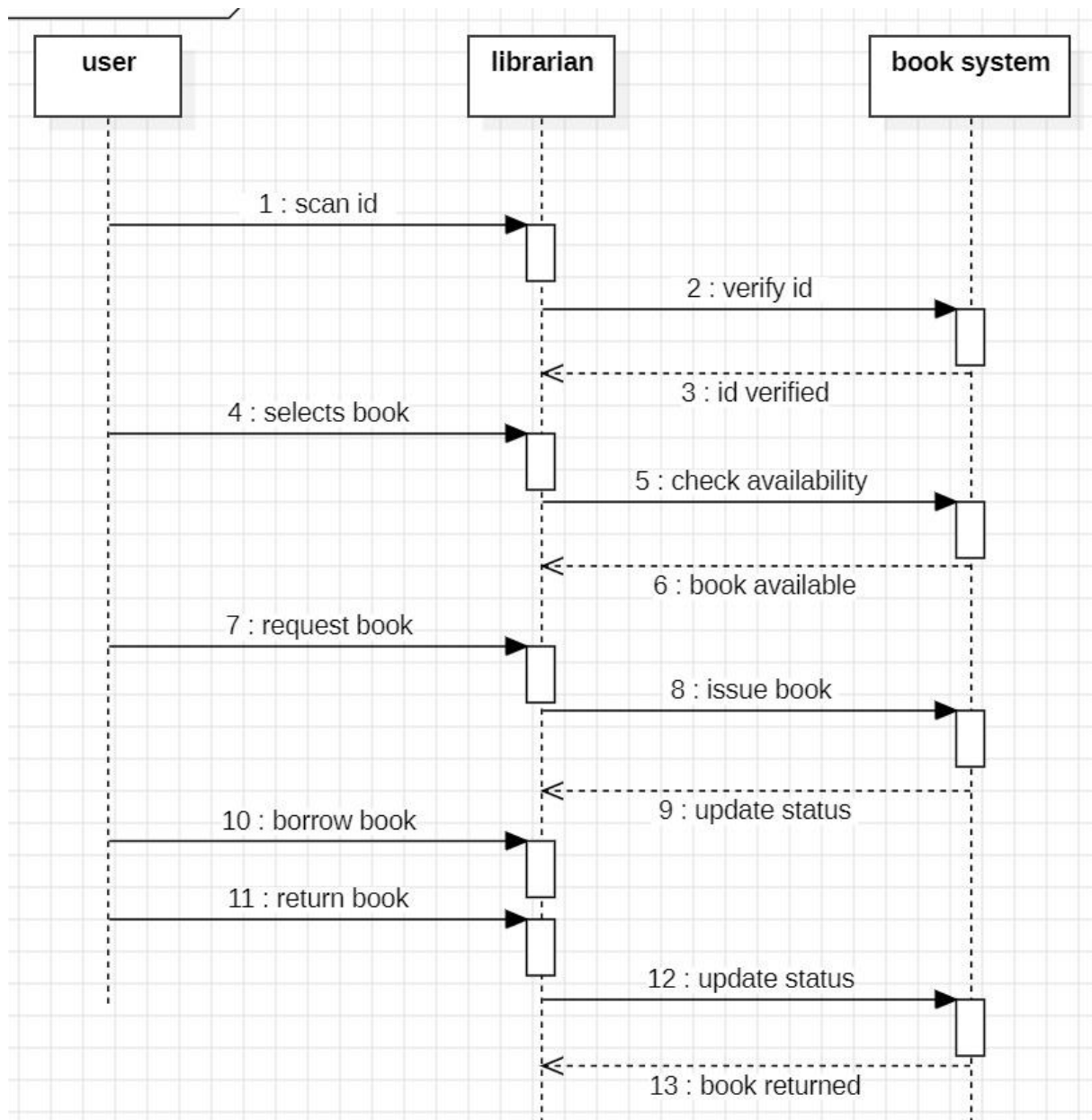
### i. Class Diagram:



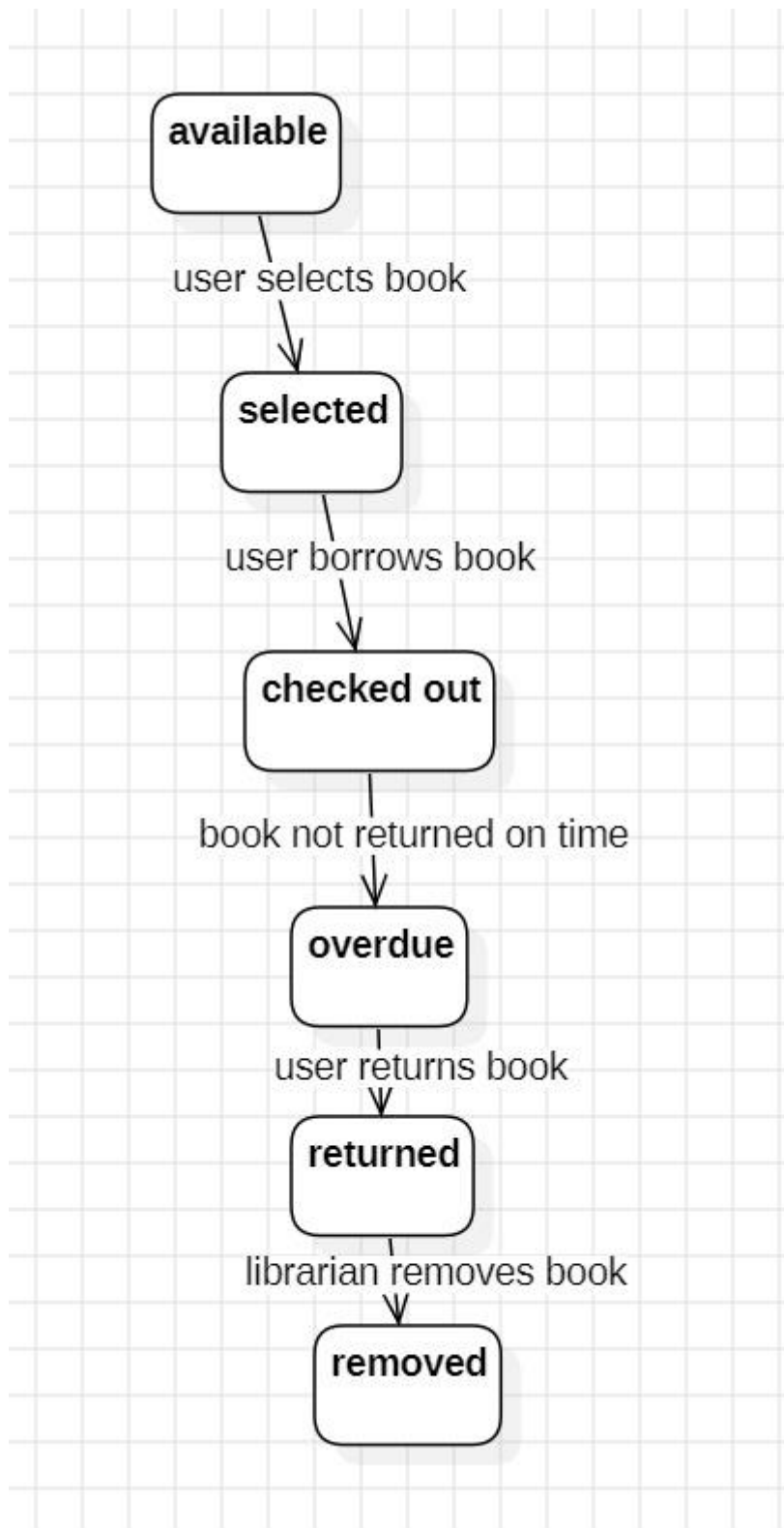
### ii. Use Case Diagram:



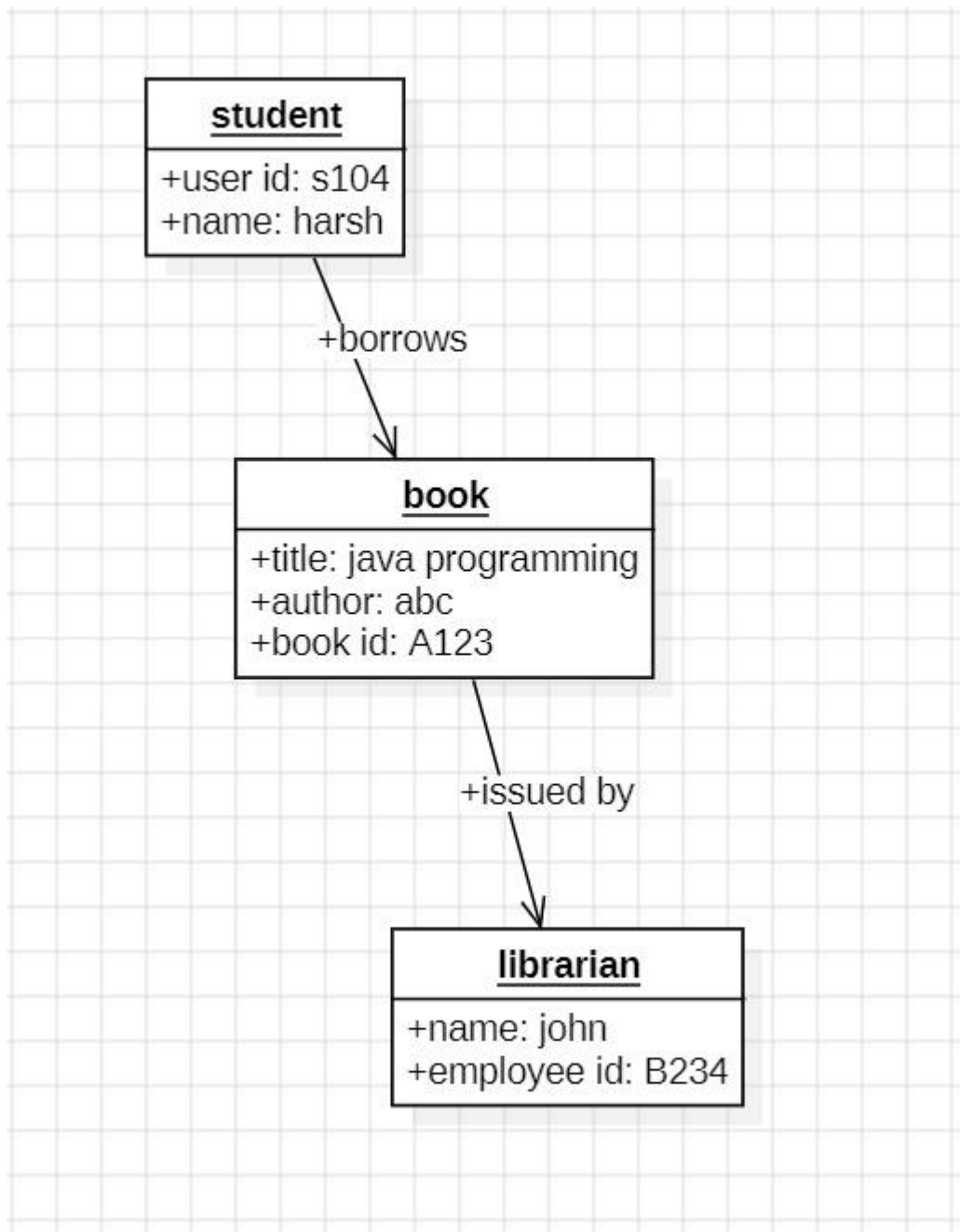
### iii. Sequence Diagram:



#### iv. State Activity Diagram:



v. Object Diagram:



# Java Basic Programs

## 1. Armstrong Number:

Code:

```
import java.util.Scanner;

public class ArmstrongNumber {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter a number: ");
        int num = sc.nextInt();
        int original = num, sum = 0, digit;

        while (num > 0) {
            digit = num % 10;
            sum += digit * digit * digit;
            num /= 10;
        }

        if (sum == original) {
            System.out.println(original + " is an Armstrong Number.");
        } else {
            System.out.println(original + " is not an Armstrong Number.");
        }

        sc.close();
    }
}
```

Output:

```
PS D:\Code\Basic Java Program> javac ArmstrongNumber.java
PS D:\Code\Basic Java Program> java ArmstrongNumber
Enter a number: 11
11 is not an Armstrong Number.
PS D:\Code\Basic Java Program> |
```

## 2.Count Digits:

Code:

```
import java.util.Scanner;

public class CountDigits {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter a number: ");
        int num = sc.nextInt();
        int count = 0;

        while (num > 0) {
            num /= 10;
            count++;
        }

        System.out.println("Number of digits: " + count);
        sc.close();
    }
}
```

Output:

```
PS D:\Code\Basic Java Program> javac CountDigits.java
PS D:\Code\Basic Java Program> java CountDigits
Enter a number: 11223344
Number of digits: 8
PS D:\Code\Basic Java Program> |
```



### 3. Even or Odd:

Code:

```
import java.util.Scanner;

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

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

Output:

```
PS D:\Code\Basic Java Program> javac EvenOdd.java
PS D:\Code\Basic Java Program> java EvenOdd
Enter a number: 23
23 is Odd.
PS D:\Code\Basic Java Program> |
```

## 4. Factorial:

Code:

```
import java.util.Scanner;

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

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

        System.out.println("Factorial: " + fact);
        sc.close();
    }
}
```

Output:

```
PS D:\Code\Basic Java Program> javac Factorial.java
PS D:\Code\Basic Java Program> java Factorial
Enter a number: 5
Factorial: 120
PS D:\Code\Basic Java Program> |
```

## 5.Fibonacci:

Code:

```
import java.util.Scanner;

public class Fibonacci {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the number of terms: ");
        int n = sc.nextInt();
        int a = 0, b = 1, next;

        System.out.print("Fibonacci Series: " + a + " " + b);

        for (int i = 2; i < n; i++) {
            next = a + b;
            System.out.print(" " + next);
            a = b;
            b = next;
        }

        sc.close();
    }
}
```

Output:

```
PS D:\Code\Basic Java Program> javac Fibonacci.java
PS D:\Code\Basic Java Program> java Fibonacci
Enter the number of terms: 10
Fibonacci Series: 0 1 1 2 3 5 8 13 21 34
PS D:\Code\Basic Java Program> |
```

## 6. Palindrome Check:

Code:

```
import java.util.Scanner;

public class PalindromeCheck {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter a number: ");
        int num = sc.nextInt();
        int original = num, rev = 0;

        while (num > 0) {
            rev = rev * 10 + num % 10;
            num /= 10;
        }

        if (original == rev) {
            System.out.println(original + " is a Palindrome.");
        } else {
            System.out.println(original + " is not a Palindrome.");
        }
        sc.close();
    }
}
```

Output:

```
PS D:\Code\Basic Java Program> javac PalindromeCheck.java
PS D:\Code\Basic Java Program> java PalindromeCheck
Enter a number: 123098
123098 is not a Palindrome.
PS D:\Code\Basic Java Program> |
```

## 7.Prime Check:

Code:

```
import java.util.Scanner;

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

        if (num <= 1) {
            isPrime = false;
        } else {
            for (int i = 2; i <= num / 2; 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.");
        }
        sc.close();
    }
}
```

Output:

```
PS D:\Code\Basic Java Program> javac PrimeCheck.java
PS D:\Code\Basic Java Program> java PrimeCheck
Enter a number: 2
2 is a Prime Number.
PS D:\Code\Basic Java Program> |
```

## 8. Print Number:

Code:

```
import java.util.Scanner;

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

        for (int i = 1; i <= n; i++) {
            System.out.print(i + " ");
        }
        sc.close();
    }
}
```

Output:

```
PS D:\Code\Basic Java Program> javac PrintNumbers.java
PS D:\Code\Basic Java Program> java PrintNumbers
Enter a number: 31
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
PS D:\Code\Basic Java Program> |
```

## 9.Reverse Number:

Code:

