	NAGARJUNA COLLEGE OF ENGINEERING AND TECHNOLOGY (An autonomous college under VTU) DEPARTMENT OF CSE (DATA SCIENCE)

BE 3rd Sem

Object Oriented Programming Using Java (IC)

Lab Programs

1)

A. Develop a Java program for an advanced arithmetic calculator that takes two integer operands and an operator from the user. The program should be capable of performing addition, subtraction, multiplication, and division. Ensure that the program handles input validation, including checking for the validity of the operator and non-negative values for the operands. After each calculation, ask the user if they want to perform another operation and provide a history of previous calculations upon request.

```
import java.util.Scanner;

public class AdvancedCalculator {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        char operator;
        boolean continueCalculation = true;
        String history = "";

        while (continueCalculation) {
            System.out.print("Enter the first operand: ");
            int operand1 = scanner.nextInt();

            System.out.print("Enter the operator (+, -, *, /): ");
            operator = scanner.next().charAt(0);

            System.out.print("Enter the second operand: ");
            int operand2 = scanner.nextInt();

            int result = 0;
```

```
switch (operator) {
    case '+':
        result = operand1 + operand2;
        break;
    case '-':
        result = operand1 - operand2;
        break;
    case '*':
        result = operand1 * operand2;
        break;
    case '/':
        if (operand2 != 0) {
            result = operand1 / operand2;
        } else {
            System.out.println("Error: Division by zero.");
            continue;
        }
        break;
    default:
        System.out.println("Error: Invalid operator.");
        continue;
}

history += operand1 + " " + operator + " " + operand2 + " = " + result + "\n";
System.out.println("Result: " + result);

System.out.print("Do you want to perform another calculation? (yes/no): ");
String choice = scanner.next();
continueCalculation = choice.equalsIgnoreCase("yes") || 
choice.equalsIgnoreCase("y");
```

```
        }

        if (!history.isEmpty()) {
            System.out.println("\nCalculation History:");
            System.out.println(history);
        }

        System.out.println("Calculator terminated.");
        scanner.close();
    }
}
```

B. Write a Java program to generate the first 'n' terms of the Fibonacci series and determine the following:

1. Calculate the sum of all even terms in the series.
2. Find the product of all odd terms in the series.
3. Check and display the largest prime number within the series.
4. Calculate the average of the entire series.

```
import java.util.Scanner;

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

        System.out.print("Enter the number of terms (n) for the Fibonacci series: ");
        int n = scanner.nextInt();

        long firstTerm = 0;
        long secondTerm = 1;
        long evenSum = 0;
        long oddProduct = 1;
        long largestPrime = 0;
long sum = 1;

System.out.print("Fibonacci Series: " + firstTerm + " " + secondTerm);

for (int i = 2; i < n; i++) {
    long nextTerm = firstTerm + secondTerm;
    sum += nextTerm;
    System.out.print(" " + nextTerm);

    if (nextTerm % 2 == 0) {
        evenSum += nextTerm;
    } else {
        oddProduct *= nextTerm;
    }

    if (isPrime(nextTerm) && nextTerm > largestPrime) {
        largestPrime = nextTerm;
    }

    firstTerm = secondTerm;
    secondTerm = nextTerm;
}

double average = (double) sum / n;

System.out.println("\nSum of even terms: " + evenSum);
System.out.println("Product of odd terms: " + oddProduct);
if (largestPrime == 0) {
    System.out.println("No prime numbers found in the series.");
} else {
    System.out.println("Largest prime number in the series: " + largestPrime);
```

```
            }
            System.out.println("Average of the series: " + average);

            scanner.close();
        }

        // Function to check if a number is prime
        private static boolean isPrime(long num) {
            if (num <= 1) {
                return false;
            }
            if (num <= 3) {
                return true;
            }
            if (num % 2 == 0 || num % 3 == 0) {
                return false;
            }
            for (long i = 5; i * i <= num; i += 6) {
                if (num % i == 0 || num % (i + 2) == 0) {
                    return false;
                }
            }
            return true;
        }
    }
```

2)

A. Develop a Java program showcasing method overloading with a base class "Phone" containing the dial() method, and two subclasses "CameraPhone" and "SmartPhone" that inherit from the base class and enhance its features. The program should demonstrate and print the results of these enhancements.

```
class Phone {
    public void dial() {
        System.out.println("Dialing a call...");
    }
}

class CameraPhone extends Phone {
    public void dial(String contact) {
        System.out.println("Dialing " + contact + " and capturing a photo...");
    }
}

class SmartPhone extends Phone {
    public void dial(String contact) {
        System.out.println("Dialing " + contact + " with smart features...");
    }
}

public class PhoneDemo {
    public static void main(String[] args) {
        Phone basicPhone = new Phone();
        CameraPhone cameraPhone = new CameraPhone();
        SmartPhone smartPhone = new SmartPhone();

        System.out.println("Basic Phone:");
        basicPhone.dial();

        System.out.println("\nCamera Phone:");
        cameraPhone.dial("John");

        System.out.println("\nSmart Phone:");
        smartPhone.dial("Alice");
    }
}
```

