```
using System;
using System.Collections.Generic;

namespace Neural_Network {
    class Perceptron {
        public Layer inputLayer;
        public Layer outputLayer;
        public List<Layer> hiddenLayers;
        private double initWeightMin;
        private double initWeightMax;

        public Perceptron(List<int> numberOfNeurons, double initWeightMin, double initWeightMax) {
            inputLayer = new InputLayer(numberOfNeurons[0]);
            hiddenLayers = new List<Layer>();
            for (int i = 1; i < numberOfNeurons.Count - 1; ++i) {
                hiddenLayers.Add(new HiddenLayer(numberOfNeurons[i]));
            }
            outputLayer = new OutputLayer(numberOfNeurons[numberOfNeurons.Count - 1]);

            this.initWeightMax = initWeightMax;
            this.initWeightMin = initWeightMin;

            connectFully();
        }

        private void connectFully() {
            Random rand = new Random();
            for (int i = 0; i < hiddenLayers.Count+1; ++i) {
                Layer current;
                current = i < hiddenLayers.Count ? hiddenLayers[i] : inputLayer;
                Layer next;
                if(i < hiddenLayers.Count - 1)
                    next=hiddenLayers[i + 1];
                else if(i==hiddenLayers.Count-1)
                    next=outputLayer;
                else
                    next=hiddenLayers[0];
                foreach (Neuron from in current.neurons) {
                    foreach (Neuron to in next.neurons) {
                        if (to.GetType() == typeof(BiasNeuron)) {
                            continue;
                        }
                        Synapse s = new Synapse(from, to);
                        from.addOutgoingSynapse(s);
                        to.addIncomingSynapse(s, rand.NextDouble()*(initWeightMax-initWeightMin)+initWeightMin);
                    }
                }
            }
        }

        public List<double> feedForward(TrainingInstance tr){
            if (tr.inputVector.Count != inputLayer.neurons.Count-1) { //-1 due to bias neuron
                throw new Exception("input vector size does not match input layer neuron count");
            }

            for (int i = 1; i < inputLayer.neurons.Count; ++i){
                inputLayer.neurons[i].setStaticOutput(tr.inputVector[i-1]);
            }
            for (int i = 0; i < hiddenLayers.Count; ++i) {
                for (int j = 0; j < hiddenLayers[i].neurons.Count; ++j){
                    hiddenLayers[i].neurons[j].calc();
                }
            }
            List<double> results = new List<double>();

            for (int j = 0; j < outputLayer.neurons.Count; ++j) {
                outputLayer.neurons[j].calc();
                results.Add(outputLayer.neurons[j].getCurrentOutputValue());
            }
            return results;
        }
    }
}
```