```java
import java.math.BigDecimal;
import java.math.BigInteger;
import java.math.RoundingMode;

public class Assign1 {

    /**
     * Computes the factorial of a given non-negative integer.
     * @author Gabriel Nielsen
     * @param value the non-negative integer to compute the factorial for
     * @return the factorial of the given integer as a BigInteger
     * @throws IllegalArgumentException if the input value is negative
     */
    private static BigInteger factorial(int value) {
        if (value < 0) {
            throw new IllegalArgumentException("factorial valid range: [0, " +
Integer.MAX_VALUE + "]");
        }
        BigInteger total = BigInteger.valueOf(1);

        for (int i = 1; i <= value; i++) {
            total = total.multiply(BigInteger.valueOf(i));
        }
        return total;
    }

    /**
     * Computes the nth Fibonacci number using recursion.
     * @author Gabriel Nielsen
     * @param value the index of the Fibonacci number to compute (0 <= value <= 40)
     * @return the nth Fibonacci number
     * @throws IllegalArgumentException if the input value is negative or greater
than 40
     */
    private static int fibonacci(int value) {
        if (value < 0 || value > 40) {
            throw new IllegalArgumentException("fibonacci valid range: [0, 40]");
        }
        if (value == 0 || value == 1) {
            return 1;
        }  else {
            return fibonacci(value - 1) + fibonacci(value - 2);
        }
    }

    /**
     * Computes the mathematical constant e using a series expansion.
     * @author Gabriel Nielsen
     * @param value the number of terms to include in the series (value > 0)
     * @return the computed value of e as a BigDecimal
     * @throws IllegalArgumentException if the input value is less than 1
     */
    private static BigDecimal eSeries(int value) {
        if (value < 1) {
            throw new IllegalArgumentException("Valid e iteration range: [1, " +
Integer.MAX_VALUE + "]");
        }
        BigDecimal sum = BigDecimal.ONE;
        BigDecimal term = BigDecimal.ONE;
        int SCALE = 25;

        for (int i = 1; i < value; i++) {
            term = term.divide(BigDecimal.valueOf(i), SCALE, RoundingMode.HALF_UP);
            sum = sum.add(term);
        }
        return sum;
    }

    public static void main(String[] args) {
        if (args.length == 0) {
            System.out.println("-- Assign1 Info --");
            System.out.println(" -fib [n] : Compute the nth Fibonacci number (0 <=
n <= 40)");
            System.out.println(" -fac [n] : Compute the factorial of n (n >= 0)");
            System.out.println(" -e [n] : Compute the value of e using n iterations
(n > 0)");
            return;
        }

        for (int i = 0; i < args.length; i++) {
            if (i % 2 == 0) {
                switch (args[i]) {
                    case "-fib" -> {
                        try {
                            int n = Integer.parseInt(args[i + 1]);
                            int result = fibonacci(n);
                            System.out.println("Fibonacci of " + n + " is " +
result);
                        } catch (NumberFormatException e) {
                            System.out.println("Fibonacci value must be an
integer.");
                        } catch (ArrayIndexOutOfBoundsException e) {
                            System.out.println("missing value for -fib");
                        } catch (IllegalArgumentException e) {
                            System.out.println(e.getMessage());
                        }
                    }
                    case "-fac" -> {
                        try {
                            int n = Integer.parseInt(args[i + 1]);
                            BigInteger result = factorial(n);
                            System.out.println("Factorial of " + n + " is " +
result);
                        } catch (NumberFormatException e) {
                            System.out.println("Factorial value must be an
integer.");
                        } catch (ArrayIndexOutOfBoundsException e) {
                            System.out.println("missing value for -fac");
                        } catch (IllegalArgumentException e) {
                            System.out.println(e.getMessage());
                        }
                    }
                    case "-e" -> {
                        try {
                            int n = Integer.parseInt(args[i + 1]);
                            BigDecimal result = eSeries(n);
                            System.out.println("Value of e using " + n + "
iterations is " + result);
                        } catch (NumberFormatException e) {
                            System.out.println("e value must be an integer.");
                        } catch (ArrayIndexOutOfBoundsException e) {
                            System.out.println("missing value for -e");
                        } catch (IllegalArgumentException e) {
                            System.out.println(e.getMessage());
                        }
                    }
                    default -> System.out.println("Unknown argument: " + args[i]);
                }
            }
        }
    }
}
```