B. Develop a Java program illustrating constructor overloading for calculating the area of a rectangle and a circle using appropriate constructors.

```
class Shape {
    protected double area;

    public Shape() {
        area = 0;
    }

    public Shape(double area) {
        this.area = area;
    }
}

class Rectangle extends Shape {
    private double length;
    private double width;

    public Rectangle(double length, double width) {
        this.length = length;
        this.width = width;
        area = calculateArea();
    }

    private double calculateArea() {
        return length * width;
    }

    public void display() {
        System.out.println("Rectangle - Length: " + length + ", Width: " + width + ", Area: "
+ area);
    }
}

class Circle extends Shape {
    private double radius;

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

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

    public void display() {
        System.out.println("Circle - Radius: " + radius + ", Area: " + area);
    }
}

public class ShapeDemo {
    public static void main(String[] args) {
        Rectangle rectangle = new Rectangle(5.0, 4.0);
        Circle circle = new Circle(3.0);

        rectangle.display();
        circle.display();
    }
}
```

3)

A. Create a Java program with a vehicle hierarchy, including Vehicle, Car, SportsCar, and Truck classes. Implement methods for starting and stopping in the base class and specialized methods for accelerating, adding turbo boost, and loading cargo in the subclasses, with appropriate method overrides.

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

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

class Car extends Vehicle {
    public void accelerate() {
        System.out.println("Car is accelerating.");
    }

    @Override
    public void start() {
        System.out.println("Car is starting.");
    }

    @Override
    public void stop() {
        System.out.println("Car is stopping.");
    }
}

class SportsCar extends Car {
    public void addTurboBoost() {
        System.out.println("Sports Car added turbo boost.");
    }

    @Override
    public void start() {
        System.out.println("Sports Car is starting.");
    }
    @Override
    public void stop() {
        System.out.println("Sports Car is stopping.");
    }
}

class Truck extends Vehicle {
    public void loadCargo() {
        System.out.println("Truck is loading cargo.");
    }

    @Override
    public void start() {
        System.out.println("Truck is starting.");
    }

    @Override
    public void stop() {
        System.out.println("Truck is stopping.");
    }
}

public class VehicleHierarchyDemo {
    public static void main(String[] args) {
        Vehicle genericVehicle = new Vehicle();
        Car car = new Car();
        SportsCar sportsCar = new SportsCar();
        Truck truck = new Truck();

        genericVehicle.start();
        genericVehicle.stop();

        car.start();
        car.accelerate();
        car.stop();

        sportsCar.start();
        sportsCar.accelerate();
        sportsCar.addTurboBoost();
        sportsCar.stop();

        truck.start();
        truck.loadCargo();
        truck.stop();
    }
}
```

B. Create a Java program that models electronic devices (e.g., smartphones, laptops, and tablets) using a common interface for power management. The program should allow users to interact with the devices and control their power state.

```
interface PowerManageable {
    void powerOn();

    void powerOff();

    boolean isPoweredOn();
}

class ElectronicDevice implements PowerManageable {
    private boolean poweredOn;

    public ElectronicDevice() {
        poweredOn = false;
    }

    @Override
    public void powerOn() {
        poweredOn = true;
        System.out.println("Device powered on.");
    }

    @Override
    public void powerOff() {
        poweredOn = false;
        System.out.println("Device powered off.");
    }

    @Override
    public boolean isPoweredOn() {
        return poweredOn;
    }
}

class Smartphone extends ElectronicDevice {
    private String brand;

    public Smartphone(String brand) {
        this.brand = brand;
    }

    public void makeCall(String contact) {
        if (isPoweredOn()) {
            System.out.println(brand + " smartphone is making a call to " + contact);
        } else {
            System.out.println("Device is powered off. Cannot make a call.");
        }
    }
}

class Laptop extends ElectronicDevice {
    private String brand;

    public Laptop(String brand) {
        this.brand = brand;
    }

    public void openApplication(String appName) {
        if (isPoweredOn()) {
            System.out.println(brand + " laptop is opening the " + appName + "
application.");
        } else {
            System.out.println("Device is powered off. Cannot open applications.");
        }
    }
}

class Tablet extends ElectronicDevice {
    private String brand;

    public Tablet(String brand) {
        this.brand = brand;
    }

    public void browseWeb(String website) {
        if (isPoweredOn()) {
            System.out.println(brand + " tablet is browsing the web on " + website);
        } else {
            System.out.println("Device is powered off. Cannot browse the web.");
        }
    }
}

public class ElectronicDeviceDemo {
    public static void main(String[] args) {
        Smartphone samsungPhone = new Smartphone("Samsung");
        Laptop dellLaptop = new Laptop("Dell");
        Tablet appleTablet = new Tablet("Apple");

        samsungPhone.powerOn();
        samsungPhone.makeCall("John");

        dellLaptop.powerOn();
        dellLaptop.openApplication("Word");

        appleTablet.powerOn();
        appleTablet.browseWeb("example.com");

        samsungPhone.powerOff();
        samsungPhone.makeCall("Alice");

        dellLaptop.powerOff();
        dellLaptop.openApplication("Excel");

        appleTablet.powerOff();
        appleTablet.browseWeb("google.com");
    }
}
```

4)

A. Develop a Java program that emulates a library system. Create two packages, `library` and `patron`. In the `library` package, define a `Book` class with a private title field. In the `patron` package, implement a `Patron` class that can borrow books. Demonstrate the use of packages, access protection, and class imports. Ensure that the book title remains inaccessible from outside the `library` package due to the `private` access modifier. Create a scenario where a patron, Alice, borrows a book from the library.