```
import java.util.Scanner;

public class ReverseNumber {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter a number: ");
        int num = sc.nextInt();
        int rev = 0;

        while (num > 0) {
            rev = rev * 10 + num % 10;
            num /= 10;
        }

        System.out.println("Reversed Number: " + rev);
        sc.close();
    }
}
```

Output:

```
PS D:\Code\Basic Java Program> javac ReverseNumber.java
PS D:\Code\Basic Java Program> java ReverseNumber
Enter a number: 123456789
Reversed Number: 987654321
PS D:\Code\Basic Java Program> |
```

## 10. Sum Of Natural Number:

Code:

```
import java.util.Scanner;

public class SumNaturalNumbers {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter a number: ");
        int n = sc.nextInt();
        int sum = 0;

        for (int i = 1; i <= n; i++) {
            sum += i;
        }

        System.out.println("Sum: " + sum);
        sc.close();
    }
}
```

Output:

```
PS D:\Code\Basic Java Program> javac SumNaturalNumbers.java
PS D:\Code\Basic Java Program> java SumNaturalNumbers
Enter a number: 50
Sum: 1275
PS D:\Code\Basic Java Program> |
```



# INHERITANCE

4)

## i) Students Details:

### CODE:

```
class College {  
    String collegeName = "AMRITA";  
    String address = "CHENNAI, India";  
  
    void showCollegeDetails() {  
        System.out.println("College Name: " + collegeName);  
        System.out.println("Address: " + address);  
    }  
}  
  
class Student extends College {  
    String studentName;  
    int rollNumber;  
  
    Student(String studentName, int rollNumber) {  
        this.studentName = studentName;  
        this.rollNumber = rollNumber;  
    }  
  
    void showStudentDetails() {  
        System.out.println("Student Name: " + studentName);  
        System.out.println("Roll Number: " + rollNumber);  
    }  
}
```

```
}  
  
public class SingleInheritanceExample1 {  
    public static void main(String[] args) {  
        Student s1 = new Student("Rahul", 101);  
        s1.showCollegeDetails();  
        s1.showStudentDetails();  
    }  
}
```

## OUTPUT:

```
College Name: AMRITA  
Address: CHENNAI, India  
Student Name: Rahul  
Roll Number: 101
```

## ii)Bank Details

### CODE:

```
class BankAccount {  
    String accountHolder;  
    double balance;  
  
    BankAccount(String accountHolder, double balance) {  
        this.accountHolder = accountHolder;  
        this.balance = balance;  
    }  
  
    void showBalance() {  
        System.out.println("Account Holder: " + accountHolder);  
    }  
}
```

```
        System.out.println("Balance: $" + balance);
    }
}

class SavingsAccount extends BankAccount {
    double interestRate = 5.0;

    SavingsAccount(String accountHolder, double balance) {
        super(accountHolder, balance);
    }

    void calculateInterest() {
        double interest = (balance * interestRate) / 100;
        System.out.println("Annual Interest: $" + interest);
    }
}

public class SingleInheritanceExample2 {
    public static void main(String[] args) {
        SavingsAccount acc1 = new SavingsAccount("John Doe", 5000);
        acc1.showBalance();
        acc1.calculateInterest();
    }
}
```

# MULTILEVEL INHERITANCE

**5.**

## **i)General Details**

### **CODE:**

```
class LivingBeing {  
    void breathe() {  
        System.out.println("Living beings breathe.");  
    }  
}  
  
class Human extends LivingBeing {  
    void speak() {  
        System.out.println("Humans can speak.");  
    }  
}  
  
class Student extends Human {  
    String name;  
    int studentID;  
  
    Student(String name, int studentID) {  
        this.name = name;  
        this.studentID = studentID;  
    }  
  
    void study() {
```

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

    void showDetails() {
        System.out.println("Student Name: " + name);
        System.out.println("Student ID: " + studentID);
    }
}

public class MultilevelExample1 {
    public static void main(String[] args) {
        Student s1 = new Student("Rahul", 101);
        s1.breathe();
        s1.speak();
        s1.study();
        s1.showDetails();
    }
}
```

## OUTPUT:

```
Living beings breathe.
Humans can speak.
Rahul is studying.
Student Name: Rahul
Student ID: 101
```

## ii)Employee

### CODE:

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

class Employee extends Person {
    int employeeID;
    double salary;
    Employee(String name, int employeeID, double salary) {
        super(name);
        this.employeeID = employeeID;
        this.salary = salary;
    }
    void showEmployee() {
        System.out.println("Employee ID: " + employeeID);
        System.out.println("Salary: $" + salary);
    }
}

class Manager extends Employee {
```

```

    String department;

    Manager(String name, int employeeID, double salary, String
department) {
        super(name, employeeID, salary);
        this.department = department;
    }

    void showManager() {
        System.out.println("Department: " + department);
        System.out.println("Role: Manager");
    }
}

public class MultilevelExample2 {
    public static void main(String[] args) {
        Manager m1 = new Manager("Alice", 2001, 75000, "HR");
        m1.showPerson();
        m1.showEmployee();
        m1.showManager();
    }
}

```

## OUTPUT:

```

Person Name: Alice
Employee ID: 2001
Salary: $75000.0
Department: HR
Role: Manager

```

## HIERARCHICAL INHERITANCE PROGRAMS

6.

**i)Code:**

```
class Vehicle {
    private String brand;
    private String model;
    public Vehicle(String brand, String model) {
        this.brand = brand;
        this.model = model;
    }
    public void start() {
        System.out.println("Vehicle is starting.");
    }
    public void stop() {
        System.out.println("Vehicle is stopping.");
    }
    public String getBrand() {
        return brand;
    }
    public String getModel() {
        return model;
    }
}
class Car extends Vehicle {
    private int numberOfDoors ;
    public Car(String brand, String model, int numberOfDoors) {
        super(brand, model);
        this.numberOfDoors = numberOfDoors;
    }
    public void drive() {
        System.out.println("Car is driving.");
    }
    public int getNumberOfDoors() {
        return numberOfDoors;
    }
}
class ElectricCar extends Car {
    private int batteryCapacity;
```



```

    public ElectricCar(String brand, String model, int
numberOfDoors, int batteryCapacity) {
        super(brand, model, numberOfDoors);
        this.batteryCapacity = batteryCapacity;
    }
    public void charge() {
        System.out.println("Electric car is charging.");
    }
    public int getBatteryCapacity() {
        return batteryCapacity;
    }
}
class Truck extends Vehicle {
    private double cargoCapacity;
    public Truck(String brand, String model, double
cargoCapacity) {
        super(brand, model);
        this.cargoCapacity = cargoCapacity;
    }
    public void loadCargo() {
        System.out.println("Truck is loading cargo.");
    }
    public double getCargoCapacity() {
        return cargoCapacity;
    }
}
public class Main {
    public static void main(String[] args) {
        Car car = new Car("Toyota", "Corolla", 4);
        car.start();
        car.drive();
        car.stop();
        System.out.println("Car doors: " +
car.getNumberOfDoors());
        ElectricCar electricCar = new ElectricCar("Tesla", "Model
S", 4, 100);
        electricCar.start();
        electricCar.drive();
        electricCar.charge();
    }
}

```

```
        System.out.println("Battery capacity: " +  
electricCar.getBatteryCapacity());  
        Truck truck = new Truck("Ford", "F-150", 2000.5);  
        truck.start();  
        truck.loadCargo();  
        truck.stop();  
        System.out.println("Cargo capacity: " +  
truck.getCargoCapacity());  
    }  
}
```

## OUTPUT:

```
Vehicle is starting.  
Car is driving.  
Vehicle is stopping.  
Car doors: 4  
Vehicle is starting.  
Car is driving.  
Electric car is charging.  
Battery capacity: 100  
Vehicle is starting.  
Truck is loading cargo.  
Vehicle is stopping.  
Cargo capacity: 2000.5
```

**ii)**

**Code:**

