Module 3 - Core Java

11. Factorial Calculator

```
import java.util.Scanner;

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

        if (num < 0) {
            System.out.println("Factorial is not defined for negative numbers.");
        } else {
            long factorial = 1;
            for (int i = 1; i <= num; i++) {
                factorial *= i;
            }
            System.out.println("Factorial of " + num + " is " + factorial);
        }
    }
}
```

12. Method Overloading

```
public class MethodOverloading {
    public static int add(int a, int b) {
        return a + b;
    }

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

    public static int add(int a, int b, int c) {
        return a + b + c;
    }

    public static void main(String[] args) {
        System.out.println("Sum (int, int): " + add(5, 10));
        System.out.println("Sum (double, double): " + add(5.5, 10.5));
        System.out.println("Sum (int, int, int): " + add(1, 2, 3));
    }
}
```

13. Recursive Fibonacci

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

```java
public class RecursiveFibonacci {
    public static int fibonacci(int n) {
        if (n <= 1) return n;
        return fibonacci(n - 1) + fibonacci(n - 2);
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter a positive integer: ");
        int n = sc.nextInt();
        System.out.println("Fibonacci number at position " + n + " is " + fibonacci(n));
    }
}
```

14. Array Sum and Average

```java
import java.util.Scanner;

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

        System.out.println("Enter the elements:");
        for (int i = 0; i < n; i++) {
            arr[i] = sc.nextInt();
            sum += arr[i];
        }

        double average = (double) sum / n;
        System.out.println("Sum: " + sum);
        System.out.println("Average: " + average);
    }
}
```

15. String Reversal

```java
import java.util.Scanner;

public class StringReversal {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter a string: ");
        String input = sc.nextLine();

        StringBuilder reversed = new StringBuilder(input).reverse();
        System.out.println("Reversed string: " + reversed);
    }
}
```

16. Palindrome Checker

```
import java.util.Scanner;

public class PalindromeChecker {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter a string: ");
        String input = sc.nextLine().replaceAll("[^a-zA-Z0-9]", "").toLowerCase();

        String reversed = new StringBuilder(input).reverse().toString();
        if (input.equals(reversed)) {
            System.out.println("The string is a palindrome.");
        } else {
            System.out.println("The string is not a palindrome.");
        }
    }
}
```

17. Class and Object Creation

```
class Car {
    String make, model;
    int year;

    void displayDetails() {
        System.out.println("Make: " + make + ", Model: " + model + ", Year: " + year);
    }
}

public class CarDemo {
    public static void main(String[] args) {
        Car car1 = new Car();
        car1.make = "Toyota";
        car1.model = "Corolla";
        car1.year = 2020;
        car1.displayDetails();
    }
}
```

18. Inheritance Example

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

class Dog extends Animal {
    @Override
    void makeSound() {
        System.out.println("Bark");
    }
}

public class InheritanceDemo {
    public static void main(String[] args) {
        Animal a = new Animal();
        Dog d = new Dog();

        a.makeSound();
        d.makeSound();
    }
}
```

19. Interface Implementation

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

class Game implements Playable {
    public void play() {
        System.out.println("Playing the game...");
    }
}

public class InterfaceDemo {
    public static void main(String[] args) {
        Playable game = new Game();
        game.play();
    }
}
```