a) Formation of Neural Network for OR, AND using Threshold function

Traincontroller.java

```
package controller;

import model.Train;
import service.TrainService;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.sql.SQLException;

@WebServlet(name = "TrainServlet")
public class TrainController extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
        Train train;
        int[][] input = new int[2][4];
        for(int i=0;i<2;i++){
            for(int j=0;j<4;j++){
                input[i][j] = Integer.parseInt(request.getParameter("input"+i+j));

            }
        }
        /*for(int i=0;i<2;i++){
            for(int j=0;j<4;j++){
                System.out.println(input[i][j]);
            }
        }*/
        float[] weight = new float[2];
        for(int i=0;i<2;i++){
            weight[i] = Float.parseFloat(request.getParameter("initweight" + i));
        }
        /*for (float x : weight){
            System.out.println(x);
        }*/
        train=new Train(input,weight);

        TrainService service = null;
        try {
            service = new TrainService();
        } catch (Exception e) {
            e.printStackTrace();
        }

        Train trained = service.calculateWeight(train);

        float[] finWeight=trained.getWeights();
        int it=trained.getIt();

        try {
            service.storeWeights(finWeight);
        } catch (SQLException e) {
            e.printStackTrace();
        }
        /*System.out.println(finWeight[0]);
        System.out.println(finWeight[1]);*/

        request.setAttribute("weight1",finWeight[0]);
        request.setAttribute("weight2",finWeight[1]);
        request.setAttribute("iter",it);
        request.getRequestDispatcher("/view/train.jsp").forward(request,response);
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {

    }
}
```

Testcontrolelr.java

```
package controller;

import model.Test;
import service.TestService;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;

@WebServlet(name = "Test1Controller")
public class TestController extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
        TestService service = null;
        Test weights = null;
        try {
            service = new TestService();
            weights=service.getWeights();
        } catch (Exception e) {
            e.printStackTrace();
        }

        if (weights.getWeight()[0]==0){
            request.getRequestDispatcher("/index.jsp").forward(request,response);
        }
        else {
            int[] input = new int[2];
            String valid = request.getParameter("input0");

            if(valid==null){

request.getRequestDispatcher("view/test.jsp").forward(request,response);
            }
            else{
                for(int i=0;i<2;i++){
                    input[i] = Integer.parseInt(request.getParameter("input"+i));
                }
                float[] weight = weights.getWeight();
                //System.out.println(weights);
                weights = new Test(input,weight);
                //System.out.println(weights);
                Test output=service.calculateOutput(weights);

                //System.out.println(output.getOutput);
                request.setAttribute("output",output.getOutput());
                request.setAttribute("error",output.getError());

request.getRequestDispatcher("/view/testOutput.jsp").forward(request,response);
            }
        }
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
        //System.out.println("hoho");
        doPost(request,response);
    }
}
```