```
class Person {  
    private String name;  
    private int age;  
  
    public Person(String name, int age) {  
        this.name = name;  
        this.age = age;  
    }  
  
    public void displayDetails() {  
        System.out.println("Name: " + name + ", Age: " + age);  
    }  
}  
  
class Student extends Person {  
    private int studentId;  
    private String major;  
    public Student(String name, int age, int studentId, String major) {  
        super(name, age);  
        this.studentId = studentId;  
        this.major = major;  
    }  
    public void study() {  
        System.out.println("Student is studying " + major);  
    }  
}
```

```

public void displayDetails() {
    super.displayDetails();
    System.out.println("Student ID: " + studentId + ", Major: " +
major);
}
}

class Professor extends Person {
    private String department;
    private String researchArea;

    public Professor(String name, int age, String department, String
researchArea) {
        super(name, age);
        this.department = department;
        this.researchArea = researchArea;
    }

    public void teach() {
        System.out.println("Professor is teaching in " + department);
    }

    public void displayDetails() {
        super.displayDetails();
        System.out.println("Department: " + department + ", Research
Area: " + researchArea);
    }
}

class TeachingAssistant extends Student {
    private String course;

```

```

    public TeachingAssistant(String name, int age, int studentId, String
major, String course) {
        super(name, age, studentId, major);
        this.course = course;
    }
    public void assist() {
        System.out.println("Teaching assistant is assisting in " + course);
    }
    public void displayDetails() {
        super.displayDetails();
        System.out.println("Course: " + course);
    }
}

public class Main2 {
    public static void main(String[] args) {
        Student student = new Student("Alice", 20, 101, "Computer
Science");
        student.displayDetails();
        student.study();

        Professor professor = new Professor("Dr. Smith", 45, "Computer
Science", "AI");
        professor.displayDetails();
        professor.teach();

        TeachingAssistant ta = new TeachingAssistant("Bob", 25, 102,
"Mathematics", "Calculus");
        ta.displayDetails();
        ta.study();
    }
}

```

```
        ta.assist();  
    }  
}
```

## OUTPUT:

```
Name: Alice, Age: 20  
Student ID: 101, Major: Computer Science  
Student is studying Computer Science  
Name: Dr. Smith, Age: 45  
Department: Computer Science, Research Area: AI  
Professor is teaching in Computer Science  
Name: Bob, Age: 25  
Student ID: 102, Major: Mathematics  
Course: Calculus  
Student is studying Mathematics  
Teaching assistant is assisting in Calculus
```

## HYBRID INHERITANCE PROGRAMS

7.

i)

**CODE:**

```
class Person {  
    String name;  
  
    Person(String name) {  
        this.name = name;  
    }  
  
    void showDetails() {  
        System.out.println("Name: " + name);  
    }  
}  
  
class Student extends Person {  
    int studentID;  
  
    Student(String name, int studentID) {  
        super(name);  
        this.studentID = studentID;  
    }  
}
```

```

void study() {
    System.out.println(name + " is studying.");
}
}

class Teacher extends Person {
    String subject;

    Teacher(String name, String subject) {
        super(name);
        this.subject = subject;
    }

    void teach() {
        System.out.println(name + " is teaching " + subject +
        ".");
    }
}

interface Assistant {
    void assist();
}

class TeachingAssistant extends Student implements Assistant {
    TeachingAssistant(String name, int studentID) {
        super(name, studentID);
    }
}

```



```
        public void assist() {  
            System.out.println(name + " is assisting in a lab  
session.");  
        }  
    }  
  
    public class HybridInheritanceExample1 {  
        public static void main(String[] args) {  
            TeachingAssistant ta = new TeachingAssistant("Alex",  
101);  
            ta.showDetails();  
            ta.study();  
            ta.assist();  
        }  
    }  
}
```

## OUTPUT:

```
Name: Alex  
Alex is studying.  
Alex is assisting in a lab session.
```

ii)

**CODE:**

```
class Vehicle {  
    void startEngine() {  
        System.out.println("Vehicle engine started.");  
    }  
}  
  
class Car extends Vehicle {  
    void drive() {  
        System.out.println("Car is driving.");  
    }  
}  
  
class Boat extends Vehicle {  
    void sail() {  
        System.out.println("Boat is sailing.");  
    }  
}  
  
interface Amphibious {  
    void switchMode();  
}  
  
class AmphibiousCar extends Car implements Amphibious {  
    public void switchMode() {  
        System.out.println("Switching between land and water mode.");  
    }  
  
    void sail() {
```

```
        System.out.println("Amphibious car is sailing on water.");
    }
}

public class HybridInheritanceExample2 {
    public static void main(String[] args) {
        AmphibiousCar ac = new AmphibiousCar();
        ac.startEngine();
        ac.drive();
        ac.switchMode();
        ac.sail();
    }
}
```

## OUTPUT:

```
Vehicle engine started.
Car is driving.
Switching between land and water mode.
Amphibious car is sailing on water.
```

# POLYMORPHISM

## CONSTRUCTOR PROGRAMS

8

i)

### CODE:

```
class Book {
    String title;
    int pages;
    Book(String t, int p) {
        title = t;
        pages = p;
    }
    Book(Book b) {
        title = b.title;
        pages = b.pages;
    }
    void display() {
        System.out.println("Book: " + title + ", Pages: " + pages);
    }
}

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

```
    Book b1 = new Book("Java Programming", 500);  
    Book b2 = new Book(b1);  
    b1.display();  
    b2.display();  
}  
}
```

## OUTPUT:

```
Book: Java Programming, Pages: 500  
Book: Java Programming, Pages: 500
```

# CONSTRUCTOR OVERLOADING PROGRAMS

9.

i)

## CODE:

```
class Employee {
```

```
    String name;
```

```
    int age;
```

```
    double salary;
```

```
    Employee() {
```

```
        name = "Unknown";
```

```
        age = 18;
```

```
        salary = 30000;
```

```
    }
```

```
    Employee(String n, int a) {
```

```
        name = n;
```

```
        age = a;
```

```
        salary = 40000;
```

```
    }
```

```
    Employee(String n, int a, double s) {
```

```
        name = n;
```

```

        age = a;
        salary = s;
    }

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

public class ConstructorOverloadingExample {
    public static void main(String[] args) {
        Employee e1 = new Employee();
        Employee e2 = new Employee("John", 25);
        Employee e3 = new Employee("Alice", 30, 60000);

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

```

## OUTPUT:

```

Name: Unknown, Age: 18, Salary: $30000.0
Name: John, Age: 25, Salary: $40000.0
Name: Alice, Age: 30, Salary: $60000.0

```

# METHOD OVERLOADING PROGRAMS

**10.**

**i)**

**CODE:**

```
class Employee {  
    private String name;  
    private int id;  
    private double salary;  
    void setDetails(String name, int id) {  
        this.name = name;  
        this.id = id;  
    }  
    void setDetails(String name, int id, double salary) {  
        this.name = name;  
        this.id = id;  
        this.salary = salary;  
    }  
    void setDetails(String name) {  
        this.name = name;  
    }  
    void displayDetails() {
```



```
        System.out.println("Name: " + name + ", ID: " + id + ",  
Salary: " + salary);  
    }  
}  
  
public class Main {  
    public static void main(String[] args) {  
        Employee emp1 = new Employee();  
        emp1.setDetails("Alice", 101); // Calls first method  
        emp1.displayDetails();  
        Employee emp2 = new Employee();  
        emp2.setDetails("Bob", 102, 50000.0); // Calls second  
method  
        emp2.displayDetails();  
        Employee emp3 = new Employee();  
        emp3.setDetails("Charlie"); // Calls third method  
        emp3.displayDetails();  
    }  
}
```

## OUTPUT:

```
Name: Alice, ID: 101, Salary: 0.0  
Name: Bob, ID: 102, Salary: 50000.0  
Name: Charlie, ID: 0, Salary: 0.0
```

ii)

