

**Upon successful completion of this course, students should be able to:**

CO1: Develop simple Java Programs with arrays, operators and control statements. (Target 50 )

CO2: Construct programs featuring Classes, Methods, Object creation and initialization. (Target 50 )

CO3: Implement Object oriented features like Abstraction, Inheritance & Polymorphism (Target 50 )

CO4: Handle Exceptions and perform IO operations (Target 50 )

CO5: Develop GUIs using frameworks like AWT, SWING and JAVA FX (Target 50 ) ·

CO6: Develop programs with multiple threads and address concurrency issues (Target 50)

Sl No	Question	CO
1	Sum of Array Elements: Write a program that calculates and prints the sum of all elements in an integer array	CO1
	<pre>public class ArraySum {     public static void main(String[] args) {         int[] numbers = { 1, 2, 3, 4, 5 };         int sum = 0;          for (int i = 0; i &lt; numbers.length; i++) {             sum += numbers[i];         }          System.out.println("Sum: " + sum);     } }</pre>	
2	Finding Maximum Element: Write a program that finds and prints the maximum element in an array of integers.	CO1
	<pre>public class MaxElement {     public static void main(String[] args) {         int[] numbers = { 12, 45, 67, 23, 9 };         int max = numbers[0];          for (int i = 1; i &lt; numbers.length; i++) {</pre>	CO1

	<pre>                 if (numbers[i] &gt; max) {                     max = numbers[i];                 }             }              System.out.println("Maximum element: " + max);         }     } </pre>	
3	Reverse an Array: Write a program that reverses the order of elements in an array and prints the reversed array.	CO1
	<pre> public class ReverseArray {     public static void main(String[] args) {         int[] numbers = { 1, 2, 3, 4, 5 };          System.out.println("Original array:");         for (int i = 0; i &lt; numbers.length; i++) {             System.out.print(numbers[i] + " ");         }          System.out.println("\nReversed array:");         for (int i = numbers.length - 1; i &gt;= 0; i--) {             System.out.print(numbers[i] + " ");         }     } } </pre>	CO1
4	Counting Even Numbers: Write a program that counts the number of even elements in an integer array and prints the count	CO1
	<pre> public class CountEven {     public static void main(String[] args) {         int[] numbers = { 2, 5, 8, 10, 13, 15 };         int count = 0;          for (int i = 0; i &lt; numbers.length; i++) {             if (numbers[i] % 2 == 0) {                 count++;             }         }     } } </pre>	CO1

	<pre>         }          System.out.println("Count of even numbers: " + count);     } } </pre>	
5	Write a Java program that takes an integer as input and determines if it is positive, negative, or zero using the if construct.	CO1
	<pre> import java.util.Scanner;  public class CheckNumber {     public static void main(String[] args) {         Scanner scanner = new Scanner(System.in);         System.out.print("Enter an integer: ");         int number = scanner.nextInt();          if (number &gt; 0) {             System.out.println("The number is positive.");         } else if (number &lt; 0) {             System.out.println("The number is negative.");         } else {             System.out.println("The number is zero.");         }     } } </pre>	CO1
6	Write a Java program that determines if a given year is a leap year using the if construct	CO1
	<pre> 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 year = scanner.nextInt();          if ((year % 4 == 0 &amp;&amp; year % 100 != 0)    (year % 400 == 0)) {             System.out.println(year + " is a leap year.");         } else {             System.out.println(year + " is not a leap year.");         }     } } </pre>	CO1
7	Write a Java program that takes two integers as input and finds the maximum using the if construct.	CO1


