Question. Design a class to overload a function sumSeries() as follows:

(i) void sumSeries(int n, double x): with one integer argument and one double argument to find and display the sum of the series given below:

Sum = x / 1 – x / 2 + x / 3 – x / 4 + x / 5 ... N terms.

(ii) void sumSeries(): to find and display the sum of the following series:

Sum = 1 + (1 × 2) + (1 × 2 × 3) + ... + (1 × 2 × 3 × 4 ... × 20)

Source Code

```
class Overload {
    // Method to calculate the sum of the series (alternating signs)
    void sumSeries(int n, double x) {
        double sum = 0.0;
        int sign = 1;
        for(int i = 1; i <= n; i++) {
            double term = sign * (x / i);
            sum += term;
            sign *= -1;
        }
        System.out.println("Sum of Series 1 = " + sum);
    }

    // Method to calculate the sum of factorials of the first 20 integers
    void sumSeries() {
        long sum = 0;
        for(int i = 1; i <= 20; i++) {
            long f = 1;
            for(int j = 1; j <= i; j++) {
                f *= j;
            }
            sum += f;
        }
        System.out.println("Sum of Series 2 = " + sum);
    }
}

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

```
        // Creating an object of the Overload class
        Overload obj = new Overload();

        // Executing the first sumSeries method with parameters
        obj.sumSeries(5, 2.0); // Example call with n=5 and x=2.0

        // Executing the second sumSeries method without parameters
        obj.sumSeries();
    }
}
```

Explanation of the Source Code

1. **Class Definition**:
    - `Overload` class contains two overloaded methods named `sumSeries`.
2. **Method** `sumSeries(int n, double x)`:
    - Calculates and prints the sum of the series x1−x2+x3−…\frac{x}{1} - \frac{x}{2} + \frac{x}{3} - \ldots for `n` terms.
3. **Method** `sumSeries()`:
    - Calculates and prints the sum of the factorials of the first 20 integers.
4. **Main Class**:
    - `Main` class contains the `main` method, which is the entry point of the program.
    - Creates an object `obj` of the `Overload` class.
    - Calls `sumSeries(int n, double x)` method with example parameters `5` and `2.0`.
    - Calls `sumSeries()` method without any parameters.

2.Overloaded Method for Calculating Area

Source Code

```
class AreaCalculator {
    // Method to calculate the area of a rectangle
    double calculateArea(double length, double breadth) {
        return length * breadth;
```

```
    }

    // Overloaded method to calculate the area of a circle
    double calculateArea(double radius) {
        return Math.PI * radius * radius;
    }

    // Overloaded method to calculate the area of a square
    double calculateArea(int side) {
        return side * side;
    }
}

public class Main {
    public static void main(String[] args) {
        AreaCalculator calculator = new AreaCalculator();

        // Calculate the area of a rectangle
        double rectangleArea = calculator.calculateArea(5.0, 3.0);
        System.out.println("Area of Rectangle = " + rectangleArea);

        // Calculate the area of a circle
        double circleArea = calculator.calculateArea(4.0);
        System.out.println("Area of Circle = " + circleArea);

        // Calculate the area of a square
        double squareArea = calculator.calculateArea(4);
        System.out.println("Area of Square = " + squareArea);
    }
}
```

Explanation

This program demonstrates method overloading by defining three methods with the same name, `calculateArea`, but with different parameter lists:

- One method to calculate the area of a rectangle.
- One method to calculate the area of a circle.
- One method to calculate the area of a square.

Each method computes the area based on the shape's formula and returns the result.

3. Overloaded Method for Displaying Information

Code

```
class DisplayInfo {
    // Method to display an integer
    void display(int a) {
        System.out.println("Displaying integer: " + a);
    }

    // Overloaded method to display a double
    void display(double a) {
        System.out.println("Displaying double: " + a);
    }

    // Overloaded method to display a string
    void display(String a) {
        System.out.println("Displaying string: " + a);
    }
}

public class Main {
    public static void main(String[] args) {
        DisplayInfo info = new DisplayInfo();

        // Display an integer
        info.display(5);

        // Display a double
        info.display(5.5);

        // Display a string
        info.display("Hello, World!");
    }
}
```

Explanation

This program demonstrates method overloading by defining three methods with the same name, `display`, but with different parameter lists:

- One method to display an integer.
- One method to display a double.
- One method to display a string.

Each method prints the provided argument based on its type.

4. Overloaded Methods for Greeting

Code

```
class Greeter {
    // Method to greet with a default message
    void greet() {
        System.out.println("Hello, World!");
    }

    // Overloaded method to greet a specific person
    void greet(String name) {
        System.out.println("Hello, " + name + "!");
    }

    // Overloaded method to greet a person with a personalized message
    void greet(String name, String message) {
        System.out.println(message + ", " + name + "!");
    }
}

public class Main {
    public static void main(String[] args) {
        Greeter greeter = new Greeter();

        // Greet with a default message
        greeter.greet();

        // Greet a specific person
        greeter.greet("Alice");

        // Greet a person with a personalized message
        greeter.greet("Bob", "Good morning");
    }
}
```

Explanation

This program demonstrates method overloading by defining three methods named `greet`, each with different parameter lists:

- One method with no parameters to print a default greeting.
- One method with a `String` parameter to greet a specific person by name.
- One method with two `String` parameters to greet a person with a personalized message.

5. Overloaded Methods for Printing Arrays

Code

```
class ArrayPrinter {
    // Method to print an array of integers
    void printArray(int[] array) {
        System.out.print("Integer array: ");
        for (int i : array) {
            System.out.print(i + " ");
        }
        System.out.println();
    }

    // Overloaded method to print an array of strings
    void printArray(String[] array) {
        System.out.print("String array: ");
        for (String s : array) {
            System.out.print(s + " ");
        }
        System.out.println();
    }

    // Overloaded method to print a subarray
    void printArray(int[] array, int start, int end) {
        System.out.print("Subarray: ");
        for (int i = start; i <= end && i < array.length; i++) {
            System.out.print(array[i] + " ");
        }
        System.out.println();
    }
}
```

```
public class Main {
    public static void main(String[] args) {
        ArrayPrinter printer = new ArrayPrinter();

        // Print an array of integers
        int[] intArray = {1, 2, 3, 4, 5};
        printer.printArray(intArray);

        // Print an array of strings
        String[] strArray = {"Hello", "world", "!"};
        printer.printArray(strArray);

        // Print a subarray of integers
        printer.printArray(intArray, 1, 3);
    }
}
```

Explanation

This program demonstrates method overloading by defining three methods named `printArray`, each with different parameter lists and logic:

- One method to print an array of integers.
- One method to print an array of strings.
- One method to print a subarray of integers given start and end indices.