## CODE:

```
class Shape {  
    void draw(double radius) {  
        System.out.println("Drawing a circle with radius: " +  
radius);  
    }  
    void draw(double length, double width) {  
        System.out.println("Drawing a rectangle with length: " +  
length + " and width: " + width);  
    }  
    void draw(double side1, double side2, double side3) {  
        System.out.println("Drawing a triangle with sides: " +  
side1 + ", " + side2 + ", " + side3);  
    }  
}  
  
public class Main2 {  
    public static void main(String[] args) {  
        Shape shape = new Shape();  
        shape.draw(5.0);  
        shape.draw(4.0, 6.0);  
        shape.draw(3.0, 4.0, 5.0);    }  
}
```

## OUTPUT:

```
Drawing a circle with radius: 5.0  
Drawing a rectangle with length: 4.0 and width: 6.0  
Drawing a triangle with sides: 3.0, 4.0, 5.0
```

## METHOD OVERRIDING PROGRAMS

11

i)

### CODE:

```
class Vehicle {  
    void speed() {  
        System.out.println("Vehicle is moving");  
    }  
}  
  
class Car extends Vehicle {  
    void speed() {  
        System.out.println("Car moves at 80 km/h");  
    }  
}  
  
class Bike extends Vehicle {  
    void speed() {  
        System.out.println("Bike moves at 60 km/h");  
    }  
}  
  
public class Main {  
    public static void main(String[] args) {  
        Vehicle v;
```

```
        v = new Car();
        v.speed();
        v = new Bike();
        v.speed();
    }
}
```

## OUTPUT:

```
Car moves at 80 km/h
Bike moves at 60 km/h
```

ii)

## CODE:

```
class Bank {
    double getInterestRate() {
        return 0;
    }
}

class SBI extends Bank {
    double getInterestRate() {
        return 5.5;
    }
}

class ICICI extends Bank {
    double getInterestRate() {
        return 6.7;
    }
}

class HDFC extends Bank {
    double getInterestRate() {
        return 7.2;
    }
}
```

```
}

public class Main2 {
    public static void main(String[] args) {
        Bank b;

        b = new SBI();
        System.out.println("SBI Interest Rate: " +
b.getInterestRate() + "%");

        b = new ICICI();
        System.out.println("ICICI Interest Rate: " +
b.getInterestRate() + "%");

        b = new HDFC();
        System.out.println("HDFC Interest Rate: " +
b.getInterestRate() + "%");
    }
}
```

### **OUTPUT:**

```
SBI Interest Rate: 5.5%
ICICI Interest Rate: 6.7%
HDFC Interest Rate: 7.2%
```

# INTERFACE PROGRAMS

12

i)

**CODE:**

```
interface Shape{
void getArea();
}

class Rectangle implements Shape{
    double lenght,breadth;

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

    public void getArea(){
        System.out.println("The area of rectangle
"+(lenght*breadth));
    }
}

class Circle implements Shape{
    double radius;

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

```
        public void getArea(){
            System.out.println("The are of circle is
            "+(3.14*(radius*radius)));
        }
    }

    public class Infferface1 {
        public static void main(String[] args) {
            Rectangle r1=new Rectangle(12, 12);
            r1.getArea();
            Circle c1=new Circle(4);
            c1.getArea();
        }
    }
```

## OUTPUT:

```
The area of rectangle 144.0
The are of circle is 50.24
```

ii)

## CODE:

```
interface Camara{
    void takePhoto();
}

interface MusicPlayer{
    void canPlayMusic();
}

class SmartPhone implements Camara,MusicPlayer{
    public void takePhoto(){
        System.out.println("SmartPhone can take photo");
    }
    public void canPlayMusic(){
        System.out.println("SmartPhone can play music");
    }
}

public class Infferface2 {
    public static void main(String[] args) {
        SmartPhone s1=new SmartPhone();
        s1.canPlayMusic();
        s1.takePhoto();
    }
}
```



## OUTPUT:

```
SmartPhone can play music  
SmartPhone can take photo
```

iii)

## CODE:

```
interface Payment {
```

```
    void initiatePayment(double amount);
```

```
    void getPaymentStatus();
```

```
}
```

```
class CreditCard implements Payment {
```

```
    public void initiatePayment(double amount) {
```

```
        System.out.println("Processing Credit Card payment of  
$" + amount);
```

```
    }
```

```
    public void getPaymentStatus() {
```

```
        System.out.println("Credit Card payment successful!");
```

```
    }
```

```
}
```

```
class PayPal implements Payment {
```

```

    public void initiatePayment(double amount) {
        System.out.println("Processing PayPal payment of $" +
amount);
    }

    public void getPaymentStatus() {
        System.out.println("PayPal payment successful!");
    }
}

public class Infferface3 {
    public static void main(String[] args) {
        Payment creditCardPayment = new CreditCard();
        Payment paypalPayment = new PayPal();

        creditCardPayment.initiatePayment(50.0);
        creditCardPayment.getPaymentStatus();

        paypalPayment.initiatePayment(30.0);
        paypalPayment.getPaymentStatus();
    }
}

```

## OUTPUT:

```
Processing Credit Card payment of $50.0
Credit Card payment successful!
Processing PayPal payment of $30.0
PayPal payment successful!
```

iv)

## CODE:

```
interface Vehicle{
    void start();
    void stop();
}

interface ElectricVehicle{
    void chargeBattery();
}

interface FuelVehicle{
    void refuel();
}

class HybridCar implements
Vehicle,ElectricVehicle,FuelVehicle{
    public void start(){
        System.out.println("Vehicle Started ");
    }
    public void stop(){
        System.out.println("Vehicle Stoped ");
    }
}
```

```

    }

    public void chargeBattery(){
        System.out.println("Vehicle charging battery ");
    }

    public void refuel(){
        System.out.println("Vehicle refilling the fuel tank ");
    }

}

public class Infferface4 {
    public static void main(String[] args) {
        HybridCar v1=new HybridCar();
        v1.chargeBattery();
        v1.start();
        v1.stop();
        v1.refuel();
    }
}

```

## OUTPUT:

```

Vehicle charging battery
Vehicle Started
Vehicle Stoped
Vehicle refilling the fuel tank

```

# ABSTRACT CLASS PROGRAMS

13

i)

**Code:**

```
public class Main{  
    public static void main(String[] args){  
        Car c1=new Lam();  
        c1.carName("lambogini");  
        c1.carSpeed(122,"lambogini");  
    }  
}
```

```
abstract class Car{  
    abstract void carName(String name);  
  
    abstract void carSpeed(int speed,String name);  
  
    abstract void Mileage(int fuel,double mileage);  
  
}
```

```
class Lam extends Car{  
    void carName(String name){  
        System.out.println("Your Car name is "+name);  
    }  
}
```

```

}

void carSpeed(int speed,String name){
System.out.println(name+" can travel at the speed of
"+speed+" km/h");
}

void Mileage(int fuel,double mileage){
System.out.println("I am "+ "can travel range of
"+(fuel*mileage));
}

}

```

## Output:

```

Your Car name is lambogini
lambogini can travel at the speed of 122 km/h

```

ii)

## Code:

```

abstract class Vehicle{
abstract void start(String name);

abstract void stop(String name);

}

class Car extends Vehicle{
void start(String name){
System.out.println( name+" is Starting");

```

```
}  
void stop(String name){  
    System.out.println(name+" is stopping");  
}  
}  
class Bike extends Vehicle{  
    void start(String name){  
        System.out.println( name+" is Starting");  
    }  
    void stop(String name){  
        System.out.println(name+" is stopping");  
    }  
}  
public class Main2 {  
    public static void main(String[] args){  
        Car v1=new Car();  
  
        v1.start("Lambo");  
        v1.stop("lambo");  
  
        Bike v2=new Bike();  
        v2.start("Ducati");  
        v2.stop("Ducati");  
    }  
}
```

}

## Output:

```
Lambo is Starting  
lambo is stopping  
Ducati is Starting  
Ducati is stopping
```

iii)

## Code:

```
abstract class Shape {  
    // Abstract method to calculate the area  
    abstract double calculateArea();  
}
```