	<pre> import java.util.Scanner;  public class FindMax {     public static void main(String[] args) {         Scanner scanner = new Scanner(System.in);         System.out.print("Enter the first integer: ");         int num1 = scanner.nextInt();         System.out.print("Enter the second integer: ");         int num2 = scanner.nextInt();          if (num1 &gt; num2) {             System.out.println("The maximum is: " + num1);         } else {             System.out.println("The maximum is: " + num2);         }     } } </pre>	
8	Write a Java program that takes a day number (1-7) as input and prints the corresponding day of the week using a switch case.	CO1
	<pre> import java.util.Scanner;  public class DayOfWeek {     public static void main(String[] args) {         Scanner scanner = new Scanner(System.in);         System.out.print("Enter a day number (1-7): ");         int day = scanner.nextInt();          switch (day) {             case 1:                 System.out.println("Sunday");                 break;             case 2:                 System.out.println("Monday");                 break;             case 3:                 System.out.println("Tuesday");                 break;             case 4:                 System.out.println("Wednesday");                 break;             case 5:                 System.out.println("Thursday");                 break;         }     } } </pre>	CO1


	<pre>         case 6:             System.out.println("Friday");             break;         case 7:             System.out.println("Saturday");             break;         default:             System.out.println("Invalid day number.");     } } </pre>	
9	Write a Java program that takes a grade (A, B, C, D, or F) as input and prints a message based on the grade using a switch case.	CO1
	<pre> import java.util.Scanner;  public class GradeMessage {     public static void main(String[] args) {         Scanner scanner = new Scanner(System.in);         System.out.print("Enter your grade (A, B, C, D, or F): ");         char grade = scanner.next().charAt(0);          switch (grade) {             case 'A':                 System.out.println("Excellent! Keep up the good work.");                 break;             case 'B':                 System.out.println("Good job! Keep improving.");                 break;             case 'C':                 System.out.println("Fair. Work harder for better results.");                 break;             case 'D':                 System.out.println("You need to study more.");                 break;             case 'F':                 System.out.println("You have failed. Seek help and work harder.");                 break;             default:                 System.out.println("Invalid grade.");         }     } } </pre>	CO1

10	Write a Java program that takes three integers as input and checks if they can form a triangle using the logical AND operator.	CO1
	<pre> import java.util.Scanner;  public class TriangleCheck {     public static void main(String[] args) {         Scanner scanner = new Scanner(System.in);         System.out.print("Enter three integers: ");         int a = scanner.nextInt();         int b = scanner.nextInt();         int c = scanner.nextInt();          if ((a + b &gt; c) &amp;&amp; (a + c &gt; b) &amp;&amp; (b + c &gt; a)) {             System.out.println("These integers can form a triangle.");         } else {             System.out.println("These integers cannot form a triangle.");         }     } } </pre>	
11	Write a Java program that takes an age as input and determines if a person is eligible to vote using the logical OR operator.	CO1
	<pre> import java.util.Scanner;  public class EligibilityCheck {     public static void main(String[] args) {         Scanner scanner = new Scanner(System.in);         System.out.print("Enter your age: ");         int age = scanner.nextInt();          if (age &gt;= 18    age &gt;= 16) {             System.out.println("You are eligible to vote.");         } else {             System.out.println("You are not eligible to vote yet.");         }     } } </pre>	
12	Write a Java program that takes a number as input and checks if it is both even and a multiple of 3 using logical AND and modulus operator.	CO1
	<pre> import java.util.Scanner;  public class EvenAndMultipleOf3 {     public static void main(String[] args) { </pre>	

	<pre> Scanner scanner = new Scanner(System.in); System.out.print("Enter a number: "); int number = scanner.nextInt();  if (number % 2 == 0 &amp;&amp; number % 3 == 0) {     System.out.println(number + " is both even and a multiple of 3."); } else {     System.out.println(number + " is not both even and a multiple of 3."); } } </pre>	

For Complete listing please use the following two links

 List 1 Simple problems using the fundamental syntax and semantics of Java

 common operations and concepts related to arrays in Java.