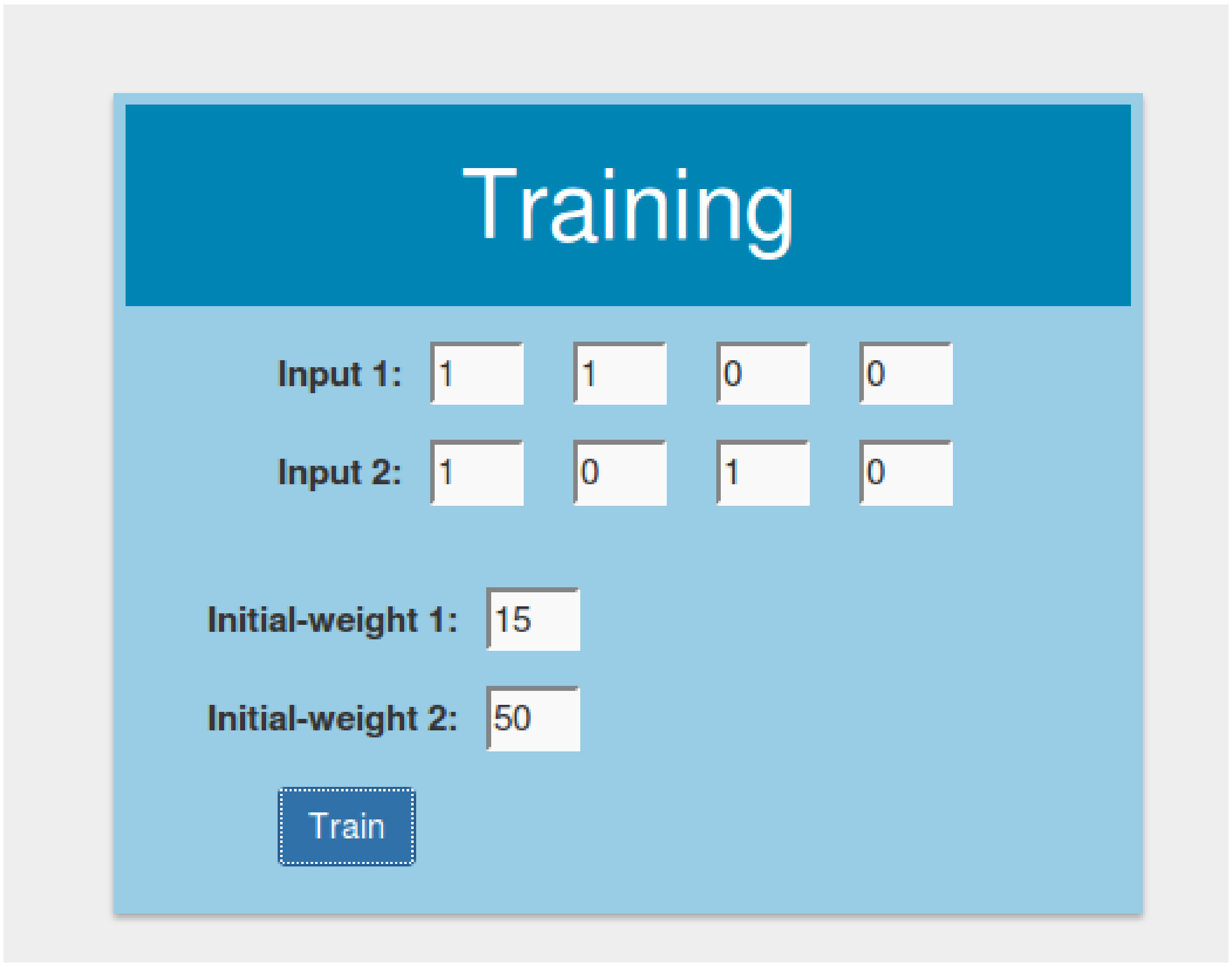
TestService.java

```
package service;

import model.Test;

import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

public class TestService {
    private Connection connection = null;
    private ResultSet resultSet = null;
    private Statement statement;
    private String query;
    private Test result = new Test();

    public TestService() throws Exception{
        connection = new Database().getDbConnection();
        statement = connection.createStatement();
    }

    public Test getWeights() throws SQLException {
        float[] weights = new float[2];
        query="select * from andweights";
        resultSet=statement.executeQuery(query);

        while (resultSet.next()){
            weights[0]=resultSet.getFloat("weight1");
            weights[1]=resultSet.getFloat("weight2");
        }
        System.out.println(weights[0]+" "+weights[1]);
        Test weight = new Test(weights);
        return weight;
    }

    public Test calculateOutput(Test test){
        int t=0;
        float out=0;
        int[] inp = test.getInput();
        float[] wt = test.getWeight();

        for (int i = 0; i <2 ; i++) {
            out+=wt[i]+inp[i];
        }
        for (int i = 0; i <2 ; i++) {
            if (inp[i]<0) {
                t=0;
                break;
            }
            else {
                t=1;
            }
        }
        float error=(t-out);

        result = new Test(t,error);

        return result;
    }
}
```