```
class Square extends Shape {  
    private double side;  
  
    public Square(double side) {  
        this.side = side;  
    }
```

@Override



```
double calculateArea() {  
    return side * side; // Area of square: side2  
}  
}  
  
public class Main3 {  
    public static void main(String[] args) {  
        Shape square = new Square(4.0);  
        System.out.println("Area of the square: " +  
square.calculateArea());  
    }  
}
```

## Output:

```
Area of the square: 16.0
```

iv)

## Code:

```
abstract class Shape2D{  
    abstract void draw();  
    abstract void resize();  
}  
  
class Rectangle extends Shape2D{  
    void draw(){  
        System.out.println("you are Drawing Rectangle");  
    }  
}
```

```

}
void resize(){
System.out.println("you can resize the lenght and breadth of
rectangle");
}
}
class Circle extends Shape2D{
void draw(){
System.out.println("you are Drawing Circle");
}
void resize(){
System.out.println("you can resize the Radius of circle ");
}
}
public class Main4{
public static void main(String[] args){
Shape2D s1=new Rectangle();
s1.draw();
s1.resize();

Shape2D s2=new Circle();
s2.draw();
s2.resize();

```

}

}

## Output:

```
you are Drawing Rectangle
you can resize the lenght and breadth of rectangle
you are Drawing Circle
you can resize the Radius of circle
```

## ENCAPSULATION PROGRAMS

### 14

i)

### Code:

```
class Patient {
    private String name;
    private double weight; // in kg
    private double height; // in meters

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

    public void setWeight(double weight) {
        this.weight = weight;
    }
}
```

```
}
```

```
public void setHeight(double height) {  
    this.height = height;  
}
```

```
public String getName() {  
    return name;  
}
```

```
public double calculateBMI() {  
    return weight / (height * height);  
}
```

```
public String getHealthStatus() {  
    double bmi = calculateBMI();  
    if (bmi < 18.5) return "Underweight";  
    else if (bmi < 24.9) return "Normal weight";  
    else if (bmi < 29.9) return "Overweight";  
    else return "Obese";  
}  
}
```

```
public class PatientTest {
```

```
public static void main(String[] args) {  
    Patient p = new Patient();  
    p.setName("John Doe");  
    p.setWeight(70);  
    p.setHeight(1.75);  
    System.out.println("Patient: " + p.getName());  
    System.out.println("BMI: " + p.calculateBMI());  
    System.out.println("Health Status: " +  
p.getHealthStatus());  
}  
}
```

## OUTPUT:

```
PS D:\Dhaksin\00P\java oops\Encapsulation> javac PatientTest.java  
PS D:\Dhaksin\00P\java oops\Encapsulation> java PatientTest  
Patient: John Doe  
BMI: 22.857142857142858  
Health Status: Normal weight  
PS D:\Dhaksin\00P\java oops\Encapsulation> |
```

ii)

**Code:**

```
class Product {  
    private String productName;  
    private double price;  
    private int stock;  
  
    public void setProductName(String productName) {  
        this.productName = productName;  
    }  
  
    public void setPrice(double price) {  
        this.price = price > 0 ? price : 0;  
    }  
  
    public void setStock(int stock) {  
        this.stock = stock >= 0 ? stock : 0;  
    }  
  
    public String getProductName() {  
        return productName;  
    }  
  
    public double getPrice() {
```

```

        return price;
    }

    public boolean isAvailable() {
        return stock > 0;
    }
}

public class ProductTest {
    public static void main(String[] args) {
        Product product = new Product();
        product.setProductName("Laptop");
        product.setPrice(1200);
        product.setStock(5);

        System.out.println("Product: " +
product.getProductName());

        System.out.println("Price: $" + product.getPrice());

        System.out.println("Available: " + (product.isAvailable()
? "In Stock" : "Out of Stock"));
    }
}

```

## OUTPUT:

```
Product: Laptop  
Price: $1200.0  
Available: In Stock
```

iii)

### Code:

```
class House {  
    private String location;  
    private double area; // in square meters  
    private double pricePerSquareMeter;  
  
    public void setLocation(String location) {  
        this.location = location;  
    }  
  
    public void setArea(double area) {  
        this.area = area > 0 ? area : 0;  
    }  
  
    public void setPricePerSquareMeter(double price) {  
        this.pricePerSquareMeter = price > 0 ? price : 0;  
    }  
  
    public String getLocation() {
```



```

        return location;
    }

    public double getTotalPrice() {
        return area * pricePerSquareMeter;
    }
}

public class HouseTest {
    public static void main(String[] args) {
        House house = new House();
        house.setLocation("Downtown");
        house.setArea(200);
        house.setPricePerSquareMeter(1500);
        System.out.println("House Location: " +
house.getLocation());

        System.out.println("Total Price: $" +
house.getTotalPrice());
    }
}

```

## OUTPUT:

```

House Location: Downtow
Total Price: $300000.0

```

**iv)**

**Code:**

```
class GameCharacter {  
    private String name;  
    private int level;  
    private int strength;  
    private int intelligence;  
    public void setName(String name) {  
        this.name = name;  
    }  
    public void setLevel(int level) {  
        this.level = Math.max(level, 1);  
    }  
    public void setStrength(int strength) {  
        this.strength = Math.max(strength, 1);  
    }  
  
    public void setIntelligence(int intelligence) {  
        this.intelligence = Math.max(intelligence, 1);  
    }  
  
    public String getName() {  
        return name;  
    }  
}
```

```
        public int getPowerLevel() {  
            return (strength * 2 + intelligence * 3) * level;  
        }  
    }  
  
    public class GameTest {  
        public static void main(String[] args) {  
            GameCharacter hero = new GameCharacter();  
            hero.setName("Ketta Paiyyan");  
            hero.setLevel(10);  
            hero.setStrength(20);  
            hero.setIntelligence(15);  
            System.out.println("Hero: " + hero.getName());  
            System.out.println("Power Level: " +  
            hero.getPowerLevel());  
        }  
    }  
}
```

## OUTPUT:

```
Hero: Ketta Paiyyan  
Power Level: 850
```

# PACKAGES PROGRAMS

15

i)

**code:**

```
import java.awt.*;
```

```
import java.awt.event.*;
```

```
public class SimpleAWTApp {
```

```
    SimpleAWTApp() {
```

```
        Frame frame = new Frame("AWT Example");
```

```
        Button button = new Button("Click Me!");
```

```
        button.setBounds(50, 100, 80, 30);
```

```
        frame.add(button);
```

```
        frame.setSize(300, 200);
```

```
        frame.setLayout(null);
```

```
        frame.setVisible(true);
```

```
        frame.addWindowListener(new WindowAdapter() {
```

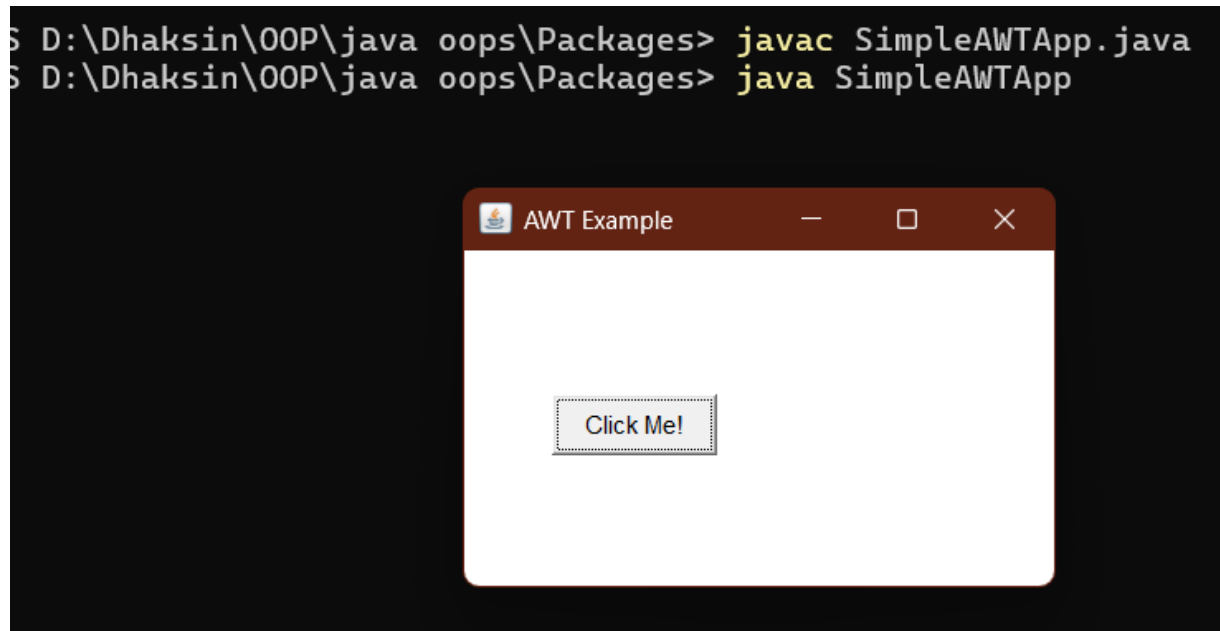
```
            public void windowClosing(WindowEvent e) {
```