**CO2: Construct programs featuring Classes, Methods, Object creation and initialization.**

 **Simple Classes without constructors**

<b>1</b>	Create a Java class named Student with attributes for student name and age. Implement methods to set and display these attributes.	CO2
	<pre> class Student {     private String name;     private int age;      public void setName(String name) {         this.name = name;     }      public void setAge(int age) {         this.age = age;     } } </pre>	

	<pre>         public void displayInfo() {             System.out.println("Name: " + name);             System.out.println("Age: " + age);         }     }      public class StudentDemo {         public static void main(String[] args) {             Student student1 = new Student();             student1.setName("Alice");             student1.setAge(20);             student1.displayInfo();         }     } </pre>	
<b>2</b>	Extend the Student class to include a method to calculate and display the grade based on marks obtained. Assume a pass mark of 50.	CO2
	<pre> class Student {     // existing code      public void displayGrade(int marks) {         if (marks &gt;= 50) {             System.out.println("Grade: Pass");         } else {             System.out.println("Grade: Fail");         }     } }  public class StudentDemo {     public static void main(String[] args) {         Student student1 = new Student();         student1.displayGrade(65);     } } </pre>	
<b>3</b>	Create a class Rectangle with attributes for length and width. Implement a method to calculate and display the area of the rectangle	CO2
	<pre> class Rectangle {     private double length;     private double width;      public void setDimensions(double length, double width) {         this.length = length;     } } </pre>	



	<pre>         this.width = width;     }      public void calculateArea() {         double area = length * width;         System.out.println("Area of the rectangle: " + area);     } }  public class RectangleDemo {     public static void main(String[] args) {         Rectangle rectangle1 = new Rectangle();         rectangle1.setDimensions(5.0, 3.0);         rectangle1.calculateArea();     } } </pre>	
<b>4</b>	Write a method in Java to calculate the factorial of a given number. Test the method with appropriate inputs.	CO2
	<pre> class FactorialCalculator {     public static int calculateFactorial(int n) {         if (n == 0    n == 1)             return 1;         else             return n * calculateFactorial(n - 1);     } }  public class FactorialDemo {     public static void main(String[] args) {         int num = 5;         int factorial = FactorialCalculator.calculateFactorial(num);         System.out.println("Factorial of " + num + " is: " + factorial);     } } </pre>	
<b>5</b>	Write a method to check if a number is prime or not. Return a boolean value. Test the method with different numbers.	CO2
	<pre> class PrimeChecker {     public static boolean isPrime(int n) {         if (n &lt;= 1)             return false;         for (int i = 2; i &lt;= Math.sqrt(n); i++) {             if (n % i == 0)                 return false;         }         return true;     } } </pre>	

	<pre>         }         return true;     } }  public class PrimeDemo {     public static void main(String[] args) {         int num = 17;         boolean isPrime = PrimeChecker.isPrime(num);         System.out.println(num + " is prime: " + isPrime);     } } </pre>	
<b>6</b>	Write a method to reverse a given string. Return the reversed string. Test the method with different strings.	CO2
	<pre> class StringReverser {     public static String reverseString(String input) {         StringBuilder reversed = new StringBuilder();         for (int i = input.length() - 1; i &gt;= 0; i--) {             reversed.append(input.charAt(i));         }         return reversed.toString();     } }  public class StringReverseDemo {     public static void main(String[] args) {         String original = "hello";         String reversed = StringReverser.reverseString(original);         System.out.println("Original: " + original);         System.out.println("Reversed: " + reversed);     } } </pre>	
<b>7</b>	Create a class Book without a constructor. Create an object and initialize its attributes separately.	CO2
	<pre> class Book {     String title;     String author;      public void displayBookInfo() {         System.out.println("Title: " + title);         System.out.println("Author: " + author);     } } </pre>	

	<pre> public class BookDemo {     public static void main(String[] args) {         Book book1 = new Book();         book1.title = "Java Programming";         book1.author = "John Doe";         book1.displayBookInfo();     } } </pre>	
<b>8</b>	Create a class Person without a constructor. Create an object and initialize its attributes using setter methods.	CO2
	<pre> class Person {     String name;     int age;      public void setName(String name) {         this.name = name;     }      public void setAge(int age) {         this.age = age;     } }  public class PersonDemo {     public static void main(String[] args) {         Person person1 = new Person();         person1.setName("Alice");         person1.setAge(25);         System.out.println("Name: " + person1.name);         System.out.println("Age: " + person1.age);     } } </pre>	
<b>9</b>	Create a class Car without a constructor. Create an object and initialize its attributes directly.	CO2
	<pre> class Car {     String make;     String model;      public void displayCarInfo() {         System.out.println("Make: " + make);         System.out.println("Model: " + model);     } } </pre>	

