1.Schritt

```
//Wiederhole bis jede Berechnung innerhalb von epsilon liegt
    //Durchlaufen der Matrix in x-Richtung
        //Durchlaufen der Matrix in y-Richtung
            //Berechnen des Wertes
            //Prüfen ob der Wert innerhalb von espilon liegt
            //Speichern des neuen Werts in der Matrix
    //Starten der nächsten Iteration
//Wenn ein Wert die Grenzen von epsilon überschritten hat wird wiederholt
//Wenn alle Werte innerhalb der Grenzen liegen wird die Berechnung abgeschlos
```

2.Schritt

```
public void solve() {
    //Wiederhole bis jede Berechnung innerhalb von epsilon liegt
        //Durchlaufen der Matrix in x-Richtung
        for (int x = 0; x < matrix.getSize(); x++) {
            //Durchlaufen der Matrix in y-Richtung
            for (int y = 0; y < matrix.getSize(); y++) {
                //Berechnen des Wertes
                //Prüfen ob der Wert innerhalb von espilon liegt
                //Speichern des neuen Werts in der Matrix
            }
        }
        //Starten der nächsten Iteration
    //Wenn ein Wert die Grenzen von epsilon überschritten hat wird wiederholt
    //Wenn alle Werte innerhalb der Grenzen liegen wird die Berechnung abgesc
}
```

3.Schritt

```
public void solve() {
    //Wiederhole bis jede Berechnung innerhalb von epsilon liegt
        //Durchlaufen der Matrix in x-Richtung
        for (int x = 0; x < matrix.getSize(); x++) {
            //Durchlaufen der Matrix in y-Richtung
            for (int y = 0; y < matrix.getSize(); y++) {
                //Berechnen des Wertes
                double result = matrix.getValue(x - 1, y) +
                        matrix.getValue(x, y - 1) +
                        matrix.getValue(x + 1, y) +
                        matrix.getValue(x, y + 1);
                result = result * 0.25;
                //Prüfen ob der Wert innerhalb von espilon liegt
                //Speichern des neuen Werts in der Matrix
            }
        }
        //Starten der nächsten Iteration
    //Wenn ein Wert die Grenzen von epsilon überschritten hat wird wiederholt
```

```
    //Wenn alle Werte innerhalb der Grenzen liegen wird die Berechnung abgesc
}
```

4.Schritt

```
public void solve() {
    //Wiederhole bis jede Berechnung innerhalb von epsilon liegt
        //Durchlaufen der Matrix in x-Richtung
        for (int x = 0; x < matrix.getSize(); x++) {
            //Durchlaufen der Matrix in y-Richtung
            for (int y = 0; y < matrix.getSize(); y++) {
                //Berechnen des Wertes
                double result = matrix.getMatrixPointValue(x - 1, y) +
                        matrix.getMatrixPointValue(x, y - 1) +
                        matrix.getMatrixPointValue(x + 1, y) +
                        matrix.getMatrixPointValue(x, y + 1);
                result = result * 0.25;
                //Prüfen ob der Wert innerhalb von espilon liegt
                //Speichern des neuen Werts in der Matrix
            }
        }
        //Starten der nächsten Iteration
    //Wenn ein Wert die Grenzen von epsilon überschritten hat wird wiederholt
    //Wenn alle Werte innerhalb der Grenzen liegen wird die Berechnung abgesc
}

//Beachten der Grenzen von x und y und rückgabe
private double getMatrixPointValue(int x, int y) {
    if (x < 0) {
        return links;
    } else if (x > matrix.getSize() - 1) {
        return rechts;
    } else if (y < 0) {
        return oben;
    } else if (y > matrix.getSize() - 1) {
        return unten;
    } else {
        return matrix.getValue(x, y);
    }
}
```

5.Schritt

```
public void solve() {
    //Wiederhole bis jede Berechnung innerhalb von epsilon liegt
    do {
        //Durchlaufen der Matrix in x-Richtung
        for (int x = 0; x < matrix.getSize(); x++) {
            //Durchlaufen der Matrix in y-Richtung
            for (int y = 0; y < matrix.getSize(); y++) {
                //Berechnen des Wertes
                double result = matrix.getMatrixPointValue(x - 1, y) +
                        matrix.getMatrixPointValue(x, y - 1) +
```

```
                        matrix.getMatrixPointValue(x + 1, y) +
                        matrix.getMatrixPointValue(x, y + 1);
                result = result * 0.25;
                //Prüfen ob der Wert innerhalb von espilon liegt
                //Speichern des neuen Werts in der Matrix
            }
        }
        //Starten der nächsten Iteration
        matrix.nextIteration();
    //Wenn ein Wert die Grenzen von epsilon überschritten hat wird wiederholt
    } while (!abbruch);
    //Wenn alle Werte innerhalb der Grenzen liegen wird die Berechnung abgesc
}

//Beachten der Grenzen von x und y und rückgabe
private double getMatrixPointValue(int x, int y) {
    if (x < 0) {
        return links;
    } else if (x > matrix.getSize() - 1) {
        return rechts;
    } else if (y < 0) {
        return oben;
    } else if (y > matrix.getSize() - 1) {
        return unten;
    } else {
        return matrix.getValue(x, y);
    }
}
```