```
                frame.dispose();
```

```
            }
```

```
    });  
}  
  
public static void main(String[] args) {  
    new SimpleAWTApp();  
}  
}
```

## output:



**ii)**

**code:**

```
#package  
package mypackage;  
  
public class basiccalc {  
    public int add(int a, int b) {  
        return a + b;  
    }  
    public int sub(int a, int b) {  
        return a - b;  
    }  
    public int mul(int a, int b) {  
        return a * b;  
    }  
    public int div(int a, int b) {  
        if (b == 0) {  
            System.out.println("Denominator with 0 is not defined  
or infinite");  
            return 0;  
        }  
        else {  
            return a / b;  
        }  
    }  
}
```

```

    }
}
import mypackage.basicalc;

public class calc {
    public static void main(String[] args) {
        basicalc c1=new basicalc();

        System.out.println("the sum is"+c1.add(2,3));
    }
}

```

## output:

```

PS D:\Dhaksin\OOP\java oops\Packages> javac -d . mypackage/basiccalc.java
PS D:\Dhaksin\OOP\java oops\Packages> javac calc.java
PS D:\Dhaksin\OOP\java oops\Packages> java calc
the sum is5

```

## iii)

### code:

```

        #package
package mypackage;

public class geometry {

    public double areaRectangle(double lenght,double breath){
        return lenght*breath;
    }
}

```

```

    }

    public double areaCircle(double radius){
        return 3.14*(radius*radius);
    }

    public double areaTriangle(double lenght,double height){
        return 0.5*(lenght+height);
    }
}

```

```

import java.lang.*;
import java.util.Scanner;
import mypackage.geometry;
public class geometerycal {
    public static void main(String[] args) {
        Scanner input=new Scanner(System.in);
        geometry a1=new geometry();
        System.out.println("Enter the lenght of Rectangle");
        double lenght=input.nextDouble();
        System.out.println("Enter the Breath of Rectangle");
        double breath=input.nextDouble();
        System.out.println("Area of Rectangle");
        System.out.println(a1.areaRectangle(lenght, breath));
        System.out.println("Enter the radius of Circle");
        double radius=input.nextDouble();
    }
}

```



```
System.out.println("Area of Circle");
System.out.println(a1.areaCircle(radius));
System.out.println("Enter lenght of triangle");
double tlength=input.nextDouble();
System.out.println("Enter height of triangle");
double height=input.nextDouble();
System.out.println("Area of Triangle");
System.out.println(a1.areaTriangle(tlength, height));
}
}
```

**output:**

```
Enter the lenght of Rectangle
5
Enter the Breath of Rectangle
3
Area of Rectangle
15.0
Enter the radius of Circle
3
Area of Circle
28.26
Enter lenght of triangle
3
Enter height of triangle
3
Area of Triangle
3.0
```

**iv)**

**code:**

```
import java.util.*;
```

```
import java.time.*;
```

```
import java.io.*;
```

```
public class EmployeePayroll {
```

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

```
        Scanner scanner = new Scanner(System.in);
```

```
        try {
```

```
            System.out.print("Enter number of employees: ");
```

```
            int numEmployees = scanner.nextInt();
```

```
            scanner.nextLine();
```

```
            ArrayList<Employee> employees = new  
ArrayList<>();
```

```
            for (int i = 0; i < numEmployees; i++) {
```

```
                System.out.println("\nEmployee " + (i + 1) + ":");
```

```
                System.out.print("Enter name: ");
```

```
                String name = scanner.nextLine();
```

```
System.out.print("Enter salary: ");  
double salary = scanner.nextDouble();
```

```
System.out.print("Enter joining year: ");  
int joiningYear = scanner.nextInt();  
scanner.nextLine();
```

```
employees.add(new Employee(name, salary,  
joiningYear));  
}
```

```
FileWriter writer = new  
FileWriter("EmployeePayroll.txt");
```

```
System.out.println("\n--- Employee Payroll ---");  
writer.write("--- Employee Payroll ---\n");
```

```
for (Employee emp : employees) {  
    emp.calculateBonus();  
    emp.displayDetails();  
    writer.write(emp.getDetailsForFile());  
}
```

```
writer.close();
```

```
        System.out.println("\nPayroll saved to  
'EmployeePayroll.txt'");
```

```
    } catch (IOException e) {
```

```
        System.out.println("An error occurred while writing to  
the file.");
```

```
    } catch (InputMismatchException e) {
```

```
        System.out.println("Invalid input. Please enter  
numbers correctly.");
```

```
    }
```

```
    scanner.close();
```

```
}
```

```
}
```

```
class Employee {
```

```
    private String name;
```

```
    private double salary;
```

```
    private int joiningYear;
```

```
    private double bonus;
```

```
    public Employee(String name, double salary, int  
joiningYear) {
```

```
        this.name = name;
```

```
        this.salary = salary;
```

```
this.joiningYear = joiningYear;
}

public void calculateBonus() {
    int currentYear = LocalDate.now().getYear();
    int yearsWorked = currentYear - joiningYear;

    if (yearsWorked >= 5) {
        bonus = salary * 0.1;
    } else {
        bonus = salary * 0.05;
    }
}

public void displayDetails() {
    System.out.println("Employee: " + name);
    System.out.println("Salary: ₹" + salary);
    System.out.println("Joining Year: " + joiningYear);
    System.out.println("Bonus: ₹" + bonus);
    System.out.println("Total Salary: ₹" + (salary + bonus));
    System.out.println();
}

public String getDetailsForFile() {
```

```
        return "Employee: " + name + "\n" +  
            "Salary: ₹" + salary + "\n" +  
            "Joining Year: " + joiningYear + "\n" +  
            "Bonus: ₹" + bonus + "\n" +  
            "Total Salary: ₹" + (salary + bonus) + "\n\n";  
    }  
}
```

## output:

```
Employee 1:  
Enter name: dhaksin  
Enter salary: 1200000  
Enter joining year: 2023  
  
Employee 2:  
Enter name: jyoo  
Enter salary: 1200400  
Enter joining year: 2024  
  
--- Employee Payroll ---  
Employee: dhaksin  
Salary: ?1200000.0  
Joining Year: 2023  
Bonus: ?60000.0  
Total Salary: ?1260000.0  
  
Employee: jyoo  
Salary: ?1200400.0  
Joining Year: 2024  
Bonus: ?60020.0  
Total Salary: ?1260420.0
```

# EXCEPTION HANDLING PROGRAMS

**16.**

**i)Division by 0**

**Code:**

```
public class main{  
    public static void main(String[] args){  
        try{  
            int result=10/0;  
        }  
        catch(ArithmeticException e){  
            System.out.println("Caught exception "+e);  
        }  
        finally {  
            System.out.println("This is the finally block. It runs  
always.");  
        }  
    }  
}
```

## Output:

```
PS D:\Dhaksin\OOP\java oops\Exception Handeling> javac main.java
PS D:\Dhaksin\OOP\java oops\Exception Handeling> java main
Caught exception java.lang.ArithmeticException: / by zero
This is the finally block. It runs always.
PS D:\Dhaksin\OOP\java oops\Exception Handeling> |
```

## ii)Age Exception with throws

### Code:

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

public class main2 {
    public static void main(String[] args){
        try{
            ageLimit(17);
        }
        catch(ageLimitException e){
            System.out.println("Caught error "+e);
        }
    }

    public static void ageLimit(int age) throws
    ageLimitException{
        if (age<18){
            throw new ageLimitException("Age must me above 18 ");
        }
    }
}
```



```
}  
}
```

## Output:

```
PS D:\Dhaksin\OOP\java oops\Exception Handeling> javac main2.java  
PS D:\Dhaksin\OOP\java oops\Exception Handeling> java main2  
Caught error ageLimitException: Age must me above 18  
PS D:\Dhaksin\OOP\java oops\Exception Handeling> |
```

## iii)Age Exception

### Code:

```
public class main3 {  
    public static void main(String[] args) {  
        try {  
            ageLimit(11);  
        }  
        catch (IllegalArgumentException e) {  
            System.out.println("Caught Error "+e);  
        }  
    }  
    public static void ageLimit(int age) {  
        if (age<18) {  
            throw new IllegalArgumentException("Not elligible to vote");  
        }  
        System.out.println("Successfully accessed");  
    }  
}
```

## Output:

```
PS D:\Dhaksin\OOP\java oops\Exception Handeling> javac main3.java
PS D:\Dhaksin\OOP\java oops\Exception Handeling> java main3
Caught Error java.lang.IllegalArgumentException: Not elligible to vote
PS D:\Dhaksin\OOP\java oops\Exception Handeling> |
```

### iv)Filemissing error

#### Code

```
import java.io.*;

class Filemissing extends Exception{

Filemissing(String message){

super(message);

}

}

public class main4{

public static void main(String[] args) throws Filemissing,
IOException {

String filename="example.txt";

try{

readfile("example.txt");

}

catch(FileNotFoundException e){

throw new Filemissing("File not found: " + filename);

}

}

}
```

```
public static void readfile(String filename) throws
Filemissing,IOException {
    FileReader file = new FileReader(filename);
    BufferedReader reader= new BufferedReader(file);
    String line=reader.readLine();
    while(line!=null){
        System.out.println(line);
        line=reader.readLine();
    }
    reader.close();
}
}
```

## Output:

```
Exception in thread "main" Filemissing: File not found: example.txt
    at main4.main(main4.java:15)
PS D:\Dhaksin\OOP\java oops\Exception Handeling> |
```

## FILE HANDLING PROGRAMS

**17.**

### **i)Reading a File**

**code:**

```
import java.io.BufferedReader;
import java.io.FileReader;
public class readfile {
    public static void main(String[] args) {
        try {
            FileReader r=new FileReader("output.txt");
            BufferedReader v=new BufferedReader(r);
            String e=v.readLine();
            while(e!=null){
                System.out.println(e);
                e=v.readLine();
            }
            v.close();
        } catch (Exception e) {
            System.out.println("error");
        }
    }
}
```

**Screen Shot:**

```
d
Karthick
okay
hii
PS D:\Code\Java>
```

## ii)Writing to a file:

### code:

```
import java.io.BufferedWriter;
import java.io.FileWriter;

public class filehandeling {
    public static void main(String[] args) {
        try{
            FileWriter file=new FileWriter("output.txt",true);
            BufferedWriter b=new BufferedWriter(file);
            b.write("dhaksin");
            b.newLine();
            b.write("hello");
            b.close();
        }
        catch(Exception e){
            System.out.println("error");
        }
    }
}
```

### Screen Shot:

```
dhaksin
hello
```

### **iii)Reading and Writing to a file with word count:**

#### **code:**

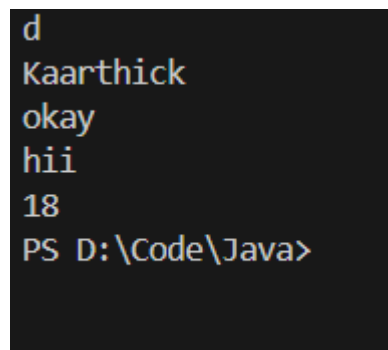
```
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
public class writeNread {
    public static void main(String[] args) {
        try (FileReader file = new FileReader("D:/Code/Java/input.txt");
            BufferedReader v = new BufferedReader(file);
            FileWriter A = new FileWriter("D://Code//Java//output.txt/");
            BufferedWriter f = new BufferedWriter(A)) {
            String line=v.readLine();
            int count=0;
            while (line!= null) {
                System.out.println(line);
                count+=line.length();
                f.write(line);
                f.newLine();
                line=v.readLine();
            }
            System.out.println(count);

        } catch (IOException e) {
```

```
        System.out.println("Some error: " + e.getMessage());
    }
}
}
```

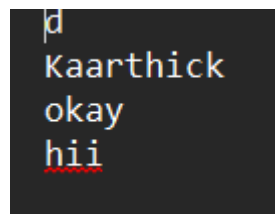
## Screen Shot:

### #Reading



```
d
Kaarthick
okay
hii
18
PS D:\Code\Java>
```

### #writing



```
d
Kaarthick
okay
hii
PS D:\Code\Java>
```

## iv)writing to file multiple try catch:

### code:

```
import java.io.*;

public class FileHandlingExample {
    public static void main(String[] args) {
        String fileName = "example.txt";
        try (FileWriter writer = new FileWriter(fileName)) {
            writer.write("Hello, this is a test file!\n");
            writer.write("This is line 2.");
            System.out.println("Successfully wrote to the file.");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

```

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

    try (BufferedReader reader = new BufferedReader(new
FileReader(fileName))) {
        System.out.println("\nFile contents:");
        String line;
        while ((line = reader.readLine()) != null) {
            System.out.println(line);
        }
    } catch (IOException e) {
        System.out.println("Error reading file: " + e.getMessage());
    }

    try (FileWriter writer = new FileWriter(fileName, true)) {
        writer.write("\nThis is an appended line.");
        System.out.println("\nSuccessfully appended to the file.");
    } catch (IOException e) {
        System.out.println("Error appending to file: " + e.getMessage());
    }

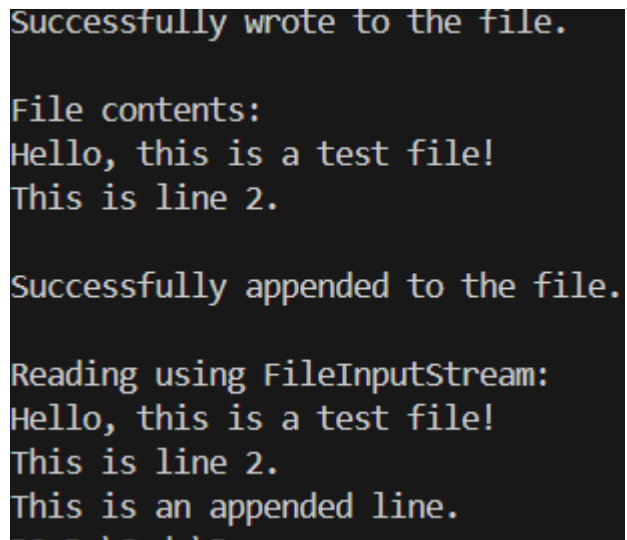
    try (FileInputStream fis = new FileInputStream(fileName)) {
        System.out.println("\nReading using FileInputStream:");
        int content;
        while ((content = fis.read()) != -1) {
            System.out.print((char) content);
        }
    } catch (IOException e) {
        System.out.println("Error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

```



```
}
```

## Screen Shot:



```
Successfully wrote to the file.  
  
File contents:  
Hello, this is a test file!  
This is line 2.  
  
Successfully appended to the file.  
  
Reading using FileInputStream:  
Hello, this is a test file!  
This is line 2.  
This is an appended line.  
test.txt
```