	<pre> public class CarDemo {     public static void main(String[] args) {         Car car1 = new Car();         car1.make = "Toyota";         car1.model = "Corolla";         car1.displayCarInfo();     } } </pre>	
--	---	--

### CO3: Implement Object oriented features like Abstraction, Inheritance & Polymorphism

1	<p>Create a class <b>Person</b> with properties <b>name</b> and <b>age</b>. Create a subclass <b>Employee</b> that inherits from <b>Person</b> and adds a property <b>employeeId</b>. Display the complete information of an employee using the <b>super</b> keyword.</p> <p>Modify the above class by adding a method <b>displayInfo()</b> in both the <b>Person</b> and <b>Employee</b> classes. Use the <b>super</b> keyword to call the <b>displayInfo()</b> method of the <b>Person</b> class from the <b>Employee</b> class</p>	CO3
	<pre> class Person {      String name;      int age;      public Person(String name, int age) {          this.name = name;          this.age = age;      }      public void displayInfo() { </pre>	CO3

	<pre>        System.out.println("Name: " + name);          System.out.println("Age: " + age);      }  }  class Employee extends Person {      String employeeId;      public Employee(String name, int age, String employeeId) {          super(name, age);          this.employeeId = employeeId;      }      @Override      public void displayInfo() {          super.displayInfo();          System.out.println("Employee ID: " + employeeId);          System.out.println("Complete Information using super keyword:");          super.displayInfo();      }  }  public class PersonDemo {      public static void main(String[] args) {          Employee employee = new Employee("John Doe", 30, "E12345");</pre>	
--	--	--

	<pre>         employee.displayInfo();      }  }</pre>	
2	<p>Create a class <b>Vehicle</b> with properties <b>brand</b> and <b>model</b>. Create subclasses <b>Car</b> and <b>Motorcycle</b> that inherit from <b>Vehicle</b>. Add properties specific to each subclass. Display information about a car and a motorcycle.</p> <p>Modify the above class by adding a method <b>startEngine()</b> in the <b>Vehicle</b> class and override it in the subclasses. Display the engine start message for each vehicle type</p>	CO3
	<pre> class Vehicle {      String brand;      String model;      public Vehicle(String brand, String model) {          this.brand = brand;          this.model = model;      }      public void displayInfo() {          System.out.println("Brand: " + brand);          System.out.println("Model: " + model);      }      public void startEngine() {          System.out.println("Starting the engine of the vehicle.");      }  }</pre>	CO3

	<pre>    }  }  class Car extends Vehicle {     int numDoors;      public Car(String brand, String model, int numDoors) {         super(brand, model);         this.numDoors = numDoors;     }      @Override     public void displayInfo() {         super.displayInfo();         System.out.println("Number of doors: " + numDoors);     }      @Override     public void startEngine() {         System.out.println("Starting the car's engine.");     } }  class Motorcycle extends Vehicle {     String type;</pre>	
--	---	--

	<pre>public Motorcycle(String brand, String model, String type) {     super(brand, model);     this.type = type; }  @Override public void displayInfo() {     super.displayInfo();     System.out.println("Type: " + type); }  @Override public void startEngine() {     System.out.println("Starting the motorcycle's engine."); } }  public class VehicleDemo {     public static void main(String[] args) {         Car car = new Car("Toyota", "Corolla", 4);         System.out.println("Car information:");         car.displayInfo();         car.startEngine();     } }</pre>	
--	---	--



