**Tank Circuit Program**

Name: Mushayev Masrur ID: 56856

**1.1) Design:**

The application is divided into 4 classes. Main class takes user input and iteratively calculates the resonant frequency, total costs, total capacitance and total inductance. The Component class was being inherited by the Capacitor and Inductor.

When the user gives necessary parameters for a component, a new object of this component is being created and inputs are stored in the object. The object is stored in an array called tankCircuit. By iterating over the tankCircuit list, we calculate the necessary information.

**1.2) Control:**

Program considers all the capacitors in parallel and all inductors in series (as specified in the question).

To match the required printing pattern of the question, 2 iterations of tankCircuit were required. Certain features that are bound to a specific component were calculated in their respective class.

**1.3) Flexibility:**

The program offers a larger range of values to be used as a component’s property. An unlimited number of components can be used. But further improvement of flexibility may focus on considering component’s can be of both series and parallel instead of predefining it.

**1.4) Error Control:**

Errors and Exceptions we handled when necessary. For example, the area of any component cannot be negative. If a user tries to enter a negative area value, a runtime exception will be thrown. If a user tries to enter something invalid, an error will be thrown including termination of the process.

**2) Copy of the Code:**

**# Main.java**

**import java.util.Scanner;**

**import java.util.ArrayList;**

**public class Main {**

**public static void main(String[] args) {**

**try{**

**double PI = Math.*PI*;**

**Scanner scanner = new Scanner(System.*in*);**

**double total\_cost = 0;**

**double total\_capacitance = 0;**

**double total\_inductance = 0;**

**double resonate\_frequency;**

**ArrayList<Component> tankCircuit = new ArrayList<>();**

**// Student name and ID**

**System.*out*.println("Student Name: Mushayev Masrur");**

**System.*out*.println("Student ID: 56856");**

**// control while loop**

**boolean toAddNewC = true;**

**boolean toAddNewI = true;**

**while (toAddNewC) { //loop for capacitor's info as user input**

**System.*out*.print("Are you adding capacitors? (Y/N): ");**

**String loop\_control = scanner.next();**

**if (loop\_control.charAt(0) == 'N') {**

**toAddNewC = false; break;**

**}**

**// ask user for area and distance**

**System.*out*.print("Enter Capacitor Area (mm^2): ");**

**double \_a = scanner.nextDouble();**

**System.*out*.print("Enter Capacitor separated distance (mm): ");**

**double \_d = scanner.nextDouble();**

**Capacitor newCapacitor = new Capacitor(); // create a new capacitor.**

**newCapacitor.setArea(\_a);**

**newCapacitor.setDistance(\_d);**

**tankCircuit.add(newCapacitor); // adding created capacitor to list.**

**}**

**//loop for inductor's info as input**

**while (toAddNewI) {**

**System.*out*.print("Are you adding inductors? (Y/N): ");**

**String loop\_control = scanner.next();**

**if (loop\_control.charAt(0) == 'N') {**

**toAddNewI = false;**

**break;**

**}**

**// asking user for inductance**

**System.*out*.print("Enter Inductance of the inductor (uH): ");**

**double \_l = scanner.nextDouble();**

**Inductor newIndc = new Inductor(); // creating a new inductor object.**

**newIndc.setL(\_l);**

**tankCircuit.add(newIndc); // adding newly created inductor to list.**

**}**

**// calculate total capacitance, inductance, and resonant frequency**

**for (int idx = 0; idx < tankCircuit.size(); idx++) {**

**total\_cost += tankCircuit.get(idx).getCost(); // total cost**

**if (tankCircuit.get(idx) instanceof Capacitor) { //if Capacitor**

**total\_capacitance += ((Capacitor) tankCircuit.get(idx)).getCapacitance();**

**}**

**else if (tankCircuit.get(idx) instanceof Inductor) { //if Inductor**

**total\_inductance += ((Inductor) tankCircuit.get(idx)).getL();**

**}**

**else{**

**System.*out*.println("Unknown circuit element");**

**}**

**}**

**//calculate resonate frequency**

**resonate\_frequency = (1 / (2 \* PI \* Math.*sqrt*((total\_inductance) \* total\_capacitance))) / 1000;**

**System.*out*.println("Mushayev Masrur's LC Tank Circuit");**

**System.*out*.printf("Resonate Frequency: %.2f MHz\n", resonate\_frequency);**

**// display information about each capacitor and inductor one by one.**

**for (int idx = 0; idx < tankCircuit.size(); idx++) {**

**if (tankCircuit.get(idx) instanceof Capacitor) { //if Capacitor**

**((Capacitor) tankCircuit.get(idx)).printInfo();**

**}**

**else if (tankCircuit.get(idx) instanceof Inductor) { //if Inductor**

**((Inductor) tankCircuit.get(idx)).printInfo();**

**}**

**}**

**System.*out*.println("Total cost: $" + total\_cost);**

**scanner.close(); // closing the scanner to prevent resource leaks.**

**}**

**catch(Exception e){**

**System.*out*.println("Exception thrown: " + e.getClass() + " : " + e.getMessage());**

**}**

**}**

**}**

**# Component.java**

**public class Component {**

**protected String Name;**

**protected String Description;**

**protected String Manufacturer;**

**protected double Cost;**

**protected boolean Obsolete;**

**public String getName() {**

**return Name;**

**}**

**public String getDescription() {**

**return Description;**

**}**

**public String getManufacturer() {**

**return Manufacturer;**

**}**

**public double getCost() {**

**return Cost;**

**}**

**public boolean isObsolete() {**

**return Obsolete;**

**}**

**}**

**# Capacitor.java**

**import java.lang.Math;**

**public class Capacitor extends Component {**

**public String Warning;**

**double area;**

**double distance;**

**public Capacitor() {**

**Name = "Capacitors";**

**Description = "Store electrical charges.";**

**Manufacturer = "Siemens";**

**Cost = 0.50;**

**Obsolete = false;**

**this.Warning = "Do not use electrolytic capacitors in tank circuits.";**

**}**

**public void setArea(double a) {**

**if(a < 0){**

**throw new RuntimeException("Area cannot be negative");**

**}**

**this.area = a;**

**}**

**public double getArea() {**

**return this.area;**

**}**

**public void setDistance(double d) {**

**this.distance = d;**

**}**

**public double getDistance() {**

**return this.distance;**

**}**

**public String getWarning() {**

**return this.Warning;**

**}**

**// calculate and return capacitance.**

**public double getCapacitance() {**

**double E = 8.85 \* Math.*pow*(10, -12);**

**double C = (E \* this.getArea() / this.getDistance());**

**return C;**

**}**

**public void printInfo() {**

**//the capacitance was multiplied by 1e11 to match pico unit. [As we have used pico Farade to express the capacitance]**