TrainService.java

```
package service;

import model.Train;

import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

public class TrainService {
    private Connection connection = null;
    private ResultSet resultSet = null;
    private Statement statement;
    private String query;
    private Train train = new Train();

    public TrainService() throws Exception{
            connection = new Database().getDbConnection();
            statement = connection.createStatement();
    }

    public void storeWeights(float[] weights) throws SQLException {
        query="delete from andweights";
        statement.execute(query);
        query="insert into andweights(weight1,weight2)
values("+weights[0]+","+weights[1]+")";
        statement.execute(query);
        connection.close();
    }

    public Train calculateWeight(Train inputs){
        float out;
        int[] y = new int[4];
        int[] t={1,1,1,0};
        boolean valid = false;
        int[][] input = inputs.getIp1();
        float[] weight = inputs.getWeights();

        for(int i=0;i<10000;i++) {
            for(int j = 0; j<4; j++) {
                out=0;
                for(int k =0; k<2;k++){
                    out += input[k][j] * weight[k];
                }
                y[j] = out >= 1 ? 1 : 0;
                /*System.out.println("y["+j+"]=="+y[j]);*/
            }
            for(int j=0;j<4;j++){
                if(t[j]==y[j]){
                    valid =true;
                }
                else{
                    valid =false;
                    break;
                }
            }
            if(valid){
                /*for(int j=0;j<4;j++){
                    //System.out.println("y["+j+"]=="+y[j]);
                    System.out.println(" y["+j+"]=="+y[j]+" t["+j+"]=="+t[j]+"
weight["+0+"]=="+weight[0]+" weight["+0+"]=="+weight[1]);
                }*/
                break;
            }
            else{
                train.incrIt();
                adjustWeight(weight);
            }
            //------------------------------------
        }
        Train trained=new Train(train.getIt(),weight);
        return trained;
    }

    private void adjustWeight(float w[]){
        int k;
        for(k=0;k<2;k++){
            if(w[k]>=1){
                w[k]-=0.5;
            }
            else{
                w[k]+=0.1;
            }
        }
    }


}
```

View-UI:


Training

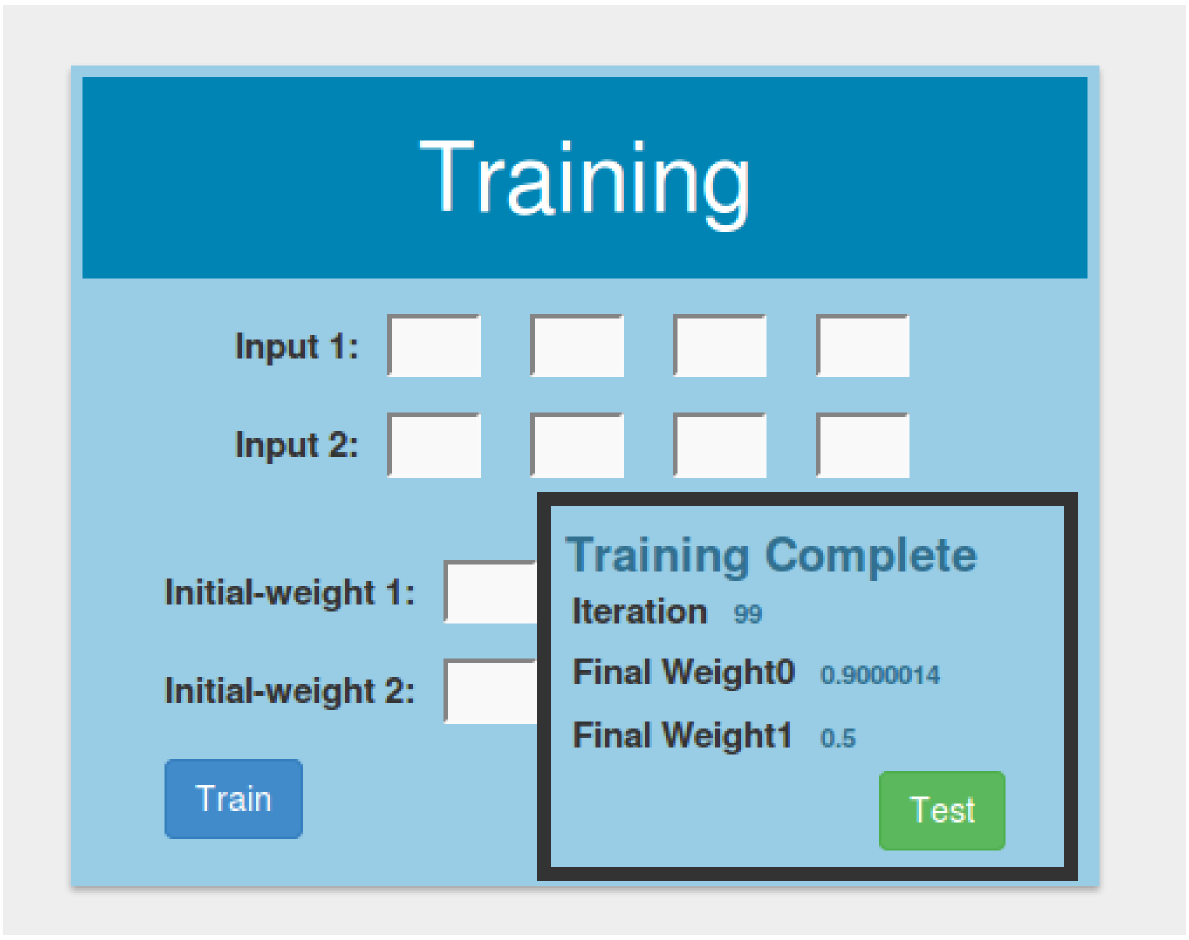
Input 1: 1 1 0 0

Input 2: 1 0 1 0

Initial-weight 1: 15

Initial-weight 2: 50

Inputs


Training

Input 1:

Input 2:

Initial-weight 1:

Initial-weight 2:

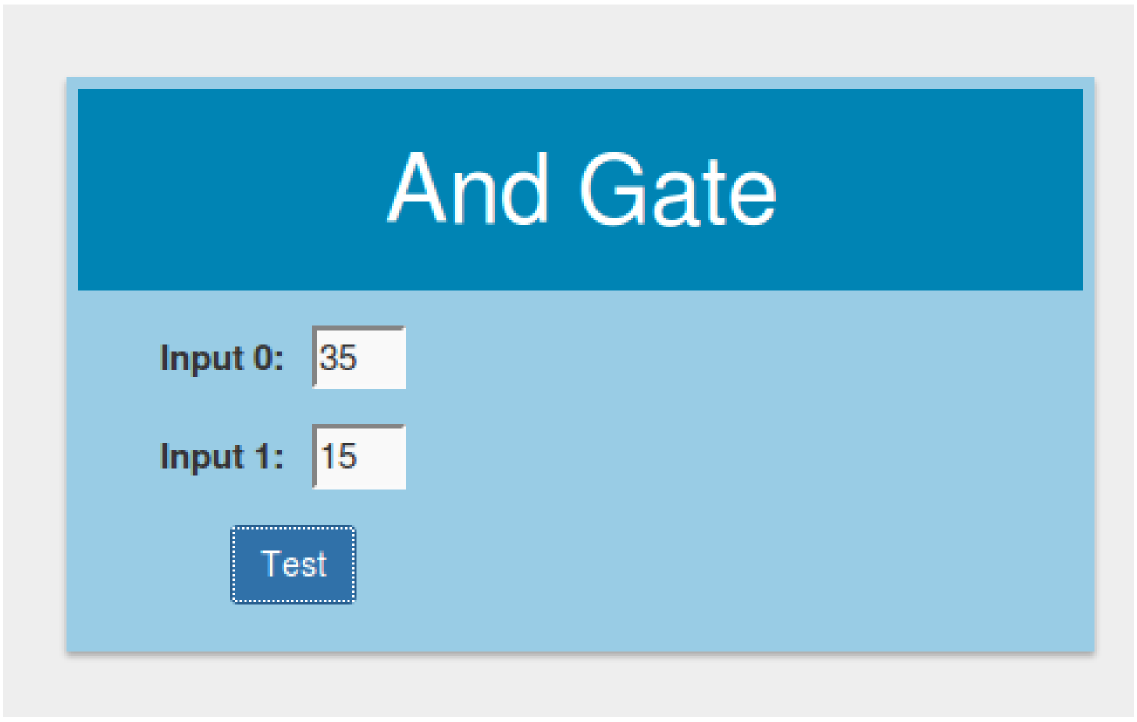
Training Complete

Iteration 99

Final Weight0 0.9000014

Final Weight1 0.5

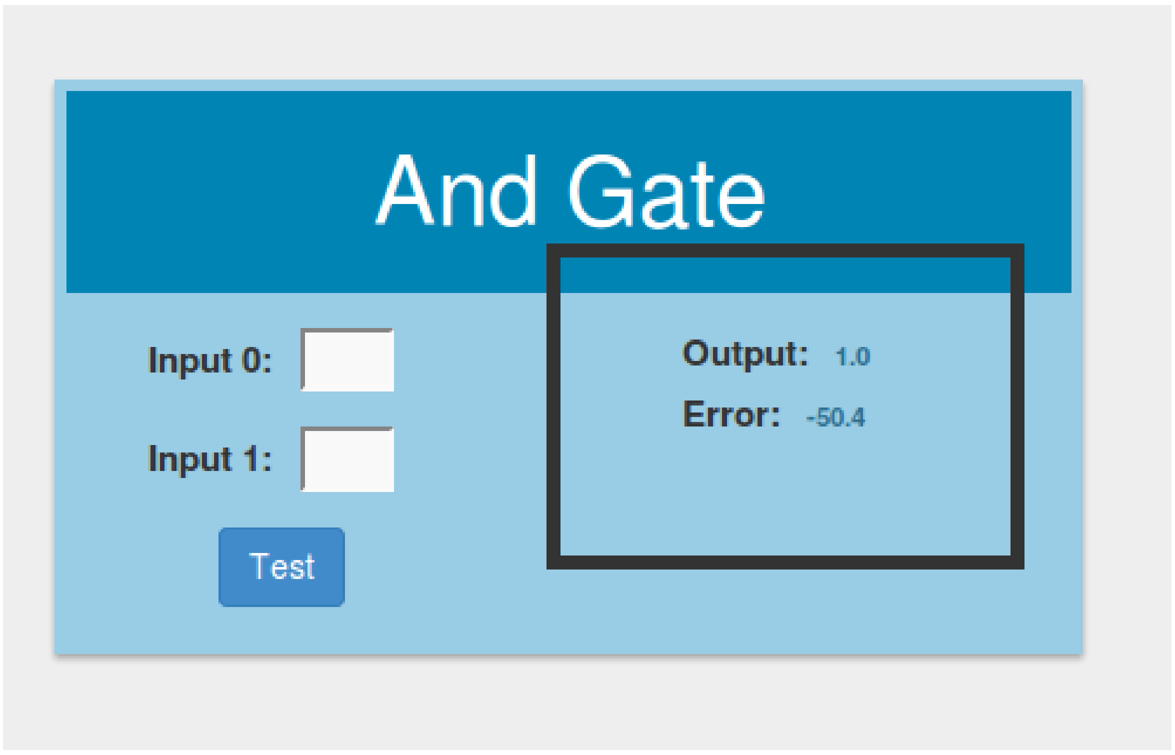
Training Complete


And Gate

Input 0: 35

Input 1: 15

Test for any imput



Output: 1.0
Error: -50.4

OutputGenerated