	<pre> System.out.println();          Motorcycle motorcycle = new Motorcycle("Honda", "CBR650R", "Sports");          System.out.println("Motorcycle information:");          motorcycle.displayInfo();          motorcycle.startEngine();      }  } </pre>	
<b>3</b>	<p>Create an abstract class called Shape with a method calculateArea(). Implement two subclasses Circle and Rectangle which extend Shape and override calculateArea() to calculate the area for a circle and rectangle respectively.</p>	CO3
	<pre> abstract class Shape {      abstract double calculateArea();  }  class Circle extends Shape {      private double radius;      public Circle(double radius) {          this.radius = radius;      }      @Override      double calculateArea() { </pre>	

	<pre>        return Math.PI * radius * radius;     } }  class Rectangle extends Shape {      private double width;      private double height;      public Rectangle(double width, double height) {          this.width = width;          this.height = height;      }      @Override      double calculateArea() {          return width * height;      }  }  public class ShapeDemo {      public static void main(String[] args) {          Circle circle = new Circle(5);          System.out.println("Area of Circle: " + circle.calculateArea());          Rectangle rectangle = new Rectangle(4, 6);</pre>	
--	---	--

	<pre>         System.out.println("Area of Rectangle: " + rectangle.calculateArea());      }  }</pre>	
4	<p>Create a base class Animal with an abstract method makeSound(). Implement two subclasses Dog and Cat which extend Animal and override makeSound() to print "Bark" for dogs and "Meow" for cats.</p>	CO3
	<pre> abstract class Animal {      abstract void makeSound();  }  class Dog extends Animal {      @Override      void makeSound() {          System.out.println("Bark");      }  }  class Cat extends Animal {      @Override      void makeSound() {          System.out.println("Meow");      }  }  public class AnimalDemo {</pre>	CO3

	<pre> public static void main(String[] args) {      Animal dog = new Dog();      dog.makeSound();      Animal cat = new Cat();      cat.makeSound();  } } </pre>	
<b>5</b>	Create an interface Shape with a method draw(). Implement two classes Circle and Square which implement Shape and override draw() to print "Drawing Circle" and "Drawing Square" respectively.	CO3
	<pre> interface Shape {      void draw();  }  class Circle implements Shape {      @Override      public void draw() {          System.out.println("Drawing Circle");      }  }  class Square implements Shape {      @Override      public void draw() { </pre>	CO3

	<pre>         System.out.println("Drawing Square");     } }  public class ShapeDemo {     public static void main(String[] args) {         Shape circle = new Circle();         circle.draw();          Shape square = new Square();         square.draw();     } } </pre>	
<b>6</b>	<p>Create a base class Vehicle with an abstract method startEngine(). Implement two subclasses Car and Bike which extend Vehicle and override startEngine() to print "Car Engine Started" and "Bike Engine Started" respectively.</p>	CO3
	<pre> abstract class Vehicle {     abstract void startEngine(); }  class Car extends Vehicle {     @Override     void startEngine() {         System.out.println("Car Engine Started");     } } </pre>	

	<pre> }  class Bike extends Vehicle {      @Override      void startEngine() {          System.out.println("Bike Engine Started");      }  }  public class VehicleDemo {      public static void main(String[] args) {          Vehicle car = new Car();          car.startEngine();          Vehicle bike = new Bike();          bike.startEngine();      }  } </pre>	
7	Create an abstract class Employee with properties name and salary. Implement two subclasses Manager and Developer which extend Employee and override a method calculateBonus() to calculate bonus for manager as 20% of salary and for developer as 10% of salary	CO3
	<pre> abstract class Employee {      protected String name;      protected double salary; </pre>	

	<pre>public Employee(String name, double salary) {      this.name = name;      this.salary = salary;  }      abstract double calculateBonus();  }  class Manager extends Employee {      public Manager(String name, double salary) {          super(name, salary);      }      @Override      double calculateBonus() {          return 0.2 * salary;      }  }  class Developer extends Employee {      public Developer(String name, double salary) {          super(name, salary);      }      @Override</pre>	
--	--	--