**System.*out*.print(this.getName() + ", ") ;**

**System.*out*.printf("%.2f pF, ", this.getCapacitance() \* 1e11);**

**System.*out*.print(this.getDescription() + ". Warning: " + this.getWarning() + "\n");**

**}**

**}**

**# Inductor.java**

**public class Inductor extends Component {**

**double L;**

**public Inductor() {**

**this.Name = "Inductors";**

**this.Description = "Store electrical charges.";**

**this.Manufacturer = "Motorola";**

**this.Cost = 0.25;**

**this.Obsolete = false;**

**}**

**public void setL(double l) {**

**this.L = l;**

**}**

**public double getL() {**

**return this.L;**

**}**

**public void printInfo() {**

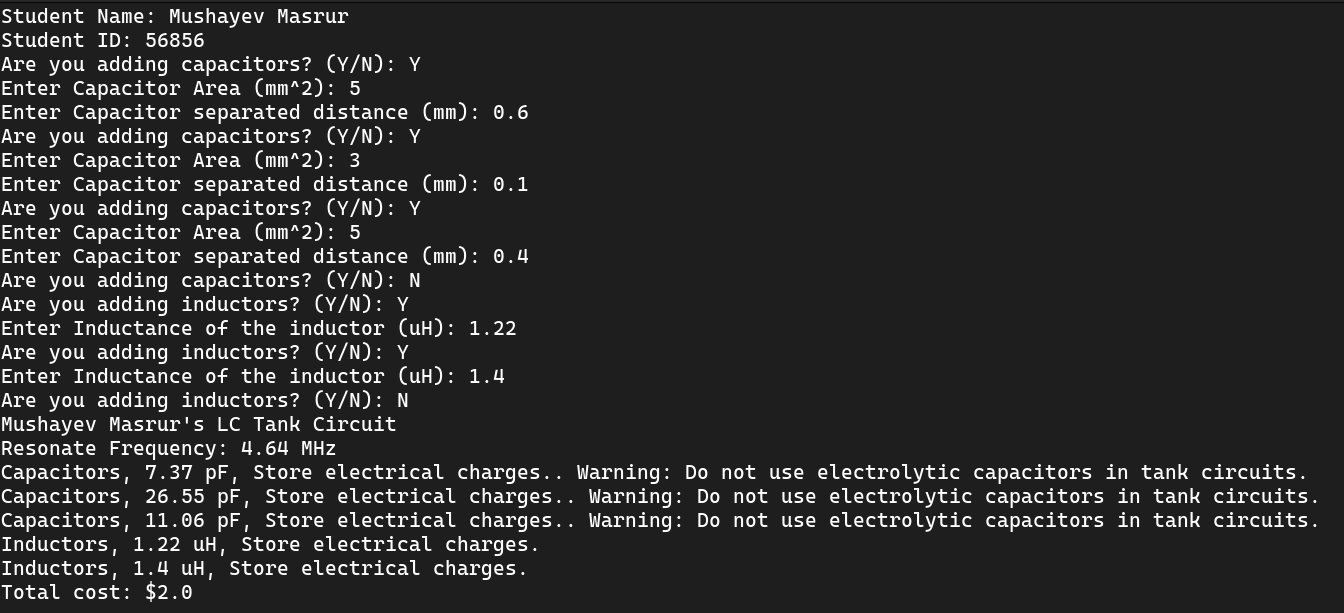
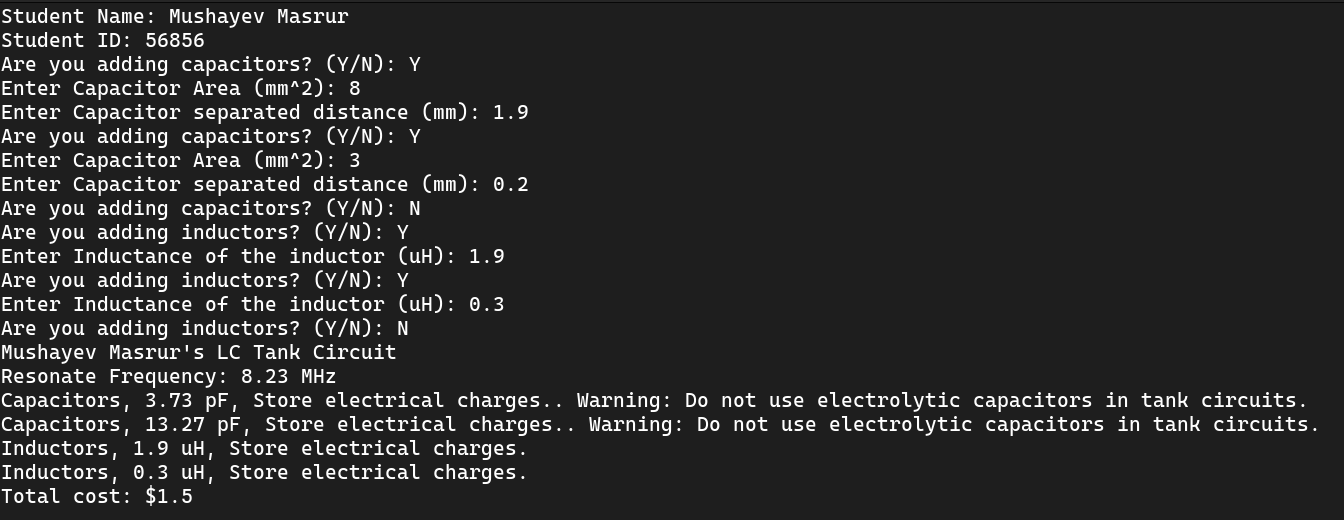
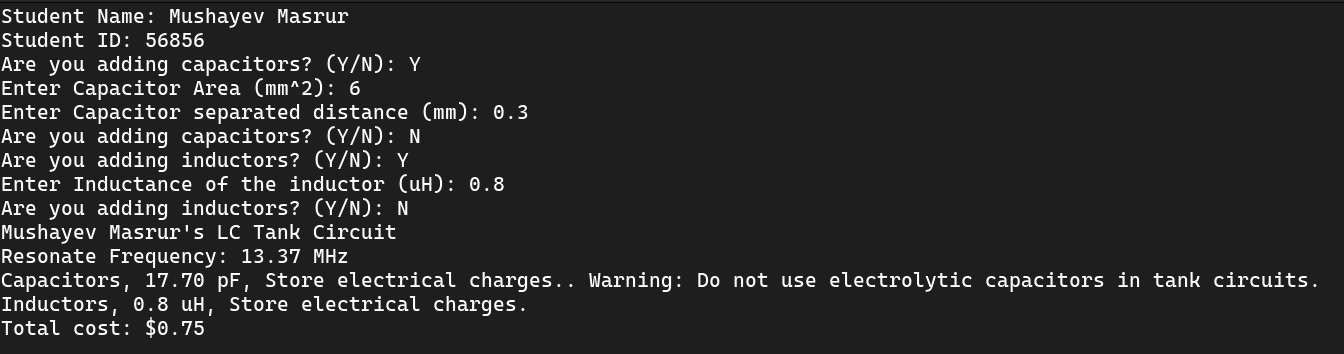
**System.out.println(this.getName() + ", " + this.getL() + " uH, " + this.getDescription());**

**}**

**}**

**3) Screenshots**

**With expected Values :**



**With non-expected Values :**