```
package library;

public class Book {
    private String title;

    public Book(String title) {
        this.title = title;
    }

    // Getter method for title
    public String getTitle() {
        return title;
    }
}

package patron;
import library.Book; // Import the Book class from the library package

public class Patron {
    private String name;

    public Patron(String name) {
        this.name = name;
    }

    public void borrowBook(Book book) {
        System.out.println(name + " borrowed the book titled: " + book.getTitle());
    }
}

import library.Book;
import patron.Patron;
```

```
public class LibraryDemo {
    public static void main(String[] args) {
        // Create a book in the library package
        Book book = new Book("The Great Gatsby");

        // Create a patron in the patron package
        Patron alice = new Patron("Alice");

        // Alice borrows a book from the library
        alice.borrowBook(book);
    }
}
```

B. Develop a Java lab program that handles exceptions for division by zero and invalid input. Use `try-catch` blocks to catch `ArithmeticException` for division by zero and `InputMismatchException` for non-integer input and provide user-friendly error messages.

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

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

        try {
            System.out.print("Enter an integer numerator: ");
            int numerator = scanner.nextInt();

            System.out.print("Enter an integer denominator: ");
            int denominator = scanner.nextInt();

            int result = divide(numerator, denominator);
            System.out.println("Result of division: " + result);
        } catch (ArithmeticException e) {
            System.out.println("Error: Division by zero is not allowed.");
        } catch (InputMismatchException e) {
            System.out.println("Error: Please enter valid integers.");
        } finally {
            scanner.close();
        }
    }
    private static int divide(int numerator, int denominator) {
        if (denominator == 0) {
            throw new ArithmeticException("Division by zero");
        }
        return numerator / denominator;
    }
}
```

5)

A. Write a Java program that implements a multi-thread application that has three threads. First thread generates a random integer for every 1 second; second thread computes the square of the number and prints; third thread will print the value of cube of the number.

```
import java.util.Random;

public class MultiThreadExample {
    public static void main(String[] args) {
        NumberGenerator numberGenerator = new NumberGenerator();
        SquareCalculator squareCalculator = new SquareCalculator(numberGenerator);
        CubePrinter cubePrinter = new CubePrinter(numberGenerator);

        Thread thread1 = new Thread(numberGenerator);
        Thread thread2 = new Thread(squareCalculator);
        Thread thread3 = new Thread(cubePrinter);

        thread1.start();
        thread2.start();
        thread3.start();
    }
}

class NumberGenerator implements Runnable {
    @Override
    public void run() {
        Random random = new Random();
        while (true) {
            int number = random.nextInt(100);
            System.out.println("Generated: " + number);
            try {
                Thread.sleep(1000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}

class SquareCalculator implements Runnable {
    private NumberGenerator numberGenerator;

    public SquareCalculator(NumberGenerator numberGenerator) {
        this.numberGenerator = numberGenerator;
    }

    @Override
    public void run() {
        while (true) {
            int number = numberGenerator.getLastGeneratedNumber();
            int square = number * number;
            System.out.println("Square: " + square);
            try {
                Thread.sleep(1000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}

class CubePrinter implements Runnable {
    private NumberGenerator numberGenerator;

    public CubePrinter(NumberGenerator numberGenerator) {
        this.numberGenerator = numberGenerator;
    }

    @Override
    public void run() {
        while (true) {
            int number = numberGenerator.getLastGeneratedNumber();
            int cube = number * number * number;
            System.out.println("Cube: " + cube);
            try {
                Thread.sleep(1000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}
```

B. Design a Java lab program to demonstrate string handling, including creating strings using constructors and literals, concatenating strings, extracting characters at a specified index, and comparing strings for equality.

```
public class StringHandlingLab {
    public static void main(String[] args) {
        // Creating strings using constructors
        String constructorString1 = new String("Hello, ");
        String constructorString2 = new String("world!");

        // Creating strings using literals
        String literalString1 = "Hello, ";
        String literalString2 = "world!";

        // Concatenating strings
        String result = constructorString1 + constructorString2;

        // Extracting characters at a specified index
        char charAtIndex = result.charAt(7);

        // Comparing strings for equality
        boolean areEqual = constructorString1.equals(literalString1);

        // Displaying the results
        System.out.println("Using constructors:");
        System.out.println(constructorString1);
        System.out.println(constructorString2);

        System.out.println("\nUsing literals:");
        System.out.println(literalString1);
        System.out.println(literalString2);
        System.out.println("\nConcatenated String:");
        System.out.println(result);

        System.out.println("\nCharacter at index 7: " + charAtIndex);

        System.out.println("\nAre   constructorString1   and   literalString1   equal?   "   +
areEqual);
    }
}
```