	<pre> double calculateBonus() {      return 0.1 * salary;  }  }  public class EmployeeDemo {      public static void main(String[] args) {          Employee manager = new Manager("John Doe", 5000);          System.out.println("Manager Bonus: \$" + manager.calculateBonus());          Employee developer = new Developer("Alice Smith", 4000);          System.out.println("Developer Bonus: \$" + developer.calculateBonus());      }  } </pre>	
8	<p>Create an abstract class Bank with a method calculateInterest(). Implement two subclasses SavingsAccount and FixedDeposit which extend Bank and override calculateInterest() to calculate interest for savings account as 5% and for fixed deposit as 7%.</p>	CO3
	<pre> abstract class Bank {      abstract double calculateInterest();  }  class SavingsAccount extends Bank {      @Override      double calculateInterest() { </pre>	



	<pre>         return 0.05;     } }  class FixedDeposit extends Bank {      @Override      double calculateInterest() {          return 0.07;      }  }  public class BankDemo {      public static void main(String[] args) {          Bank savingsAccount = new SavingsAccount();          System.out.println("Savings Account Interest Rate: " + savingsAccount.calculateInterest() * 100 + "%");          Bank fixedDeposit = new FixedDeposit();          System.out.println("Fixed Deposit Interest Rate: " + fixedDeposit.calculateInterest() * 100 + "%");      }  } </pre>	
<b>9</b>	Create a base class Fruit with an abstract method taste(). Implement two subclasses Apple and Orange which extend Fruit and override taste() to print "Sweet" for apples and "Sour" for oranges.	CO3
	<pre> abstract class Fruit { </pre>	

	<pre>        abstract void taste();     }      class Apple extends Fruit {         @Override         void taste() {             System.out.println("Sweet");         }     }      class Orange extends Fruit {         @Override         void taste() {             System.out.println("Sour");         }     }      public class FruitDemo {         public static void main(String[] args) {             Fruit apple = new Apple();             apple.taste();              Fruit orange = new Orange();             orange.taste();         }     }</pre>	
--	--	--

	}	
10	Create an interface Animal with a method sound(). Implement two classes Dog and Cat which implement Animal and override sound() to print "Bark" for dogs and "Meow" for cats	CO3
	<pre> interface Animal {      void sound();  }  class Dog implements Animal {      @Override      public void sound() {          System.out.println("Bark");      }  }  class Cat implements Animal {      @Override      public void sound() {          System.out.println("Meow");      }  }  public class AnimalDemo {      public static void main(String[] args) {          Animal dog = new Dog(); </pre>	

	<pre> dog.sound();  Animal cat = new Cat();  cat.sound();  }  } </pre>	
11	<p>Create an abstract class Device with an abstract method turnOn(). Implement two subclasses Laptop and MobilePhone which extend Device and override turnOn() to print "Laptop turned on" and "Mobile phone turned on" respectively.</p>	CO3
	<pre> abstract class Device {      abstract void turnOn();  }  class Laptop extends Device {      @Override      void turnOn() {          System.out.println("Laptop turned on");      }  }  class MobilePhone extends Device {      @Override      void turnOn() {          System.out.println("Mobile phone turned on");      }  } </pre>	

	<pre> }  public class DeviceDemo {      public static void main(String[] args) {          Device laptop = new Laptop();          laptop.turnOn();          Device mobilePhone = new MobilePhone();          mobilePhone.turnOn();      }  } </pre>	
<b>12</b>	Create an interface Shape with a method draw(). Implement two classes Circle and Rectangle which implement Shape and override draw() to print "Drawing Circle" and "Drawing Rectangle" respectively.	CO3
	<pre> interface Shape {      void draw();  }  class Circle implements Shape {      @Override      public void draw() {          System.out.println("Drawing Circle");      }  } </pre>	

	<pre> class Rectangle implements Shape {      @Override      public void draw() {          System.out.println("Drawing Rectangle");      }  }  public class ShapeDemo {      public static void main(String[] args) {          Shape circle = new Circle();          circle.draw();          Shape rectangle = new Rectangle();          rectangle.draw();      }  } </pre>	
--	---	--

