Annaly Rocha
09/05/2023
COP4331

1)

```
public class Fib {
    private int f0;
    private int f1;

    public Fib(int f0, int f1) {
        this.f0 = f0;
        this.f1 = f1;
    }
    public int calculateFibonacci(int n, boolean iterative) {
        if (n < 0) {
            throw new IllegalArgumentException("n must be non-negative");
        }
        if (n == 0) return f0;
        if (n == 1) return f1;

        int prev1 = f0;
        int prev2 = f1;
        int current = 0;

        if (iterative) {
            for (int i = 2; i <= n; i++) {
                current = prev1 + prev2;
                prev1 = prev2;
                prev2 = current;
            }
        } else {
            current = calculateFibonacci(n - 1, false) + calculateFibonacci(n - 2, false);
        }

        return current;
    }

    public static void main(String[] args) {
        if (args.length != 3) {
            System.out.println("Usage: java Fib <F(0)> <F(1)> <n>");
            return;
        }

        int f0 = Integer.parseInt(args[0]);
        int f1 = Integer.parseInt(args[1]);
        int n = Integer.parseInt(args[2]);

        Fib fib = new Fib(f0, f1);
        System.out.println("Fibonacci Series (Iterative):");
        for (int i = 0; i <= n; i++) {
            System.out.print(fib.calculateFibonacci(i, true) + " ");
        }
        System.out.println();

        System.out.println("Fibonacci Series (Recursive):");
        for (int i = 0; i <= n; i++) {
            System.out.print(fib.calculateFibonacci(i, false) + " ");
        }
        System.out.println();
    }
}
```

2)

```
  public class Greeter {
    private String name;

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

    public String getName() {
        return name;
    }

    public void swapNames(Greeter other) {
        String temp = this.name;
        this.name = other.name;
        other.name = temp;
    }

    public Greeter createQualifiedGreeter(String qualifier) {
        return new Greeter(qualifier + " " + this.name);
    }
```

3)

```
import java.io.*;
import java.util.LinkedList;


public class DataAnalyzer {
    private final LinkedList<Integer> numList;
 public DataAnalyzer(LinkedList<Integer> numList) {
        this.numList = numList;
    }

    /**
    public int min() {
        if (numList.isEmpty()) {
            throw new IllegalStateException("The list is empty.");
        }
        int min = numList.getFirst();
        for (int num : numList) {
            if (num < min) {
                min = num;
            }
        }
        return min;
    }
    public int max() {
        if (numList.isEmpty()) {
            throw new IllegalStateException("The list is empty.");
        }
        int max = numList.getFirst();
        for (int num : numList) {
            if (num > max) {
                max = num;
            }
        }
        return max;
    }

     public double average() {
        if (numList.isEmpty()) {
            throw new IllegalStateException("The list is empty.");
        }
        int sum = 0;
        for (int num : numList) {
            sum += num;
        }
        return (double) sum / numList.size();
    }
}
import java.io.*;
import java.util.LinkedList;
import java.util.Scanner;

public class DataAnalyzerTester {
    public static void main(String[] args) {
        if (args.length != 1) {
            System.out.println("Usage: java DataAnalyzerTester <output_filename>");
            System.exit(1);
        }

        String outputFileName = args[0];
        LinkedList<Integer> numList = new LinkedList<>();
        Scanner scanner = new Scanner(System.in);

        System.out.println("Enter a sequence of integers (one per line, end with a non-integer input):");
        while (scanner.hasNextInt()) {
            numList.add(scanner.nextInt());
        }

        try (PrintWriter writer = new PrintWriter(new FileWriter(outputFileName))) {
            for (int num : numList) {
                writer.println(num);
            }

            DataAnalyzer analyzer = new DataAnalyzer(numList);
            int min = analyzer.min();
            int max = analyzer.max();
            double average = analyzer.average();

            System.out.println("Minimum: " + min);
            System.out.println("Maximum: " + max);
            System.out.println("Average: " + average);

            writer.println("Minimum: " + min);
            writer.println("Maximum: " + max);
            writer.println("Average: " + average);
        } catch (IOException e) {
            System.err.println("Error writing to the file: " + e.getMessage());
        }
    }
}
```

4)

```
class Greeter {
    private String name;

    public Greeter(String name) {
        this.name = name;
    }
    public String sayHello() {
        return "Hello, " + name + "!";
    }
}

public class Main {
    public static void main(String[] args) {
        int x = 0;

        try {
            Greeter g1 = new Greeter("Alice");
            Greeter g2 = new Greeter("Alice");

            if (!g1.sayHello().equals(g2.sayHello())) {
                g2 = null;
            }

            x = 1;
            System.out.println(g2.sayHello());
            x = 2;

        } catch (NullPointerException ex) {
            x = 10;
        } catch (RuntimeException ex) {
            x = 4;
        } finally {
            x++;
        }

        System.out.println("Final value of x: " + x);
    }
}
```

5)

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

public class PrimeFactorizer {
    private int n;
    private ArrayList<Integer> primes;
    private ArrayList<Integer> exponents;
    private boolean computed;

    public PrimeFactorizer(int n) {
        this.n = n;
        this.primes = new ArrayList<>();
        this.exponents = new ArrayList<>();
        this.computed = false;
    }

    public int getN() {
        return n;
    }

    public void compute() {
        if (!computed) {
            int num = n;
            for (int i = 2; i * i <= num; i++) {
                if (num % i == 0) {
                    int count = 0;
                    while (num % i == 0) {
                        num /= i;
                        count++;
                    }
                    primes.add(i);
                    exponents.add(count);
                }
            }
            if (num > 1) {
                primes.add(num);
                exponents.add(1);
            }
            computed = true;
        }
    }

    public void getFactorsAndExponents(int n, ArrayList<Integer> primes, ArrayList<Integer>
exponents) {
        compute();
        primes.addAll(this.primes);
        exponents.addAll(this.exponents);
    }

    @Override
    public String toString() {
        compute();
        StringBuilder result = new StringBuilder(n + " = ");
        for (int i = 0; i < primes.size(); i++) {
            result.append(primes.get(i));
            if (exponents.get(i) > 1) {
                result.append("^").append(exponents.get(i));
            }
            if (i < primes.size() - 1) {
                result.append(" * ");
            }
        }
        return result.toString();
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a positive integer: ");
        int n = scanner.nextInt();

        PrimeFactorizer factorizer = new PrimeFactorizer(n);
        String factorization = factorizer.toString();

        System.out.println(factorization);
    }
}
```