6.Schritt

```
public void solve() {
    //Wiederhole bis jede Berechnung innerhalb von epsilon liegt
    do {
        abbruch = true;
        //Durchlaufen der Matrix in x-Richtung
        for (int x = 0; x < matrix.getSize(); x++) {
            //Durchlaufen der Matrix in y-Richtung
            for (int y = 0; y < matrix.getSize(); y++) {
                //Berechnen des Wertes
                double result = matrix.getMatrixPointValue(x - 1, y) +
                        matrix.getMatrixPointValue(x, y - 1) +
                        matrix.getMatrixPointValue(x + 1, y) +
                        matrix.getMatrixPointValue(x, y + 1);
                result = result * 0.25;
                //Prüfen ob der Wert innerhalb von espilon liegt
                double change = result - matrix.getValue(x, y);
                if (change > epsilon) {
                    abbruch = false;
                }
                //Speichern des neuen Werts in der Matrix
                matrix.setValue(result, x, y);
            }
        }
```

```
            //Starten der nächsten Iteration
            matrix.nextIteration();
        //Wenn ein Wert die Grenzen von epsilon überschritten hat wird wiederholt
        } while (!abbruch);
        //Wenn alle Werte innerhalb der Grenzen liegen wird die Berechnung abgesc
    }

    //Beachten der Grenzen von x und y und rückgabe
    private double getMatrixPointValue(int x, int y) {
        if (x < 0) {
            return links;
        } else if (x > matrix.getSize() - 1) {
            return rechts;
        } else if (y < 0) {
            return oben;
        } else if (y > matrix.getSize() - 1) {
            return unten;
        } else {
            return matrix.getValue(x, y);
        }
    }
```

7.Schritt

```
    public void solve() {
        //Wiederhole bis jede Berechnung innerhalb von epsilon liegt
        do {
            abbruch = true;
            //Durchlaufen der Matrix in x-Richtung
            for (int x = 0; x < matrix.getSize(); x++) {
                //Durchlaufen der Matrix in y-Richtung
                for (int y = 0; y < matrix.getSize(); y++) {
                    //Berechnen des Wertes
                    double result = solve(x, y);
                    //Prüfen ob der Wert innerhalb von espilon liegt
                    double change = result - matrix.getValue(x, y);
                    if (change > epsilon) {
                        abbruch = false;
                    }
                    //Speichern des neuen Werts in der Matrix
                    matrix.setValue(result, x, y);
                }
            }
            //Starten der nächsten Iteration
            matrix.nextIteration();
        //Wenn ein Wert die Grenzen von epsilon überschritten hat wird wiederholt
        } while (!abbruch);
        //Wenn alle Werte innerhalb der Grenzen liegen wird die Berechnung abgesc
    }

    //Refactoring
    private double solve(int x, int y) {
        double result = getMatrixPointValue(x - 1, y) +
                getMatrixPointValue(x, y - 1) +
```

```
                getMatrixPointValue(x + 1, y) +
                getMatrixPointValue(x, y + 1);
        result = result * 0.25;

        return result;
    }

    //Beachten der Grenzen von x und y und rückgabe
    private double getMatrixPointValue(int x, int y) {
        if (x < 0) {
            return links;
        } else if (x > matrix.getSize() - 1) {
            return rechts;
        } else if (y < 0) {
            return oben;
        } else if (y > matrix.getSize() - 1) {
            return unten;
        } else {
            return matrix.getValue(x, y);
        }
    }
```

8.Schritt

```
    //Added Threading for enabling automatic UI-Updates
    public class SequentialSolver extends Solver {

        public SequentialSolver(int oben, int unten, int links, int rechts, doubl
            super(oben, unten, links, rechts, epsilon, matrix, toBeRefreshed);
        }

        /**
         * Zu implementieren. Loese das Problem anhand der Variablen oben, unten,
         * Dabei wird die matrix verwendet, um die Temperaturentwicklung der Mess
         * In dieser Klasse ist das Problem sequentiell zu loesen.
         */
        public void solve() {
            (new Thread(new SequentialSolverThread(this))).start();
        }

        public class SequentialSolverThread extends Thread {
            /**
             * Solver, to refresh UI
             */
            protected final Solver solver;
            private boolean abbruch = true;

            public SequentialSolverThread(Solver solver) {
                this.solver = solver;
            }

            public void run() {
                //Wiederhole bis jede Berechnung innerhalb von epsilon liegt
                do {
```

```
                abbruch = true;
                //Durchlaufen der Matrix in x-Richtung
                for (int x = 0; x < matrix.getSize(); x++) {
                    //Durchlaufen der Matrix in y-Richtung
                    for (int y = 0; y < matrix.getSize(); y++) {
                        //Berechnen des Wertes
                        double result = solve(x, y);
                        //Prüfen ob der Wert innerhalb von espilon liegt
                        double change = result - matrix.getValue(x, y);
                        if (change > epsilon) {
                            abbruch = false;
                        }
                        //Speichern des neuen Werts in der Matrix
                        matrix.setValue(result, x, y);
                    }
                }
                //Starten der nächsten Iteration
                matrix.nextIteration();
            //Wenn ein Wert die Grenzen von epsilon überschritten hat wird wi
            } while (!abbruch);
            //Wenn alle Werte innerhalb der Grenzen liegen wird die Berechnun
        }

        //Refactoring
        private double solve(int x, int y) {
            double result = getMatrixPointValue(x - 1, y) +
                    getMatrixPointValue(x, y - 1) +
                    getMatrixPointValue(x + 1, y) +
                    getMatrixPointValue(x, y + 1);
            result = result * 0.25;

            return result;
        }

        //Beachten der Grenzen von x und y und rückgabe
        private double getMatrixPointValue(int x, int y) {
            if (x < 0) {
                return links;
            } else if (x > matrix.getSize() - 1) {
                return rechts;
            } else if (y < 0) {
                return oben;
            } else if (y > matrix.getSize() - 1) {
                return unten;
            } else {
                return matrix.getValue(x, y);
            }
        }
    }
}
```