#### CO4: Handle Exceptions and perform IO operations

1	Write a Java program to read an integer from the user and display its square. Handle the InputMismatchException if the user enters a non-integer value	CO4
	<pre> import java.util.InputMismatchException; import java.util.Scanner;  public class SquareCalculator {     public static void main(String[] args) {         Scanner scanner = new Scanner(System.in);         try {             System.out.print("Enter an integer: ");             int num = scanner.nextInt();             int square = num * num; </pre>	CO4

	<pre>         System.out.println("Square of " + num + " is: " + square);     } catch (InputMismatchException e) {         System.out.println("Invalid input. Please enter an integer.");     } } </pre>	
2	Write a Java program to read a file and display its content. Handle the <code>FileNotFoundException</code> if the file does not exist.	CO4
	<pre> import java.io.BufferedReader; import java.io.FileNotFoundException; import java.io.FileReader; import java.io.IOException;  public class FileContentReader {     public static void main(String[] args) {         try {             BufferedReader reader = new BufferedReader(new FileReader("example.txt"));             String line;             while ((line = reader.readLine()) != null) {                 System.out.println(line);             }             reader.close();         } catch (FileNotFoundException e) {             System.out.println("File not found.");         } catch (IOException e) {             System.out.println("Error reading the file.");         }     } } </pre>	
3	Write a Java program to read an array of integers from the user and display the sum of all elements. Handle the <code>ArrayIndexOutOfBoundsException</code> if the array index is out of bounds.	CO4
	<pre> import java.util.InputMismatchException; import java.util.Scanner;  public class ArraySumCalculator {     public static void main(String[] args) {         Scanner scanner = new Scanner(System.in);         try {             System.out.print("Enter the number of elements: ");             int n = scanner.nextInt();             int[] arr = new int[n];             System.out.println("Enter the elements:");             for (int i = 0; i &lt; n; i++) { </pre>	

	<pre> arr[i] = scanner.nextInt(); } int sum = 0; for (int i = 0; i &lt; n; i++) {     sum += arr[i]; } System.out.println("Sum of array elements: " + sum); } catch (InputMismatchException e) {     System.out.println("Invalid input. Please enter an integer."); } catch (ArrayIndexOutOfBoundsException e) {     System.out.println("Array index out of bounds."); } } } </pre>	
4	Write a Java program to read a file and display its content line by line. Handle any exception that may occur.	CO4
	<pre> import java.io.BufferedReader; import java.io.FileReader; import java.io.IOException;  public class FileLineReader {     public static void main(String[] args) {         try {             BufferedReader reader = new BufferedReader(new FileReader("example.txt"));             String line;             while ((line = reader.readLine()) != null) {                 System.out.println(line);             }             reader.close();         } catch (Exception e) {             System.out.println("An error occurred: " + e.getMessage());         }     } } </pre>	
5	Write a Java program to read an integer from the user and display its reciprocal. Handle the <b>ArithmeticException</b> if the user enters zero.	CO4
	<pre> import java.util.InputMismatchException; import java.util.Scanner;  public class ReciprocalCalculator {     public static void main(String[] args) {         Scanner scanner = new Scanner(System.in);         try {             System.out.print("Enter an integer: ");             int num = scanner.nextInt(); </pre>	



	<pre>         if (num == 0) {             throw new ArithmeticException("Cannot calculate reciprocal of zero.");         }         double reciprocal = 1.0 / num;         System.out.println("Reciprocal of " + num + " is: " + reciprocal);     } catch (InputMismatchException e) {         System.out.println("Invalid input. Please enter an integer.");     } catch (ArithmeticException e) {         System.out.println("Arithmetic error: " + e.getMessage());     } } </pre>	
6	Write a Java program to read a file and display its content. If the file does not exist, display a custom error message.	CO4
	<pre> import java.io.BufferedReader; import java.io.FileNotFoundException; import java.io.FileReader; import java.io.IOException;  public class FileContentReader {     public static void main(String[] args) {         try {             BufferedReader reader = new BufferedReader(new FileReader("example.txt"));             String line;             while ((line = reader.readLine()) != null) {                 System.out.println(line);             }             reader.close();         } catch (FileNotFoundException e) {             System.out.println("File not found: " + e.getMessage());         } catch (IOException e) {             System.out.println("Error reading the file: " + e.getMessage());         }     } } </pre>	

