JIGJIGA UNIVERSITY

SCHOOL OF MECHANICAL & ELECTRICAL ENGINEERING

DEP'T : ELECTRICAL & COMPUTER ENGINEERING

COURSE: OBJECT OREINTED PROGRAMMING
COURSE CODE: ECEG-2133

INDIVIDUAL ASSIGNMENT

STUDENT NAME: ABDIFATAH MAHDI IBRAHIM
ID. NO: R-5901-14

MAY 20241.

1.Calculate the sum of prime
numbers between 1 and 100 in java

Here's a Java program to calculate the sum of prime numbers between 1 and 100:

```
public class SumOfPrimes {
    public static void main(String[] args) {
        int sum = 0;
        for (int i = 2; i <= 100; i++) {
            if (isPrime(i)) {
                sum += i;
            }
        }
        System.out.println("The sum of prime numbers between 1 and 100 is: " + sum);
    }

    public static boolean isPrime(int num) {
        if (num <= 1) {
            return false;
        }
        for (int i = 2; i <= Math.sqrt(num); i++) {
            if (num % i == 0) {
                return false;
            }
        }
        return true;
    }
}
```

When you run this program, it will output the sum of prime numbers between 1 and 100.

2.Calculate the LCM of the five
numbers.

Sure! Here's a Java program to calculate the Least Common Multiple (LCM) of five numbers:

```
public class LCMCalculator {
    public static void main(String[] args) {
        int num1 = 12;
        int num2 = 18;
        int num3 = 24;
        int num4 = 30;
        int num5 = 36;

        int lcm = findLCM(num1, num2, num3, num4, num5);
        System.out.println("The LCM of " + num1 + ", " + num2 + ", " + num3 + ", " + num4 + ", and "
+ num5 + " is: " + lcm);
    }
```

```
    public static int findLCM(int a, int b) {
        return a * b / findGCD(a, b);
    }

    public static int findLCM(int a, int b, int c, int d, int e) {
        return findLCM(findLCM(findLCM(findLCM(a, b), c), d), e);
    }

    public static int findGCD(int a, int b) {
        if (b == 0) {
            return a;
        }
        return findGCD(b, a % b);
    }
}
```

You can modify the values of num1, num2, num3, num4 and num5 to calculate the LCM of any five numbers. When you run this program, it will output the LCM of the provided numbers.

3.Write about Inheritance in Object oriented programming. & Write about Polymorphism in Object oriented programming

1. **Inheritance in Object-Oriented Programming:**
Inheritance is a fundamental concept in object-oriented programming that allows a class to inherit properties and behavior from another class, known as the superclass or parent class. The class that inherits these properties and behavior is called the subclass or child class. Inheritance promotes code reusability and establishes a hierarchical relationship between classes.

When a subclass inherits from a superclass, it automatically gains access to all the public and protected members (fields and methods) of the superclass. The subclass can then extend the functionality of the superclass by adding new fields and methods or by overriding existing methods.

Key points about inheritance:
- The subclass can only inherit from one superclass (single inheritance in languages like Java), but it can implement multiple interfaces.
- Inheritance supports the "is-a" relationship, where a subclass is a specialized version of its superclass.
- Inheritance allows for method overriding, where a subclass can provide its own implementation of a method defined in the superclass.

Example:

```
class Animal {
    void eat() {
        System.out.println("Animal is eating");
    }
}

class Dog extends Animal {
    void bark() {
        System.out.println("Dog is barking");
```

```
    }
}

public class Main {
    public static void main(String[] args) {
        Dog dog = new Dog();
        dog.eat(); // Output: Animal is eating
        dog.bark(); // Output: Dog is barking
    }
}
```

2. **Polymorphism in Object-Oriented Programming:**
Polymorphism is another key concept in object-oriented programming that allows objects of different classes to be treated as objects of a common superclass. There are two types of polymorphism: compile-time polymorphism (method overloading) and runtime polymorphism (method overriding).

- **Compile-time Polymorphism:** Method overloading allows a class to have multiple methods with the same name but different parameters. The compiler determines which method to call based on the number and type of arguments passed to it.

Example:

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

    double add(double a, double b) {
        return a + b;
    }
}
```

- **Runtime Polymorphism:** Method overriding allows a subclass to provide a specific implementation of a method that is already defined in its superclass. The actual method that gets executed is determined at runtime based on the object's type.

Example:

```
class Animal {
    void sound() {
        System.out.println("Animal makes a sound");
    }
}

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

```
public class Main {
    public static void main(String[] args) {
        Animal animal = new Cat();
        animal.sound(); // Output: Cat meows
    }
}
```

Polymorphism simplifies code maintenance and promotes flexibility by allowing objects to be treated uniformly despite their individual differences.