```java
package main;

import exception.NumberMismatchException;
import lombok.Getter;

import java.io.Serializable;
import java.util.*;
import java.util.stream.Collectors;

@Getter
public class Neuron implements Serializable {
    private List<Double> weights;
    private List<Double> weightChanges;
    private double[] inputs;
    private double output;
    private double errorSignal;
    private boolean isBipolar;

    private static final double WEIGHT_UPPER_BOUND = 0.5;
    private static final double WEIGHT_LOWER_BOUND = -0.5;

    /***
     * Constructor of the Neuron
     * @param numWeights: number of weights that link to this neuron
     */
    public Neuron(int numWeights, boolean isBipolar) {
        this.weights = Arrays.stream(new double[numWeights]).boxed().collect(Collectors.toList());
        Random random = new Random();
        double range = this.WEIGHT_UPPER_BOUND - this.WEIGHT_LOWER_BOUND;
        this.weights = this.weights.stream().map(x -> x = random.nextDouble() * range + this.WEIGHT_LOWER_BOUND).collect(Collectors.toList());
        this.output = -1;
        this.weightChanges = Arrays.stream(new double[numWeights]).boxed().collect(Collectors.toList());
        this.isBipolar = isBipolar;
    }

    /***
     * The sum of this neuron based on the input X.
     * @param X: the inputs
     * @return the dot product of the X and weights
     */
    public double sum(double[] X) throws NumberMismatchException {
```

```
42          this.inputs = X;
43          if (X.length != this.weights.size()) throw new
    NumberMismatchException("");
44          double res = 0;
45          for (int i = 0; i < X.length; i++) {
46              res += X[i] * this.weights.get(i);
47          }
48          return res;
49      }
50
51
52      /**
53       * Zero out all the weights
54       */
55      public void zeroWeights() {
56          this.weights = new ArrayList<>(Collections.nCopies(
    this.weights.size(), 0.0));
57      }
58
59
60      /**
61       * @return the current output
62       */
63      public double getOutput() {
64          return output;
65      }
66
67
68      /**
69       * @return the error signal corresponding to the
    current weight
70       */
71      public double getErrorSignal() {
72          return errorSignal;
73      }
74
75
76      /**
77       *
78       * @param errorSignal: the error signals of the layer
    above
79       */
80      public void setErrorSignal(double errorSignal, double
    weight) {
81          this.errorSignal = isBipolar ? (this.output + 1
    ) * (1 - this.output) * errorSignal * weight * 0.5 :
82                  this.output * (1 - this.output) *
    errorSignal * weight;
```

```
 83     }
 84
 85     public void setOutput(double output) {
 86         this.output = output;
 87     }
 88
 89     public void setErrorSignalForOutputNeuron(double error
    ) {
 90         this.errorSignal = isBipolar ? error * (this.
    output + 1) * (1 - this.output) * 0.5 :
 91                 error * this.output * (1 - this.output);
 92     }
 93
 94
 95     public double getWeightByIndex(int i) {
 96         return this.weights.get(i);
 97     }
 98
 99     public void updateWeights(double momentum, double
    stepSize) {
100         for (int i = 0; i < this.weightChanges.size(); i
    ++) {
101             double curWeightChange = this.weightChanges.
    get(i);
102             double curWeight = this.weights.get(i);
103             double updatedWeightChange = momentum *
    curWeightChange + stepSize * this.errorSignal * this.
    inputs[i];
104             this.weightChanges.set(i, updatedWeightChange
    );
105             this.weights.set(i, curWeight +
    updatedWeightChange);
106         }
107     }
108 }
109
```