**CO6: Develop programs with multiple threads and address concurrency issues  
(Target 50)**

1	Write a Java program to create and start two threads that print numbers from 1 to 5. Ensure that the threads execute concurrently and do not interfere with each other.	CO6
	<pre> class NumberPrinter extends Thread {     private int start;     private int end;      public NumberPrinter(int start, int end) {         this.start = start;         this.end = end;     }      public void run() {         for (int i = start; i &lt;= end; i++) {             System.out.println(i);         }     } }  public class NumberPrinterDemo {     public static void main(String[] args) {         NumberPrinter thread1 = new NumberPrinter(1, 5);         NumberPrinter thread2 = new NumberPrinter(6, 10);          thread1.start();         thread2.start();     } } </pre>	
2	Write a Java program to create and start three threads that print "Hello World" five times. Ensure that each thread prints a unique identifier along with "Hello World".	CO6
	<pre> class HelloWorldPrinter extends Thread {     private String threadId;      public HelloWorldPrinter(String threadId) {         this.threadId = threadId;     }      public void run() {         for (int i = 1; i &lt;= 5; i++) {             System.out.println("Thread " + threadId + ": Hello World");         }     } } </pre>	

	<pre> public class HelloWorldPrinterDemo {     public static void main(String[] args) {         HelloWorldPrinter thread1 = new HelloWorldPrinter("Thread 1");         HelloWorldPrinter thread2 = new HelloWorldPrinter("Thread 2");         HelloWorldPrinter thread3 = new HelloWorldPrinter("Thread 3");          thread1.start();         thread2.start();         thread3.start();     } } </pre>	
<b>3</b>	Write a Java program to create and start two threads that print even and odd numbers from 1 to 20 respectively. Ensure that each thread prints numbers in order.	CO6
	<pre> class NumberPrinter extends Thread {     private int start;     private int end;      public NumberPrinter(int start, int end) {         this.start = start;         this.end = end;     }      public void run() {         for (int i = start; i &lt;= end; i += 2) {             System.out.println(i);         }     } }  public class EvenOddPrinterDemo {     public static void main(String[] args) {         NumberPrinter thread1 = new NumberPrinter(2, 20); // Even         numbers         NumberPrinter thread2 = new NumberPrinter(1, 19); // Odd         numbers          thread1.start();         thread2.start();     } } </pre>	
<b>4</b>	Write a Java program to create and start two threads that alternate printing "Ping" and "Pong" for a specified number of times	CO6
	<pre> class PingPongPrinter extends Thread {     private String message;     private int times; </pre>	

	<pre> public PingPongPrinter(String message, int times) {     this.message = message;     this.times = times; }  public void run() {     for (int i = 0; i &lt; times; i++) {         System.out.println(message);     } }  public class PingPongPrinterDemo {     public static void main(String[] args) {         int numberOfTimes = 5;          PingPongPrinter pingPrinter = new PingPongPrinter("Ping",         numberOfTimes);         PingPongPrinter pongPrinter = new PingPongPrinter("Pong",         numberOfTimes);          pingPrinter.start();         pongPrinter.start();     } } </pre>	
<b>5</b>	Write a Java program to create and start three threads that display "A", "B", and "C" respectively in a sequence. Ensure that each thread executes its task in order.	<b>CO6</b>
	<pre> class AlphabetPrinter extends Thread {     private String letter;      public AlphabetPrinter(String letter) {         this.letter = letter;     }      public void run() {         System.out.println(letter);     } }  public class AlphabetPrinterDemo {     public static void main(String[] args) {         AlphabetPrinter thread1 = new AlphabetPrinter("A");         AlphabetPrinter thread2 = new AlphabetPrinter("B");         AlphabetPrinter thread3 = new AlphabetPrinter("C");          thread1.start();         thread2.start();         thread3.start();     } } </pre>	